BUILDING BLOCKS FOR ADAPTABLE IMAGE SEARCH IN DIGITAL LIBRARIES

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philospohisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Michael Springmann
aus Achern, Deutschland

Zürich, 2014

Additional Notices:

As this thesis focuses on *image search*, it does contain many images and illustrations as these add many insights to verbal explanations. In Section 15.2.3 on page 477 I will give detailed credits. The individual images may have different licenses, some of them much more liberal than the ⓒⓘⓢⓔ. For many other images I got the permission from the copyright owner to use them – even if the license under which they were initially published would not be compatible. Only due to this, the thesis was possible in its current form.

If you intend or would like to use, reuse, remix individual parts of this thesis, please contact me: I'm willing to relax above mentioned restrictions for any part that I can.

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät

auf Antrag von

Prof. Dr. Heiko Schuldt, Dissertationsleiter
Prof. Alberto Del Bimbo, Korreferent

Basel, den 27.03.2012

Prof. Dr. Martin Spiess, Dekan

Zusammenfassung

Mit der Verfügbarkeit von einfachen und kostengünstigen Methoden zur Erzeugung und Speicherung von Bildern in digitaler Form ist die Menge an visueller Information dramatisch angewachsen, welche elektronisch aufbewahrt und geteilt wird. Bilder stellen ein wichtiges Medium dar um das menschliche Wissen, Erfahrungen und Gefühlen zu bewahren, auszudrücken und zu kommunizieren. Daher ist es wünschenswert oder sogar unausweichlich, dass digitale Bibliotheken nicht nur textuelle Informationen enthalten, sondern auch Bilder. Ein zentraler Aspekt digitaler Bibliotheken stellt die Fähigkeit dar, ihre Inhalte den Anwendern leicht zugänglich zu machen. Hierfür müssen also auch angemessene Suchmechanismen für Aufgaben angeboten werden, welche bildbezogene Suchen beinhalten. Traditionelle text- und metadaten-orientierte Ansätze reichen hierfür nicht, da für persönliche wie auch automatisch zusammengestellten Bildkollektionen gewöhnlich detaillierten Beschreibungstexten fehlen, welche in Suchvorgängen genutzt werden könnten.

Um Bildsuchen besser in digitalen Bibliotheken zu unterstützen benötigt es auch Methoden der inhaltsbasierten Bildsuche (engl. *Content-based image retrieval* oder kurz *CBIR*): CBIR bietet Mechanismen um Bilder alleine anhand ihres Inhalts zu finden und die Bilder anhand (visueller) Angaben des Anwenders zu vergleichen sowie die Resultate nach deren Ähnlichkeit zu diesen Angaben zu bewerten und ordnen. Das Ziel dieser Arbeit besteht darin, Bausteine zu identifizieren, zu erstellen und zu evaluieren welche genutzt werden können, um digitale Bibliotheken mit der Fähigkeit zu schaffen, Ähnlichkeitssuche für Bilder zusätzlich zu den traditionellen Ansätzen zu ermöglichen.

Diese Arbeit folgt einem Top-Down-Ansatz und umfasst drei Hauptbeiträge:

Als erstes führen wir ein neues Modell zur Charakterisierung von bildbezogenen Suchaufgaben –genannt *Image Task Model (ITM)*– ein. Dieses Modell integriert und verfeinert bestehende Modelle in ein einziges, präzises Modell für Interaktionsabsichten. Es bezieht die möglichen Angaben und Ziele des Nutzers (engl. *Task Input and Aim*), die erlaubten Abweichungen von den Eingaben (engl. *Matching Tolerance*) sowie die angestrebte Verwendung der Resultate durch den Nutzer (engl. *Result Usage*) ein.

Zweitens nutzen wir ITM um konzeptuellen Bausteine zu ermitteln, welche die benötigte Funktionalität zur inhaltsbasierte Bildsuche und Ähnlichkeitssuche im Allgemeinen in Digitalbibliotheken anbieten: Inhaltsverwaltung, Anfrageformulierung und -ausführung und Benutzerinteraktion. Diese konzeptuellen Bausteine und ihre wechselseitigen Bezeigungen und Abhängigkeiten werden untersucht und eine umfassende Übersicht zum Stand der Forschung auf diesem Gebiet erstellt, welche auf Grundlage von ITM die Fähigkeit zur Unterstützung bei Suchaufgaben von existierenden Ansätze bewertet und dabei Stärken und Schwächen aufzeigt.

Drittens präsentieren wir eine detaillierte Diskussion von ausgewählten Bausteinen zusammen mit unserer Implementierung dieser, welche die bestehenden Ansätze erweitert und verbessert um dadurch Bildähnlichkeitssuchen in Digitalbibliotheken besser zu unterstützen. Das Hauptprinzip welches wir hierzu verfolgen ist die Anpassung der erlaubten Abweichung der Benutzerangaben an die Bedürfnisse einer

Aufgabe, so dass existierende Bausteine für verschiedene Anwendungsgebiete wiederverwendet und optimiert werden können. Diese Wiederverwendbarkeit demonstrieren wir anhand prototypischer Implementierungen von kompletten Digitalbibliothek-systemen für drei unterschiedlichen Anwendungsgebieten: automatische Klassifikation medizinischer Bildern, Suche bekannter Bilder mittels gezeichneter Skizzen und das nachträgliche Zuweisen von Geokoordinaten zu Bildern.

Diese Arbeit unterstützt somit die zukünftige Entwicklung von digitalen Bibliotheken welche Bildsuchfunktionalität anbieten beginnend mit dem frühen Stadium des Erfassens der Nutzeranforderung durch die Charakterisierung der Benutzeraufgaben mittels ITM, über die Auswahl geeigneter konzeptueller Bausteine welche die benötigte Funktionalität bieten, bis schliesslich hin zur Implementierung kompletter Systeme welche in der Wiederverwendung existierender Bausteine gipfeln kann.

Abstract

With the availability of easy and inexpensive methods to create and store images in digital formats, the visual information preserved and shared electronically has grown dramatically. As images are important means to archive, express, and communicate human knowledge, experience, and feelings it is desirable or even unavoidable that digital libraries do not contain only textual information, but also such images. One central aspect of digital libraries is the ability, to make its content easily available to its users and therefore also to provide adequate retrieval mechanisms for image-related search tasks. Traditional text- and metadata-based approaches are not sufficient as personal digital libraries as well as automatically acquired image collections commonly lack detailed descriptions that could be used in searches.

To better support image search in digital libraries also methods from content-based image retrieval (CBIR) are needed: CBIR provides mechanisms to search for images by using the image content itself and compare the images with (visual) input the user provides and ranking the results based on similarity. The aim of this thesis is to identify, implement, and evaluate building blocks that can be used to build digital libraries with the ability to perform similarity search for images in addition to traditional approaches.

This thesis follows a top-down approach and has three main contributions:

First, we introduce the Image Task Model (ITM) to characterize the user's intention in image-related search tasks. This new model integrates and refines pre-existing models into one concise model for interaction intentions. It considers the user's *Task Input and Aim*, *Matching Tolerance*, and intended *Result Usage*.

Second, we use ITM to identify conceptual building blocks that provide the required functionality in digital libraries to support CBIR and similarity search in general: *Content Management*, *Query Formulation and Execution*, and *User Interaction*. These conceptual building blocks and their interactions are analyzed and a comprehensive survey reviews state-of-the-art approaches to which extent they can support search tasks on the basis of ITM to identify strong and weak spots.

Third, we present a detailed discussion of selected building blocks together with our own implementations that extend and improve state-of-the-art approaches to better support similarity searches for images in digital libraries. The key principal that we follow is adjusting the matching tolerance to the needs of a task, such that existing building blocks can be reused and optimized for different application domains. To demonstrate the reusability, we show prototypical implementations of complete digital library systems based on our building blocks for three different domains: automatic classification of medical images, sketch-based search for known images, and retrospective geotagging of images.

This thesis therefore supports future development of digital libraries with image search functionality from the early stage of understanding the user requirements through characterizing user tasks in ITM over the selection of appropriate conceptual building blocks for providing the required functionality to finally implement entire systems with the potential to reuse existing building blocks.

Contents

List of Figures

List of Tables

Part I

Foundation

1

Introduction

The aim of this thesis is to provide building blocks, that can be used to build digital libraries to provide adequate support for image-related search tasks in digital libraries. For this, one needs to understand the user's needs and basic search strategies. In Part I we will derive a model to characterize image-related search tasks. This model will be used in the subsequent parts to identify building blocks for digital libraries, and then provide and evaluate implementations of these building blocks that can be used in digital library systems for various domains. The key consideration for the building blocks is, that they shall allow reuse in different applications and we will present three prototype applications that reuse the developed building blocks for automatic classification of medical images, sketch-based search for known images, and retrospective geotagging of images.

This chapter will introduce the main terms and challenges that currently exist in the domain of managing images in digital libraries. Section 1.1 will describe digital libraries and how they did and still have to evolve in the presence of massive amounts of digital images. Section 1.2 introduces our approach of identifying conceptual building blocks that are needed to build or extend digital libraries with content-based access methods to their content. Section 1.3 will motivate why we believe that a detailed analysis of user tasks is necessary to be able to design reusable building blocks instead of focussing just on how to handle individual queries. Section 1.4 provides some example scenarios to illustrate some of the problems and potential solutions in a less abstract way. Finally, Section 1.5 will describe the structure of the next chapters and highlight the contribution of this thesis.

1.1 Digital Libraries and Images

Since the introduction of computers, the way we store information and we interact with it has changed significantly and is still in change. Looking at libraries, the first step was certainly to simply digitize the library catalog and provide text query interfaces to it.

Instances of such catalogues can be found at almost every university library as an OPAC (Online Public Access Catalogue), e.g. the local catalog of the university library

in Basel[1]. The federation of such library catalogues made it easier to retrieve documents independently of their physical location. In a next step, not only the bibliographic information about a book or article has been stored, but also an abstract or even the full text. Obvious examples of this category are MEDLINE[2] or the ACM Digital Library[3].

The development is certainly not limited exclusively to libraries and information retrieval is not limited exclusively to text documents. For instance, by 1933 the definitions of "document" began to include explicitly museum objects [Jörgensen, 2003, p. 2]. Given the fact that images and paintings have been the first documented communication between human beings that is still preserved and still is used, it is only natural to include images as well in the documents we want to include in digital libraries.[4]

But a *Digital Library (DL)* does not only consist of its content. [Borgman, 1997] describes digital libraries as two complementary ideas.[5]

> 1. Digital libraries are a set of resources and associated technical capabilities for creating, searching, and using information. In this sense they are an extension and enhancement of information storage and retrieval systems that manipulate digital data in any medium (text, image, sounds; static or dynamic images) and exist in distributed networks. The content of digital libraries include data, metadata that describe various aspects of the data (e.g., representation, creator, owner, reproduction rights), and metadata that consist of links or relationships to other data or metadata, whether internal or external to the digital library.
> 2. Digital libraries contain information collected and organized on behalf of a community of users and provide functional capabilities to support information needs and uses of that community. They are a component of virtual communities on which individuals and groups interact with each other and with data, information, and knowledge resources and systems. In this sense they are an extension, enhancement, and integration of a variety of information institutions as physical places where resources are selected, collected, organized, preserved, and accessed in support of a user community. These information institutions include, among others, libraries, museums, archives, and schools. They are also an extension of the environments in which information is created, used, and managed, including classrooms, offices, laboratories, homes, and public spaces.

As it is necessary to provide a technical infrastructure for any digital library, a novel software category named *Digital Library Management System (DLMS)* [Del Bimbo et al., 2004, Ioannidis et al., 2005] has emerged for providing a generic foundation that can be used and adapted for individual installations.

[1] `http://aleph.unibas.ch/`

[2] `http://www.ncbi.nlm.nih.gov/pubmed/`

[3] `http://portal.acm.org/dl.cfm`

[4] To use OPAC-like access to images, sometimes the term IPAC (Image Public Access Catalogue) is used.

[5] Another, but similar definition is given in [Witten and Bainbridge, 2010]: "a digital library is defined as a focused collection of digital objects, including text, video, and audio, along with methods for access and retrieval, and for selection, organization, and maintenance of the collection."

DLMS have been characterized in [Ioannidis et al., 2008] as:

> A DLMS is a software system that captures all common management aspects of DLs, and supports the easy development of digital library systems (DLS) pretty much in the same way a database management system (DBMS) facilitates building and managing databases and database applications. Ideally, a DLMS should make available to the developer an environment to create a DLS and at least a basic set of DL functionality. Any further application-specific requirements on content manipulation or user interfaces should be developed in a customized fashion on top of DLMSs.

Today, not only institutions can collect significant amount of data: *Personal Digital Libraries* mean bodies of information that are mostly of importance to individuals or small groups [Graham et al., 2002]. This is in particular of interest when considering images.

Over the last years, digital cameras and flash memory cards with capacities of several gigabytes have become increasingly popular. This has led to a very rapid growth of (personal) photo collections, both in terms of sheer size and numbers of objects. As the capabilities to not only store but also search and use information are integral aspects of digital libraries, searching in these large collections has become an important challenge. Browsing through a collection, which used to be the most common approach, is no longer a satisfactory solution as it does not scale with the increasing collection sizes.

Usually, searching for images requires a high degree of manual activity from a user: Either she has to properly organize her digital photos on the hard disk (using meaningful folder structures and/or file names that characterize the images' content) or she has to annotate each photo with a set of metadata tags that properly describe the image content. Both tasks are very time consuming. In addition, good results are extremely difficult to achieve as future queries need to be anticipated already at the time a photo is tagged in order to make sure that it will actually be found.

Significant parts of these personal image collections are no longer kept private. They are shared over the internet and form huge repositories. While a few years ago, stock photo collections on CD-ROM like Corel Stock Photo Library formed the biggest photo collections that were easily accessible[6], today even Getty Images[7] and Corbis[8] are outnumbered easily by the images shared in photo sharing communities and social networks of the so-called Web 2.0 and its user generated content.[9]

Flickr[10], a popular photo sharing site that was founded in February 2004, announced in October 2009 that they have reached 4 billion images [Champ, 2009] – less than one year after they announced the 3 billions picture [Champ, 2008]. Flickr Commons[11] is

[6] The Corel Stock Photo Library has been used frequently for Image Retrieval Benchmarks in earlier days for this particular reason, but it basically disappeared from the market

[7] `http://www.gettyimages.com/`

[8] `http://www.corbisimages.com/`

[9] A fact, that has raised the interest of Getty Images to collaborate with Flickr to provide a selection of images also inside their stock photo portfolio.
See `http://www.flickr.com/help/gettyimages/`, `http://www.flickr.com/groups/callforartists/` and `http://www.gettyimages.com/Creative/Frontdoor/Flickr`.

[10] `http://www.flickr.com`

[11] `http://www.flickr.com/commons`

a subsection of Flickr that contains images for which museums, libraries, and archives want to ease the access, such as the U.S. Library of Congress [Oates, 2008] as a pioneering partner amongst others[12].

Another website launched in 2004 collecting media files, in particular for which are public domain and freely-licensed educational media content, is Wikimedia Commons[13], which also received significant contributions from museums, libraries, and archives – for instance the contribution of 250'000 from the Deutsche Fotothek[14]. It holds also the repository of material used inside the free encyclopedia Wikipedia[15]. Compared to Flickr, it is rather small in size with more than 6 million files [Wikimedia Foundation, 2010]. This is very well in line with the aim to provide content that can be used in education; not to provide a space to present arbitrary images to everyone (as Flickr would do). Therefore it is more selective and frequently annotated in more details.

Currently, the biggest and fastest growing photo sharing website is Facebook with more than 15 billion photos and 220 million new photos per week [Vajgel, 2009].[16] Unlike Flickr, it did not start as a photo sharing platform in 2004, but as a social network with the Photos application being added as a (very popular) feature [Beaver, 2007]. And due to its bigger focus on social aspects, by far not all images are accessible directly by public.

To enable search in such huge collections, frequently tags are used. By "crowdsourcing" the load of tagging images, e.g., using the ESP Game [von Ahn and Dabbish, 2004] or in the context of sharing photos in social networks like Flickr or Facebook, the workload can be distributed. However, two severe problems remain even when tagging of shared images can be distributed:

- The tags associated in the first case might not be specific enough to handle very precise queries whereas the tags associated with images in social networks are typically very subjective, do not follow an established terminology and are thus not always usable for searches [Bischoff et al., 2008]. Such tags can be sufficient for searching particular concepts in an image like a person you know by name, particular mountains, famous buildings, or any sunset — but it reaches its limits when a name is not known.

[12] A list of partnering institutions can be found at:
`http://www.flickr.com/commons/institutions/`.

[13] `http://commons.wikimedia.org`

[14] `http://commons.wikimedia.org/wiki/Commons:Deutsche_Fotothek`

[15] `http://www.wikipedia.org`

[16] It's hard to get accurate, comparable numbers as one has to rely on the information provided by the image hosting platform – which usually does not cover precisely the same point in time and hard to verify, in particular as social networks usually provide functionality to preserve privacy of images such that they cannot be crawled from outside. [Schonfeld, 2009] tried to aggregate numbers in which ImageShack had the biggest number with 20 billion images in total, but with 100 million images per month a much smaller growth than Facebook and expected that Facebook would have more images by the end of 2009. In more recent posts, Facebook states that their users upload more than 100 million photos per day [Odio, 2010]. And with the increased maximum dimension of images [Odio, 2011], probably even more people will use Facebook for sharing images rather than dedicated sites like Flickr which allowed bigger sizes much earlier, in particular for paying Pro members.

- Moreover, photos shared with friends in a social network are usually small fractions of the overall collection a user has in her personal information space. Thus, a large number of photos is left without annotations anyways and looking for one particular image will very frequently terminate without success.

1.2 Building Blocks for Content-Based Image Retrieval

If no or not enough information about an image is known, ordering based on similarity can improve the situation (cf. [Wan and Liu, 2008]). For this, the content of the image itself is analyzed and compared to the content of a query image (*Content-Based Image Retrieval (CBIR)*) that the user has to provide as a positive example (*Query by Example*). In situations, where the user has no appropriate image at hand, letting the user draw a simple sketch can improve the situation (*Query by Sketch*). In both cases, the results can be presented to the user such that the most similar items are shown first (*Similarity Search*).

With the increasing amount of non-textual information, e.g. images, audio, video preserved in digital format, applications to manage and retrieve this content are needed. Rather than building such Digital Library applications from scratch one would like to reuse existing components. So far, the possibility for reuse of existing components has been limited: The individual components do not only have to be available for the desired platform, but also their internal model and APIs must match or have to be wrapped. In this thesis, we therefore use an approach on several layers of abstraction. We will analyze the requirements of digital libraries in order to provide content-based access like similarity search on a higher level of abstraction.

We use the term *Building Blocks* to refer to shared concepts that are independent of the concrete language or platform used for the implementation. Conceptual building blocks encapsulate functionality belonging to certain group. We will also describe how these building blocks interact and give insights in the implementation. The implementation of building blocks will further structure how functionality is provided, therefore introduce specialized building blocks to implement concrete, individual required functionality. While the interactions of building blocks can be analyzed already on the conceptual level, requires the evaluation of individual building blocks to be based on the specialized building blocks and their implementation.

1.3 Search Tasks vs. Query

In order to evaluate the utility of any system or individual component and/or judge whether a building block is needed and well suited, we need to understand the task that the user wants to perform. In this work, all tasks of the user will be related to searching. In order to successfully finish even a single task, the user may issue a sequence of queries.

It is important to notice that usually the system itself never has a chance to know what the actual task of the user is – its interface allows the user to perform operations

including issuing queries and it may be able to group such queries as a session. The user may stop a session without having completed a task, e.g., because the returned results do not seem helpful in fulfilling the task or because the task requires the use of more than one single system or application.

Technologically it is easy to create query logs and these can be analyze later in detail – but this can still only lead to a partial understanding of the user's task for the reason given before. There is plenty of literature related to Information Retrieval[17] that describes and evaluates:

- Query models and query languages as well as their expressiveness
- Potential optimizations to reduce the required execution time to perform queries and to present query results to the user
- Quality measures for query results and evaluation of systems considering these measures

All these aspects are important, but building systems only with respect to them is not sufficient to deliver satisfactory results. To improve usability of the system, everything needs to be considered that might affect the user in performing the search task (cf. [Nielsen and Phillips, 1993]) – which may include the user interface to issue queries, the layout of the query results and the time the user has to spend to assess them, and even the number of systems the user needs to consult to perform the task. This extension to tasks has been prominently requested in [Järvelin and Ingwersen, 2004] and user evaluations have shown that user satisfaction depends on more aspects than only the system effectiveness [Al-Maskari and Sanderson, 2010] and that a common quality measure like computing the precision of individual query results alone might not be a good indicator for the overall impression the system delivers [Turpin and Scholer, 2006, Smith and Kantor, 2008].[18] On the other hand, task-oriented measures like the task completion time are only meaningful to the extent the tasks selected for evaluation are representative [Khan et al., 2009]. It is therefore of utmost importance to understand the various (search) tasks that users may want to perform with the system on order to build a system that satisfies their needs and provides good user experience.

For instance, if the user is looking to find an image that she already knows, the search can end successfully as soon as she sees it in the result list. If the user is instead looking for the best picture to illustrate some aspect, it might be necessary to first consult a number of other related images before coming up with a conclusion – no matter how good the first results were. It might even be necessary that the system returns every image that satisfies a certain criteria, e.g., if at a public event pictures are taken and you want to ensure that the request "Please, no pictures of me." is obeyed by deleting all images that accidentally contain this person. Such different needs require different approaches in order to support the users well.

In days before the digitization when the physical presence of the searched material forced users to access the material in particular places, i.e., libraries for books, museums

[17] And such literature also exists in the domain of Database research.

[18] However, it is clear that the precision has impact and this has been shown in a controlled experiments in [Smucker and Jethani, 2010, Al-Maskari et al., 2008].

and galleries for paintings, and requests to stock photography agencies, there was still a chance to identify and note down the task that the user was working on due to the more immediate interaction. With the intermediate technology that now allows to get access to vast information resources online, it's still possible or even easier than ever to record individual queries, but it becomes harder to ask users about and assist them with their task. [Jörgensen, 2003, p. 121-134] reviewed several studies on how people search for images on a number of different image collections and archives. It cites from the earlier works of [Roddy, 1991]: "One of the great failures of image access at present is its inability to provide reliable information on what might be called a typical session" – just to add "a comment as true today as it was over a decade ago." [Jörgensen, 2003, p. 129]. [Enser, 2008, p. 535] added to that: "and it seems to the present author that the situation has not changed greatly in the interim."

In order to present a small subset of the possible variety of search tasks related to images, the next section will show some examples of what users might look for and illustrate in a less abstract way the challenges systems have to tackle to support the users well.

1.4 Example Search Problems

The following problems contain some examples of search tasks that do relate to visual information, but do not necessarily have an image as an result. Rather, the image can also be some input to the search or an intermediate result in the entire step.

1.4.1 Scenario 1: Fountains in Basel

Richard stands on Petersplatz in Basel looking at the fountain depicted in Figure 1.1. He gets interested in additional information about this particular fountain, e.g. when it was build, how it has been constructed, etc.

Of course, there are many ways to find such additional information. One possibility would be to look for a plate on the fountain that may contain information. Unfortunately, only very few –if any– of the approximately 170 fountains in Basel, carry such a plate. Another option would be to ask people passing by whether they know anything about this fountain – probably its name for a start. A third option would be to consult a book, like a tourist guide on Basel or dedicated books on architecture, buildings and in particular fountains in Basel like [Trachler, 1998]. Another very promising resource could be the Internet to look up information.

Whenever referring to external knowledge resources, it is important to verify that the found information really refers to the object. If there is an image of the fountain inside a book or on a website, Richard may compare this with the fountain to see, whether it is the right one.

If there is a software system supporting the user, it would be great if Richard could take a picture, e.g., with his cell phone, upload the image and obtain information of fountains from the area which also look similar to Figure 1.1.

Figure 1.1: Image of the fountain on Petersplatz in Basel, close to Petersgraben

To illustrate why this could be helpful, one can easily compare it to how one would probably search in a book for the fountain: Not knowing the name and therefore not being able to look it up in the index, one would probably limit the search to the area (to best knowledge and if the book is organized that way) and then skim through the pages and stop on images that might be similar to further refine the search. Applying this strategy to common online information sources does not yield satisfactory results if applied without very careful selection of sources.

For common web search engine, it is necessary to carefully select the keywords. And beware: The term "Petersplatz" most commonly refers to St. Peter's Square in Rome where there is a also a fountain.[19]

Using a dedicated search engine for images like Google Images[20], Microsoft Bing Image Search[21] or Yahoo! Images[22] do provide faster feedback as one can skip any returned image that is obviously not similar without having to follow a link first. Figure 1.2 shows a typical search result. The most-promising looking results from the picture sharing site Flickr show the right fountain, but unfortunately do not give additional

[19] In fact, the fountain by Gian Lorenzo Bernini on St. Peter Square in Rome is certainly much more famous. Even when combining it with the term Basel, many of the search engine hits refer to tourists from Basel that took pictures of the fountain in Rome on their vacation.

[20] `http://images.google.ch/`

[21] `http://www.bing.com/?scope=images`

[22] `http://images.search.yahoo.com`

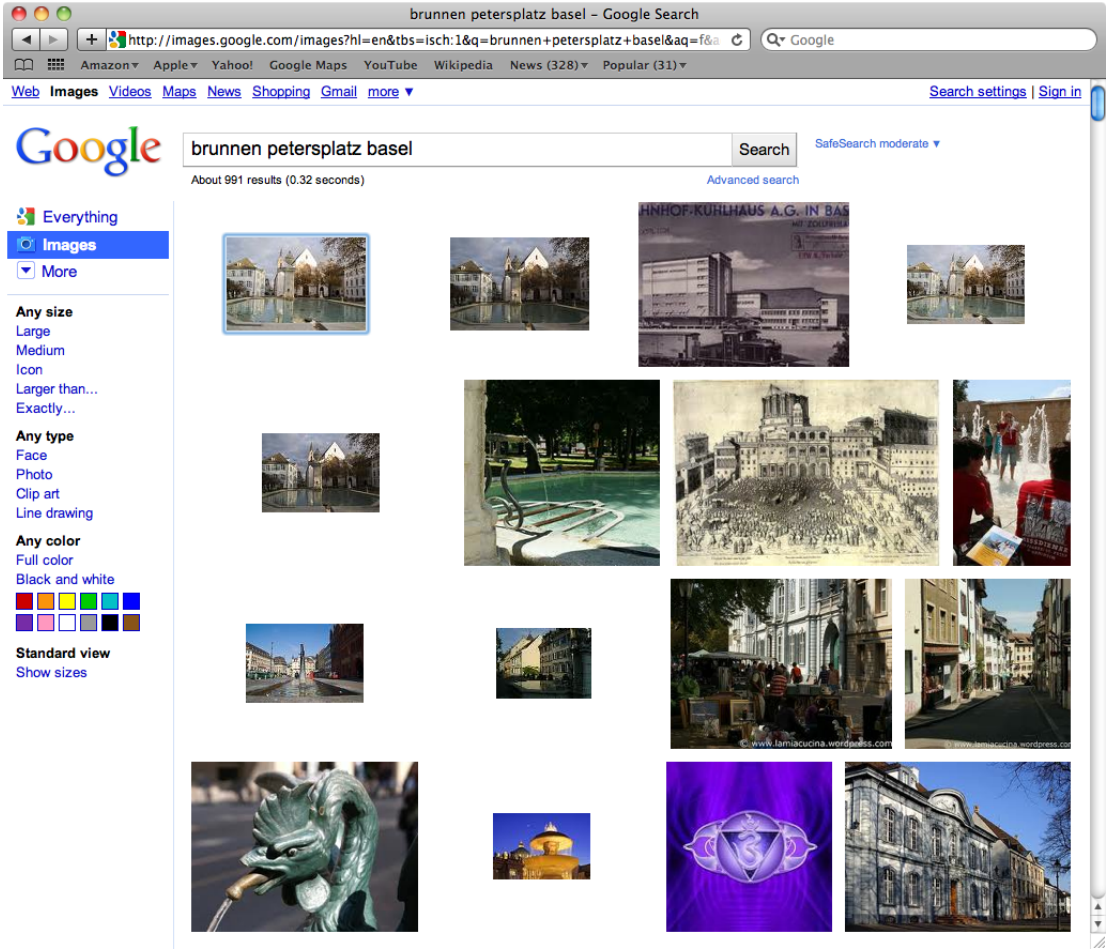


Figure 1.2: Search results of Google Images for query "brunnen petersplatz basel", Search performed on October 13, 2010

information as the description, tags, and comments at the time of writing do not contain any novel information that could help Richard in his task (cf. Figure 1.3).

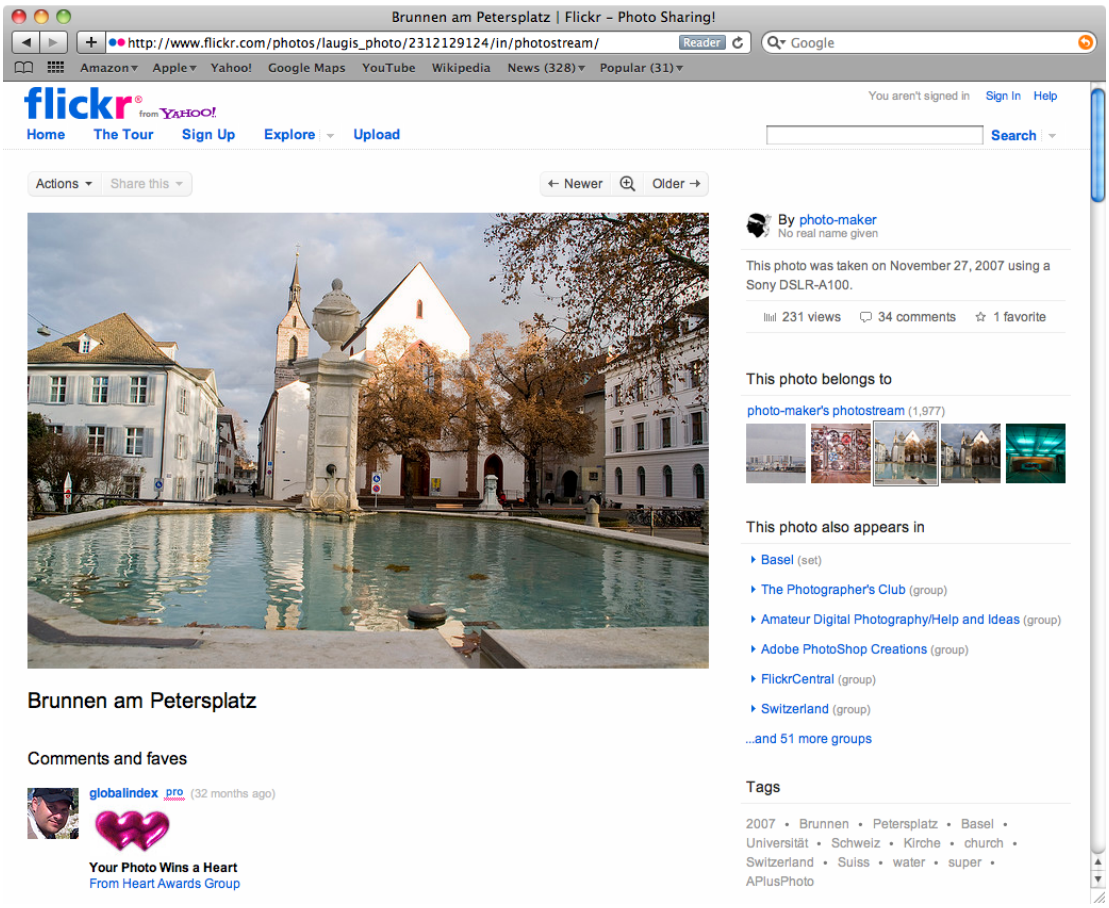


Figure 1.3: Another image of the same fountain found on the web at `http://www.flickr.com/photos/9655482@N08/2312129124/` (Accessed on October 13, 2010) – first result of text search of Figure 1.2

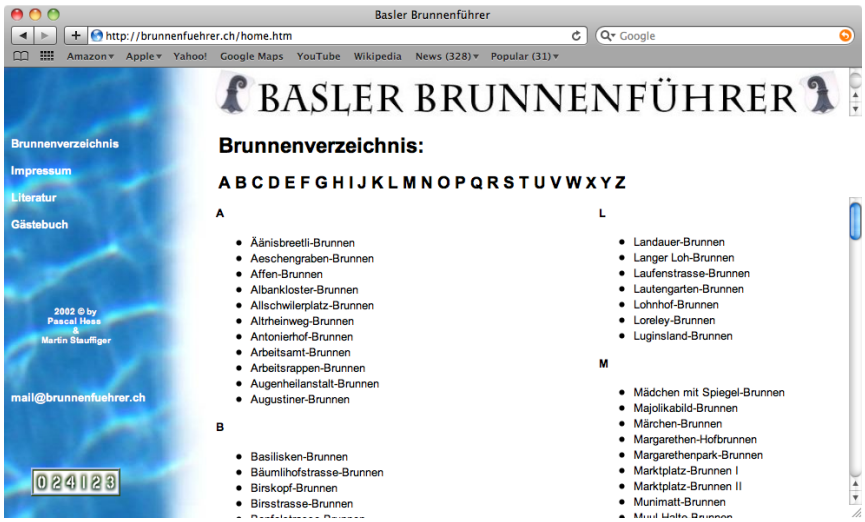


Figure 1.4: Alphabetic list of fountains on `http://brunnenfuehrer.ch/` (Accessed on October 13, 2010)

Investing more time into the search, one may come across the website Brunnenfuehrer.ch[23], which provides a list of fountains in Basel in alphabetic order and a detailed description of each. The entire site has in total about 400 images of the approximately 170 fountains, which would result in worst case having to navigate to 170 subpages.[24]

Figure 1.5 shows the desired result: a web page giving additional information about the fountain, including its name "Grabeneck-Brunnen", that it was built in 1779, and that is under protection of historical monuments since 1915.[25]

Reordering the images of the website based on similarity could ease the task for the user: As shown in Figure 1.6, with an image as input, the user may stop comparing images at rank 5.[26]

[23] `http://brunnenfuehrer.ch/`

[24] In order to avoid this, tools like Google Images provide parameters like "site:brunnenfuehrer.ch" to limit to results from this website. This can ease browsing through the images, but in worst case, the user would still have to watch up to 400 images to find the desired fountain.

[25] The free encyclopedia Wikipedia, as a common source of information on the internet, also mentions the fountain and when it was built on the German page about the Petersplatz in Basel. But it does not show a picture or its name or link to any further information about this particular fountain in the version accessed on October 13, 2010 `http://de.wikipedia.org/w/index.php?title=Petersplatz_(Basel)&oldid=76684704`.

[26] There is a Google Labs project called "Similar Images". It has recently been integrated into Google's image search. So far, it is only available for some images and only in combination with text search. Similar functionality is also available in Microsoft Bing Image search: when a text search returned results and moving the mouse over one result, similar images can be returned. Both services so far do not allow to upload images for search; Google Goggles `http://www.google.com/mobile/goggles/` allows this for mobile phones - for some platforms as a standalone application, for some integrated into the Google Mobile App `http://www.google.com/mobile/google-mobile-app/`. As of the date of writing, Google Goggles was not able to find the fountain.

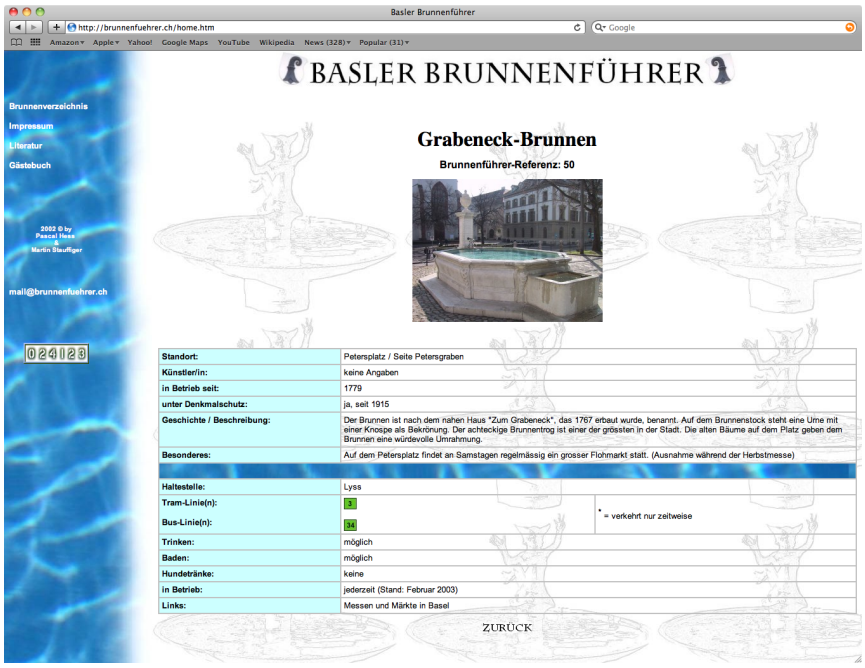


Figure 1.5: Details of fountains "Grabeneck" on `http://brunnenfuehrer.ch/` (Accessed on October 13, 2010)

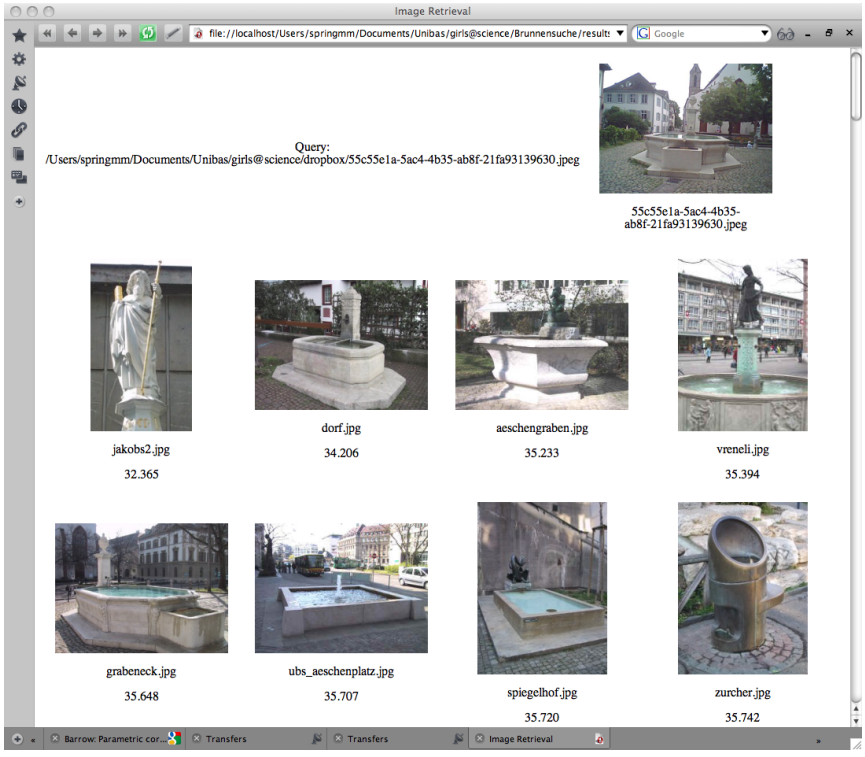


Figure 1.6: Ordering the images based on similarity with the feature Color Moments with 3x3 non-overlapping rectangles: Grabeneck is listed on rank 5 out of 400.

(a) First beach picture found (b) An image showing a beach at sunset

Figure 1.7: Images showing beaches at sunset

1.4.2 Scenario 2: Beach at Sunset

Tom and Carol are getting married. The entire family is busy with organizing everything. Tom's parents want to pay for the honeymoon as a wedding gift. In order to present this in a nice way, Tom's sister Christina was asked to design a gift voucher with a "personal touch".

Christina is now looking for images to illustrate nicely the idea of a romantic vacation. Sandy beaches at sunset, that would be something nice! In order to use an image that is original and available in a resolution that is sufficient for a printout in decent quality, she searches her own collection of images. The first image she finds is Figure 1.7(a), which shows her answering a phone call at the beach. It is not ideal, but probably there are more images of beach scenes. Figure 1.7(b) shows an example that's she is happy with: It just shows the beach, the sand, the sunset: That's much better! And there are a couple of more images like this.

Christina starts designing the voucher. While doing so, she realizes that she will have to do a little bit of image editing, e.g., to crop the image such that it fits nicely in the format of the voucher, alter the colors a bit to make them look nice in print, but also to make sure that there's enough room for placing text. She is familiar with such tasks, but not to an extent where this would work without some time consuming trial and error. In order to save time, she looks several times back at the image collection and tries out which image can be taken with the least effort in image editing to achieve the design she wants.

Like in the previous example, if tools are available to display images based on similarity, this can break up the original order in which they are stored. So instead of a rather common order that follows the time and events at which the images were taken or transferred from the camera to the computer, images will get displayed next to each other which share, for instance, similar color distribution that is typical for sunsets – independent of whether they were taken on the some occasion or probably in different vacation in different years.

(a) Car before wedding (b) Car driving away

Figure 1.8: Subset of wedding pictures showing the car

1.4.3 Scenario 3: Car at Wedding

Mark and Steve have both been to Tom and Carols wedding quite a while ago. They meet some time later and chat again about one topic they both like: cars. Both of them remember that there was a nice old car at the wedding in which the newlyweds were driving away, but they cannot recall what particular make and model it was.

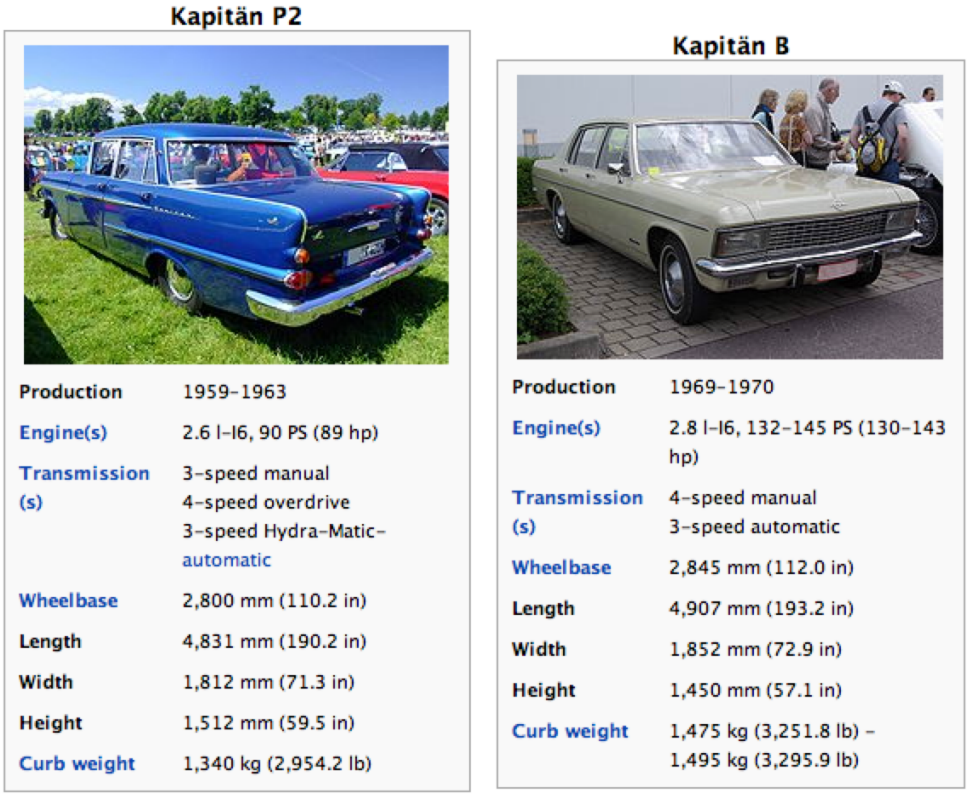
Steve knows that he has taken a picture of this car before the wedding ceremony. So he decides to look through his collection of 300 pictures that were taken at the wedding. After some time he finds the one showing the parked car (cf. Figure 1.8(a)). It shows very well the emblem with a flash which clearly identifies the car as an Opel. So now they know the make, but still not the model.

Chris asks whether Steve didn't also take a picture when the freshly married couple were driving away, so that they can have a look at the back of the car. Steve is not completely sure about this, but they can look if they find any images showing the car that way. In order to jump over all the pictures taken during the ceremony and not to jump to far to the party that took place afterwards at a different place, Steve switches to a thumbnail view in which only small previews of the images are shown. Just by browsing quickly through these thumbnails, Steve can ignore all the many images of the ceremony and the couple leaving the building as they are even on first glance and in the reduced size very different in colors and overall appearance from the images of the car. The next similar pictures of the car are taken when the couple is riding away (cf. Figure 1.8). The model name is written on the side, not ideally readable, but starts with "Kap". Looking up the list of models that Opel did produce at Wikipedia[27], the model must be the Opel Kapitän.

By comparing with the images on the Wikipedia site for that model[28] - the question can be answered: It was a Opel Kapitän P2, which was build between 1959 and 1963 as shown in Figure 1.9(a).

[27] http://en.wikipedia.org/w/index.php?title=Category:Opel_vehicles&oldid=386782518

[28] http://en.wikipedia.org/w/index.php?title=Opel_Kapit.C3.A4n&oldid=361525224



(a) Same model embedded in Wikipedia Infobox (b) Different model embedded in Wikipedia Infobox

Figure 1.9: Information that can be found on Wikipedia about Kapitän models

Tools for similarity search could help the user also in this example to reduce the need to browse through images, but order them in a more helpful way: For instance, starting from one image of the car at the wedding, arrange the other pictures of the wedding in a way to show images that may also contain the same car due to visual similarity next to the first example.

1.4.4 Summary of scenarios

What we have seen in the three scenarios is, that the tasks users perform with images might not be restricted just to the problem of finding some image. The complete task requires to derive information from the image, consult information resources that are just linked to some of the images, or even embed and alter the image in new images and documents. All these aspects are very different from each other and may require to provide a tailored set of tools. It is highly unlikely that a single piece of software will ever be able to cover all aspects. Even when restricting to just the problem of assisting the user in aspects related with the retrieval of images required to work on the user's task, there are still many different and sometimes conflicting requirements if attempting to satisfy the search needs with a single system.

It is therefore important to first analyze the individual needs that the various kinds of searches impose. In a next step it should then be attempted to identify common functionality needed in several searches. As mentioned in the individual scenarios, all seem to have potential to benefit from similarity search. What is also common to all scenarios is the general process in which any similarity search is performed: It starts with a dataset in which the user searches for images and the formulation of a query to express the user's needs for the current task. Then the similarity of the images in the dataset in relation to the query is determined to generate a result. The user will review the result to identify and use images that satisfy the need. If needed, the process will be repeated with modified queries until results are good enough or the user gives up.

However, what will differ in each scenario is not only the dataset and query, but also each of the scenarios has a slightly different notion of what the best measure of similarity is, what information needs to be preserved to provide the relevant information, and what would be the best user interface to express the user's information need and present results.

When developing a digital library handling the retrieval of images, common functionality can be handled by building blocks, that should not be limited to just a single use case, but reusable and exchangeable to serve the needs in different use cases and different systems. This can best be achieved by having individual building blocks for implementing a particular functionality that can be combined to built complete systems.

1.5 Contribution

The contribution of this thesis will manifest on several levels:

In the remainder of Part I we analyze in Chapter 2 in detail image related tasks in the context of searches in digital libraries. It will identify different aspects of a task, group them schematically, and present the novel Image Task Model (ITM) to integrate these aspects into one (graphical) representation.

In Part II we identify functional building blocks that are needed to build a digital library system that handles aspects of image related tasks and to describe the interaction between the functional building blocks. We also review the state-of-the-art for this functionality and analyse their interaction with particular user tasks – for which ITM from Part I serves as a model.

In Part III we provide detailed information on how building blocks identified in Part II can be implemented. We present our own implementations of selected building blocks and provide evaluations of the benefits of our implementations to proof their suitability for the targeted user tasks and how they enhance the state-of-the-art.

In Part IV we highlight how building blocks have been used in the implementation of actual systems. This includes aspects on how the individual building blocks can get integrated and reused to build complete systems. Chapter 15 concludes this thesis.

2

Image-Related Search Tasks

As seen with the three example scenarios in Chapter 1.4 there are very different use cases in which users might search for images. The aim of this chapter will be to analyze in detail the requirements that exist in various use cases. This will ease the grouping of system functionality into building blocks in order to be able to build systems that satisfy the requirements that particular search tasks impose and satisfy the user's needs. To characterize the users' tasks as well as the functionality provided by building blocks, Sections 2.2–2.4 will focus on the important individual aspects and Section 2.5 will introduce our Image Task Model (ITM) to represent these aspects jointly.

But before taking a closer look on the individual search tasks, it might be worthwhile to begin with a look on the criteria that will determine the quality of search results in general. This will be the content of Section 2.1.

2.1 Assessing Quality of Results: Relevance and Ranks

In order to assess how good applications assist users in finding information it is essential to understand which information it is they need.

[Wilson, 1997, p. 553] lists the following types of *Information Needs*:

- need for new information
- need to elucidate the information held
- need to confirm information
- need to elucidate beliefs and values held
- need to confirm beliefs and values held

The individual information need will then lead to an *Information Seeking Behavior*, which –if successfully supported by the information system– will lead to *Information*

Use. Whether or not the user is satisfied with the information depends on the information need itself and may reflect on the need itself. [Wilson, 1981]

As a consequence, systems need to support the interaction intentions and corresponding information seeking strategies [Xie, 1997] that the users chose as their seeking behavior. The assessment not only needs to take the particular strategy but also the information need itself into consideration.

Therefore for each item in the collection one of the key questions will always be: Is it *relevant* to the user, the user's current information need, and to the current operation performed by the user as part of the seeking strategy?

Hence, we need to be able to separate each document $d \in Docs$ into one of the two classes or sets, the set of relevant documents Rel and the set of irrelevant documents which consist of $Docs \setminus Rel$. The system may not retrieve and present all documents to the user: only a subset of retrieved documents is presented as a result $Res \subset Docs$. The ideal result would contain all relevant documents and none of the irrelevant, in which case $Res = Rel$.

However, there are two ways in which the result can diverge from this ideal case.

1. The retrieved documents do not contain all relevant items, therefore in this case the following inequality holds:

$$|(Res \cap Rel)| < |Rel| \tag{2.1}$$

   In order to measure this property, one takes the proportion between the two set sizes, which is called *Recall* [Büttcher et al., 2010, p. 407] and presented in Equation 2.2.

$$\text{Recall} = \frac{|Res \cap Rel|}{|Rel|} \tag{2.2}$$

2. The other option is that not all retrieved documents are relevant, therefore in this case the following inequality holds:

$$|(Res \cap Rel)| < |Res| \tag{2.3}$$

   Again, in order to measure this property, one takes the proportion between the two set sizes, which is called *Precision* [Büttcher et al., 2010, p. 407] and presented in Equation 2.4.

$$\text{Precision} = \frac{|Res \cap Rel|}{|Res|} \tag{2.4}$$

Notice that there is always a tradeoff between achieving high recall and high precision, e.g., simply returning all documents of the collection would result in a recall of 1, but at the cost of low precision.

When evaluating how well systems support the user tasks in achieving their aims as presented in Table 2.1, relevance of items in the result is the first decisive factor.

The second decisive factor is, how much effort the user still has to invest in order to successfully finish the task. Ignoring the time it takes to generate the results and details of the user interface on how they present the results and just assuming it will present the results as some kind of list, the order of the results remains the aspect determining when search ends. When results are presented in a certain order, and $Res[k]$ is the k-th item in the result list, then k is called the *Rank*.

The rank has two effects: First of all and very intuitively, relevant items should be ranked high in the result list and if there are non-relevant items, they should be ranked low. Second, the rank can be used to define a cut-off value which represents the number of items the user is willing to look at before giving up.

With the addition of rank, the existing measures precision and recall can be extended, e.g., to *Precision at k Documents (P@k)* [Büttcher et al., 2010, p. 408], where the rank k is the cut-off value:

$$\text{P@k} = \frac{|Res[1..k] \cap Rel|}{|Res|} = \frac{|Res[1..k] \cap Rel|}{k} \tag{2.5}$$

To evaluate the quality of a system and to compare it to other systems, it is commonly not sufficient to do this just with a single query. In order to get a single value even when several queries are used to measure the quality, this is usually done by computing aggregated value over all queries, like the arithmetic mean used for the mean average precision (MAP) and mean reciprocal rank (MRR).[1]

When analyzing the image related tasks in more detail, different notions on which items are relevant and additional requirements on ranking will define which measures are meaningful to particular tasks and which are not.

If we consider use cases given in the three example scenarios in Chapter 1.4, each of them may consist of one or more particular search task. Only if we can break them down into particular search tasks with clear task aims, information to start the search with, and criteria whether the task was performed successfully, we will be able to assess whether the user needs are met by the search results.

[1] See [Büttcher et al., 2010, pp. 406–410] for details and [Robertson, 2006] for the proposal to use the geometric mean rather than arithmetic mean, as the average of average precision values usually does not give good intuition about the overall performance of a system. Notice that the Mean Reciprocal Rank mentioned here is not the same as the Modified Retrieval Rank used during the calculation of the ANMRR (Average Normalized Modified Retrieval Rank) [Manjunath et al., 2001, Chalechale et al., 2004, Springmann et al., 2010a]. ANMRR is yet another retrieval quality measure based on rank which has been proposed in the development of MPEG-7.
Another approach for evaluating CBIR systems based on ranks is the Rank-Difference Trend Analysis (RDTA) proposed in [Dimai, 1999a].

2.2 Task Input and Aim: User's Contribution and Desired Results

The tasks users want to perform have been of great interest already for traditional libraries storing physical artifacts like books and journals. [Xie, 1997] analyzed 40 cases of usage of libraries and the installed OPAC and Non-OPAC library systems. Out of this, the following seven different *Interaction Intentions* of interactive information retrieval have been derived:

1. *Identifying* refers to identifying information for what to be searched and often occurs when a user tries to consult personal sources before starting to search.

2. *Exploring* refers to a) find something interesting, or b) satisfy curiosity.

3. *Learning* refers to learning a) system function, b) system structure, but also c) system content. "System" includes any IR system in the library, such as OPAC, card catalog, and the structure of the library is considered part of the the IR system.

4. *Looking for known item(s)* refers to looking for item(s) for which a user has bibliographic information.

5. *Finding items with common characteristics* refers to finding items with either similar content or similar structure. Finding items with similar content occurs not only as the initial interaction intention, but also as an alternative approach – in particular when the search for a known item failed.

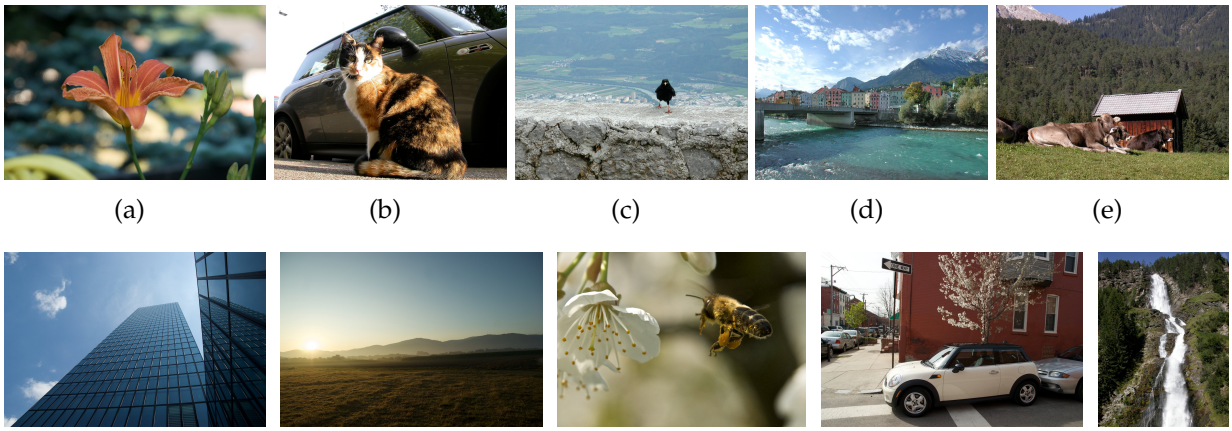
6. *Evaluating* refers to evaluating a) the relevance of item(s), b) correctness of item(s), c) duplication of item(s), or d) fitness of item(s).

7. *Locating* refers to locating either familiar or unfamiliar item(s). A familiar item is an item that the user saw before in a library, unfamiliar is an item whose location is not clear to the user.

In the context of *digital libraries* as opposed to the traditional libraries observed for that study, the linking between the entry in the library system can remove intermediate steps needed between finding a record in the system, e.g., the card in the catalog, and accessing the item itself, for which it had to be located in the library.

The definition of known items being items of which the user has bibliographic information might also be too narrow, e.g., there might be items that the user knows and has seen somewhere - not necessarily in some library, but may have forgotten the title or even from whom it was. When switching from the domain of books and journals towards images, this becomes even more likely: Many people have a hard time to remember how a painting or picture was called and frequently works of artists are even untitled or -in particular in the case of personal photo collections- are simply left with the filename given automatically by the digital camera.

If we restrict the view just on search-related tasks for images, we can identify three different main groups with respect to their input and aim:



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 2.1: Small collection of digital photographs to illustrate different tasks and aims.

1. *Known Image Search:* (See 2.2.1)
   The user knows that the image exists and has probably seen it before. The search task will end successfully only if the user has found this particular image.

2. *Classification:* (See 2.2.2)
   The user is concerned about instances that fall into a particular category or all images that share a certain property, where the presence of that property is already sufficient to say whether an image is important to the user or not.

3. *Themed Image Search:* (See 2.2.3)
   The user has the theme in mind that leads to certain preferences. The user is looking for images that match those preferences and search can end at any time the user found enough images that satisfy the needs – independent of how many and which images are left unseen.

These tasks are clearly related to interaction intentions defined in[Xie, 1997], but they are not identical as they leave aside some of the interactions involved and also have been adjusted to be of better use for the domain of images. For instance, *Known Image Search* is closely related to *Looking for known item(s)*, but also to *Locating familiar item(s)*. *Classification* is closely related to *Finding items with common characteristics*, but also *Evaluation* of a given item. And *Themed Image Search* may result in *Exploring* the collection. A dedicated section in this chapter for each of these groups will describe their differences in detail, but a very short example setting will first try to show the differences, in particular between the second two on the list.

Let's assume the user has built an image collection, therefore *Docs* will be a subset of the space of all images $\mathcal{I}$. The pictures in this collection have been taken with a digital camera over a certain period. To keep the setting minimal, we will assume the collection consist of only ten images displayed in Figure 2.1.

Given this collection, the user may now perform the following tasks:

Task 1: Find the image that represents the colored version from which the following grayscale image was generated: (Instance of *Known Image Search*)

Solution: Figure 2.1(d)

Task 2: Find all images that show one or more animals. What kind of animal is shown in each image? (Instance of *Classification*)

Solution: Figure 2.1(b): cat, (c): bird, (e): cows, (h): bee

Task 3: Choose as many images as you like to print a poster or framed image to decorate your home. (Instance of *Themed Image Search*)

Solution: Depends only on your preferences – and you may also choose none if you don't like any of the pictures.

Task 2 already shows that there can be a two-folded nature in *Classification:* On one hand, one may want to retrieve images that belong to a certain class. On the other hand, one may have some image and want to find out to which class it belongs. Abstracting a bit from the particular examples, we can categorize the task inputs and aims into the table displayed in Table 2.1.

Table 2.1: Overview of Task Input and Aim

	User Knowns	Result Relevance	Search Ends
Known Image Search	The image	Only the image	When image is found (or given up)
Retrieval by Class	The class	All images of the class equally	When all images of class returned (even if none)
Image Classification	The image but not its class	Correct assignment of one or more classes	When class(es) is/are determined
Themed Search	His/her preferences	Ranked based on preferences	When result is good enough for the user needs (or gives up)

Analysis of Scenarios from Chapter 1.4

We can look back on the little more complex scenarios in Chapter 1.4 which also resemble more closely real-world problems. The scenarios contain individual image-related

subtasks which are necessary to solve the problems. Those tasks can be categorized using the proposed scheme:

When the user is looking for a romantic sandy beach at sunset as in Scenario 2 in Chapter 1.4.2 – that does express a theme. The user may have certain preferences, but there are millions of images in the world that might satisfy her needs and the task involves to find some image which is good enough.

In contrast, in Scenario 3 in Chapter 1.4.3 Steve is looking for a particular image that he has seen before. His way of navigating can be very different and the kind of input he can give to the system in search may be much more precise. To satisfy his requirements, he needs to find this particular image – or apply a completely different strategy like asking somebody else who was at the wedding as well and may remember the car better, e.g., the driver of the car, but this would be an instance of a *shift of interaction intention* [Xie, 2000] due to failure of the initial approach.

But how does the fountain search of Scenario 1 in Chapter 1.4.1 fit into these patterns? It can be seen as two subtasks that are linked together: The first one takes an images as input and identifies the object in it – the particular fountain. The other subtask shall return more images of this particular object. Or formulated in a different way, the first subtask assigns a class label to the image – the class of images showing the Grabeneck fountain in Basel. The second subtask ideally retrieves images (and only images) that belong to the same class. The quality of the images itself is not as much of a criterion whether the image is helpful or not, because in the end, the user does not exploit an image itself, but the information associated with one of the images.

Notice that a similar situation we find towards the end of Scenario 3 in Chapter 1.4.3 where the car found in the wedding pictures is compared to the cars on Wikipedia to identify the precise model. In contrast, in the beach example in Scenario 2 in Chapter 1.4.2 the quality of the image is important, very important. There might even be the situation where it becomes much more important that the image shows a beautiful romantic scenery than that it actually shows a sandy beach at sunset. It may not necessarily have to be sandy, probably rocks at the beach might also be okay as long as they appear nice to rest on so they won't ruin the atmosphere. And certainly it is not that important whether the scene takes place at sunset or sunrise. So in this case, the relevance of individual result items is very different from the other examples.

2.2.1 Known Image Search

Conceptually, analyzing the properties with respect to relevance and ranking is easiest for *Known Item Search of Images* or shorter: *Known Image Search*.

In this setting, the user knows the image and tries to retrieve this particular image $\mathfrak{I}$ from the collection: $\mathfrak{I} \in \mathit{Docs}$. The use of the image might be embedded in a broader information need, e.g. as described in the example of the particular image of the car that Steve is looking for in 1.4.3, finding the image itself may only be an intermediate step towards the fulfilling the overall goal of getting more information about the car. In this case, the user has only the memory of the image (with a lack of some if its details), and needs to find the image in order to retrieve more details. So whenever the user does

not have the image but knows about its existence, it might be necessary to retrieve the image again.

Also in cases where the user still has a representation or version of the image, it might be necessary to access again the known image. One example might be if the version exists only in a form, that is too small or where the quality is too poor for the intended use – and there exists a different version of higher resolution and quality. Another example might be if the user has only a low quality print-out of the image on paper. *Task 1* in Section 2.2 is similar to such a setting in which the colored version to a graylevel picture has to be returned. Another example might be if the desired information is not contained in the image itself, but *associated / linked* to the image. This could be, for instance, the date of the wedding, the location in form of geographical coordinates, what model of camera has been used, or shooting parameters which might be stored inside the Exif[2] information of the original image file, but are no longer present in any processed or printed version of the image. Another reason to retrieve a particular version of an image is, that since the widespread use of hyperlinks in particular through internet, images may be embedded in arbitrarily complex documents that may contain much more information. As those links usually are lost when a different version of an image or even a bitwise identical copy is retrieved instead of the one embedded image. That makes it even more important to support known image search.

For all these examples, the task can only be finished successfully if the known image is found again. In literature, this has also been noted as *Target Search* in [Cox et al., 2000] and defined as follows:

> Target-Specific Search or, Simply, Target Search: Users are required to find a specific target image in the database; search termination is not possible with any other image, no matter how similar it is to the singular image sought. This type of search is valuable for testing purposes [...] and occurs, for example, when checking if a particular logo has been previously registered, or when searching for a specific historical photograph to accompany a document, or when looking for a specific painting whose artist and title escapes the searcher's memory.

[Smeulders et al., 2000, p. 1351] proposes a slightly different definition, extending it with the target search by example:

> The search may be for a precise copy of the image in mind, as in searching art catalogues [...]. Target search may also be for another image of the same object the user has an image of. This is target search by example. Target search may also be applied when the user has a specific image in mind and the target is interactively specified as similar to a group of given examples[...]. These systems are suited to search for stamps, art, industrial components, and catalogues, in general.

[2]Exchangeable image file format, part of the Design rule for Camera File System (DCF) of the Japan Electronics and Information Technology Industries Association (JEITA) that can be used to store metadata like camera properties and settings either inside an additional file on the camera's storage media or directly inside popular image formats like JPEG or TIFF [JETIA, 2010]

Target Search as described in both settings is not identical to *Known Image Search* as they do not consider linking between image and associated information, therefore not taking into account that even different copies of the same image may not solve the task. The definition of *Target Search* in [Smeulders et al., 2000] does include searching for another image of the same object. This is not included in the definition of *Known Image Search* as those other images might be images that the user has never seen before and therefore may not be able to say whether or not some image returned by the system showing the same object is the desired image. Of course, such cases in which objects inside images instead of entire images are sought, are of great interest as well – but they have different characteristics and thus will be discussed in the context of *Object Detection* in Section 2.2.2 and *Retrieval based on Class* in Section 2.2.2 on pages 34–36.

As a last difference: The definitions and examples given for *Target Search* do not only describe the *interaction intentions* as defined in [Xie, 1997], but also add already particular *information seeking strategy*: For instance the example in [Cox et al., 2000] of searching for the painting, it is noted explicitly that the artist and title has escaped the searcher's memory and therefore may not use this information for searching.[3] In known *Known Image Search* it wouldn't matter whether or not the user remembers the information – the main important aspect is that the users knowns the image and wants to retrieve this particular image. This use case is not uncommon, e.g., if we look at how search engines are used.

Importance of Known Image Searches

In web searches, at least 20% [Tyler and Teevan, 2010] and up to 40% [Teevan et al., 2007] of all queries are issued for re-finding pages that the user has visited before and therefore represent known item searches in the domain of web search. In this context, due to the dynamic nature of web sites, a significant page revisits are performed only because the user expects to find new or updated content in them at the time of revisiting. For an image itself instead of (parts of) a web page, this cannot be expected: once an known image is found again, the user has to expect that it didn't change – otherwise it wouldn't be the same image anymore.

However, there are still good reasons why a user may wish to find a particular image again. E.g., the user may not have bookmarked the page on which it is was shown, but realizes later the she wants to share it with somebody else. It might also be, that the user has to re-find the image even when the page was bookmarked: If the page was altered in the meantime and the image is no longer displayed on that page, the image has to be found via new searches.

If the view is shifted away from web image searches towards searches on personal information spaces like the user's own computer, we can expect the importance of known item search for images to increase significantly: Usually, the user has to expect (or even to hope), that all content of her computer –and therefore also the images stored

[3] It might be, that this information was only added to the example in order to better illustrate the setting. However, it may blur a bit the clear distinction to the next category of searches in [Cox et al., 2000], *Category Search*, which would be the case if the user searches for paintings by a particular artist, for which the name of the artist must be remembered.

on it– did not change unless this was performed upon request by the user. Hence, no novel pictures should appear on this device. The user may copy or download additional images from other devices and watch them on her device, but at the time she searches for images, it is very likely that she wants to find images again rather than new ones.

This assumption is not limited to the private use of image collections. Also in corporate environments, search has a much more intense focus towards re-finding information which is known inside the organization. Therefore, enterprise search is dominated by known item search [Grefenstette, 2010].

Good Results for Known Image Searches

As mentioned already, in *Known Image Search*, the search ends successfully if and only if the user finds the known item. Taking a strict notion of relevance, the set of relevant images contains only the number of known items that are sought. So in the elementary case of searching for a single known image $\mathfrak{I}$, $Rel = \{\mathfrak{I}\}$.

If the user is presented a result list containing the set of results *Res*, the most important question to ask in order to determine if it is possible to successfully finish the task is, whether $\mathfrak{I} \in Res$ or not. Compared to text documents, it is much easier for images to identify on a very short first glance whether an image is the image one is looking for, in particular if the user has seen the image before. Rather than presenting links and text snippets, the system can present thumbnails to the user and if these thumbnail images are not too similar among each other, in many cases just by looking at the thumbnails the user can identify the known item. Presenting 10 - 20 thumbnails on a single page / screen on a common PC screen is easily possible without forcing the user to scroll. Therefore *Precision at k Documents* as defined in Equation 2.5 with $k \in \{10, 20\}$ might already provide some indication of the quality of the retrieval as it is able to express whether the sought image(s) will appear on the first screen. For this reason, there might be little to no need for any system to support the user with the small example collection presented in Figure 2.1 – for solving Task 1, just browsing the collection from top left to bottom right will work fine as already the fourth of the ten images is the desired image and search can end successfully.

So in general, the search can always end successfully already before reaching the end of the list if the image was found. And when scrolling, even much longer lists can be handled without big problems: it just depends on where in the list the image was displayed. Therefore the rank of the image becomes very important - in particular in relation to the overall number of documents $|Docs|$. If a system returns the image among the first 20 images out of a collection of just 50 images, this might be helpful, but not much more helpful than letting the user browse the entire collection. But if a system returns the image among the first 20 images out of a collection of 5,000 images, this will definitely help the user. The rank might also provide an estimate how this would scale with increasing collection sizes: assuming that the collection will grow with similar content as it has already to a size of 10,000 and the image was ranked in first position out of 5,000, one would expect to find the sought image still within the first 20 results.

But if it was ranked 17 out of 5,000, one might have doubt that it would be still within the top 20 results out of 20,000.[4]

From these very simple thoughts on the problem we can already come to the following conclusions:

1. If the number of images in the collection is very small, *browsing* thumbnails of the entire collection is already a strategy that will lead to satisfactory results for the user; meaning that the user doesn't have to invest much time to find the sought image. If any more sophisticated search strategy requires significant time, e.g., to pose a query, it will not be helpful to the user.

2. For huge collections, it might not be necessary that the rank at which the search system returns the image is rank 1. It might be more important, that it is rather reliably able to return a set *Res* that does contain $\mathfrak{I}$ – in particular for cases in which $|Res|$ is small enough that the user will still be willing to browse the set.

3. If comparing two systems that are both able to deliver this property for big collections and do not impose different requirements, of course, the system with the better ranking capability is preferable as it will a) reduce the effort the user has to invest in browsing the results and b) has the better potential to handle also growing collection sizes well.

Strategies for Known Image Search

As mentioned, *browsing* is only an acceptable strategy when the potential number of items to look at is not too big. To improve the situation, the user therefore may try to provide additional information in order to limit the search space, and combines browsing with *filtering*. One easy way to provide such functionality is limiting searches just to a predefined subset of all images of the collection, e.g., all images in a particular folder or that have been grouped into an event. In particular for personal photo collections, the user may very well remember the context in which a particular image was taken. Thus, if the collection organization allows to retrieve images based on events, this can help the user significantly [Mulhem and Lim, 2003].

Even if the image collection itself is not organized well, images and in particular digital photographs are usually attributed with metadata like the time the picture was taken or even the location as geo-coordinates. Such information can be used to perform a *Faceted search* [Yee et al., 2003, Hearst, 2006] in which the user restricts the returned documents to only those satisfying some criterion, e.g., a time range in which the picture was taken. If the images are attributed keywords or tags, these are also good candidates to restrict the results even more. In general, many of the possibilities of *Retrieval by Class* as it will be described in depth in Section 2.2.2 can be used for filtering the results – but the aim of this tasks needs to be kept in mind: the user tries to find particular images that are known. The user may be able to provide information to filter the results, but there might always be some uncertainty.

[4]Of course, if the ranking is performed based on the content, e.g., using similarity search based on visual features, it depends on the content itself, what the final rank would be. So even if the item was ranked in position 1, as soon as 20 new items are added, it might now be ranked just in position 21.

1. *Missing information about available Metadata:* The user may not know which information in addition to the image is present at all and therefore which information can be used for search in the first place.

2. *Quality of Metadata:* The user may not know if metadata is stored accurately, e.g., during the image processing, some information like the date when the image was taken might have been removed or (sometimes worse for retrieval) might have been overwritten with the time of the last modification of the image.[5]

3. *Ability to Memorize:* The user may forget about details and is no longer able to recall the metadata that would help in search.

4. *Incompatibility of Metadata:* There might be metadata available, but the user cannot take advantage of it. This does not necessarily have to be caused by technical problems, e.g., there may not be a controlled vocabulary for natural language information like keywords or tags, therefore the user might simply search for the right concept using the wrong words.

As a consequence, relying *only on filtering* to reduce the number of images to browse down to a manageable size, the first query q_1 may sometimes lead to results with less results $|Res_1| < |Docs|$, but it may no longer contain the sought item: $\mathfrak{I} \notin Res_1$. In order to find out whether the result set contains the item or not, the user has to process the list until the item is found or the end of the list is reached. In the latter situation, the user would have to reformulate the query to q_2 and repeat until some query q_n will contain $\mathfrak{I} \in Res_n$ at rank k. Therefore the total number of image to browse t is not just k but can be computed as:

$$t = k + \sum_{i=1}^{n-1} |Res_i|$$

And in case the user is unlucky, $t > |Docs|$ – so the user might have as well just browsed the entire collection. Even worse, if the user is certain that $\mathfrak{I} \in Docs$, one would expect that on average the item will be found by pure browsing at position $|Docs|/2$. In any case, the user may sometimes not recognize the image on the first look – then the user may have to repeat browsing from the start as there is no indication where it slipped the eye.

It might be helpful if the system can provide a sorted list. But what can the user provide to define valid sorting criteria? If the user is able to provide metadata of which it is known that it can be used and is correct, using this as a filter will always perform better than ranking the list and processing sequentially or even skipping some item until the known metadata criterion is met. If the metadata cannot be used or is not correct, in worst case still the entire collection has to be browsed.

[5] Another common example in case of digital cameras are dates that are not set properly due to long times without usage with empty batteries or not adjusting the time zone when traveling. Other metadata, like the geolocation collected from GPS receivers has inherently varying quality due to changes in the strength and availability of the signal; so they may be accurate down to centimeters some times, but only in the range of about 50 meters at other times – and lose the signal entirely when entering a building or the satellite signal got blocked by other objects.

What can be used for search and has not been yet considered, is the content of the image itself: As the system knowns the images inside the collection, it may compare these images to whatever the user provides.

The user may provide an query image $\mathfrak{Q}$ that is similar to the sought image $\mathfrak{Q} \sim \mathfrak{I}$ and let the system sort all images in the collection $\mathfrak{R} \in \mathit{Docs}$ based on some computed similarity score $sim_{\text{Image}}(\mathfrak{Q}, \mathfrak{R})$. For instance, if the system provides an appropriate measure of similarity sim_{nocolor} that can compute similarity between a gray image and a colored image, *Task 1* in Section 2.2 to find the colored version of a given grayscale image can become very simple.

Such a strategy is called *Query by Example (QbE)*, but has one major limitation in common Known Image Search settings: It may not be possible to provide a good query image $\mathfrak{Q}$ to the system as the user does not have such an image at hand – the user is looking for a known image frequently exactly because she doesn't have the image. One approach could therefore be to let the user start browsing the collection or query different sources until she finds a different image $\mathfrak{I}'$ that somehow resembles the sought image $\mathfrak{I}$ and then start the search on the collection with $\mathfrak{I}'$ as the example. This approach will still only work if $sim(\mathfrak{I}', \mathfrak{I})$ is very good compared to the other images in the collection – otherwise the system may return images which are similar to $\mathfrak{I}'$ but still too dissimilar to $\mathfrak{I}$ to be ranked very high.

Another approach is, to let the user draw a sketch of the sought image $\mathfrak{S}$ to search with. Such a search is called *Query by Sketch (QbS)* and has been proposed already in early works on content-based image retrieval, e.g. [Hirata and Kato, 1992]. So far, two main problems have significantly impacted the successful application of query by sketching to known image search: First, the mouse as most widely available input device limits the user-friendliness and expressiveness for drawing sketches. Second, users usually do not sketch complete images but concentrate on the parts which are most interesting for them. Thus, when comparing user-drawn sketches with images, there will usually be parts of the images to be queried that do not have any corresponding part in the sketch as the user may not remember or may not be able to draw all details of the sought image in the sketch. The user may also place the sketch not exactly at the right position, with proper scale, and/or orientation. Therefore, the corresponding parts may not be at the same coordinates in sketch and image. In order to successfully apply query by sketching to CBIR, both problems need to be solved jointly [Springmann et al., 2010a].

In both cases, *QbE* as well as *QbS*, any appropriate similarity measure will provide the ability that user will realize whenever the results are "too far off", meaning they do not resemble anymore the sought image $\mathfrak{I}$: if $\mathit{Res}[k] \not\sim \mathfrak{I}$ the user can abort the search. However, this will only happen if either a) the user missed $\mathfrak{I}$ in which case she could go back in the list or b) the similarity measure was inappropriate. But there may be a third situation in which this -normally unwanted- situation may become helpful:
Notice that (at least in theory) there is no problem in combining strategies for *filtering* with *ranking*: The user may provide a sketch and still limit the search images which have been annotated with a particular keyword. In contrast to the use of searching with the keyword without some ranking criterion, the user does no longer have to browse all

results on the list in order to find out that $\mathfrak{I} \notin Res$, but can alter the query already when the results get too dissimilar.[6]

2.2.2 Classification

As mentioned already, *classification* w.r.t. image-search tasks can be broken down into two distinct activities:

1. *Image Classification:* is the task when there is an image and it has to be determined to which classes this image belongs. For this task, the image can be commonly be seen in isolation from any other image in or outside the collection. And the outcome of the task is in the easiest case a single class annotation of the image, in more complex cases multiple class annotation which may relate only to distinct areas of the image. Therefore appropriate quality measures for such a task will be not related to retrieval directly, but to accuracy of assigned labels. The common basic measure of how accurate labels are assigned is actually based on how the proportion of images that have been misclassified: the error rate. Errors are distinguished into set of *false positives* F^+, meaning images have been assigned to a class although they don't belong to it and *false negatives* F^-, meaning images have not been assigned to the class although they should have. The corresponding names for set of correct classified images are *true positives* T^+ and *true negatives* T^-. The error rate is then computed as presented in Equation 2.6:

$$\text{Error rate} = \frac{|F^+| + |F^-|}{|F^+| + |F^-| + |T^+| + |T^-|} \tag{2.6}$$

   If one has influence on the classification step, one can trim the parameters to trade the false positive against the false negatives - depending on the needs of the application.

2. *Retrieval by Class:* can be performed when class labels are available for the images inside a collection. The task itself is rather simple: Given a particular class, return all images that belong to it. As all images within the class have to be considered relevant and all have to be returned, this is a task that aims for *recall = 1*: All relevant images should get returned. If we have control over the classification step, this means we would prefer to reduce the false negatives to an extreme at the cost of additional false positives and then take the precision as the quality measure for the retrieval.

A short excursion can show how closely the precision and error rate are connected: To achieve a recall of 1, there have to be no false negatives: $F^- = \{\}$. This is a consequence of the following properties: *Res* contains all items that have been correctly or incorrectly assigned to the class, therefore:

[6] In practice, the possibility to combine the two approaches might be limited by the system in order to be able to provide faster access through indexes, in particular high-dimensional index structures for features extracted from images in the collection to deliver ranked results for QbE/QbS much faster.

$$Res = T^+ \cup F^+ \tag{2.7}$$

Rel contains the true positives T^+ and the missing false negatives F^-. The recall can be computed as presented in Equation 2.8:

$$\text{Recall} = \frac{|Res \cap Rel|}{|Rel|} = \frac{|(T^+ \cup F^+) \cap (T^+ \cup F^-)|}{|T^+ \cup F^-|} = \frac{|T^+|}{|T^+ \cup F^-|} \tag{2.8}$$

The computation of Precision is presented in Equation 2.9:

$$\text{Precision} = \frac{|Res \cap Rel|}{|Res|} = \frac{|T^+|}{|(T^+ \cup F^+)|} \tag{2.9}$$

To compare the precision with error rate, we can first rewrite 1 - error rate:

$$\begin{aligned} 1 - \text{Error rate} &= 1 - \frac{|F^+| + |F^-|}{|F^+| + |F^-| + |T^+| + |T^-|} \\ &= \frac{(|F^+| + |F^-| + |T^+| + |T^-|) - (|F^+| + |F^-|)}{|F^+| + |F^-| + |T^+| + |T^-|} \\ &= \frac{|T^+ \cup T^-|}{|T^+ \cup F^+ \cup F^- \cup T^-|} \end{aligned}$$

So if we can achieve recall of 1 by adjusting the classification parameters to have $F^- = \{\}$, this will lead to Equation 2.10:

$$\text{1 - Error rate} = \frac{|T^+ \cup T^-|}{|T^+ \cup F^+ \cup T^-|} \tag{2.10}$$

If we compare now Equation 2.9 to Equation 2.10 we can see, that they are very similar with the only difference, that Precision ignores the number of true negatives, so the number of images that have not been included in the result correctly are not used in the computation. At this point, we want to end this excursion and come back to the more general analysis of the setting.

Image Classification in isolation

Since *Image Classification* and *Retrieval by Class* seem to be strongly connected to each other, why does it make sense to analyze them as two different steps? A rather pragmatic view is, that they can be performed at different times: While the retrieval only happens whenever the user issues a search to find all images of a particular class, the classification of the elements inside the collection can already be performed at the time the images are stored inside the collection or whenever there is free capacity available. Another reason is, that from the view of identifying building blocks, the building block for retrieving all images of one class can be implemented independently of the concrete way of how the class labels are assigned. And there can be plenty of approaches for assigning labels, e.g.:

- *Human Classification:* Letting human curators categorize the images in the collections.
- *Human Free-Text Annotaions:* Let human annotators add keywords or tags to the images. This is what happens in large scale with shared online collections like flickr or can be performed with the ESP Game described in [von Ahn and Dabbish, 2004].
- *Classification derived from Metadata:* From the metadata of the image, e.g. Exif information like the time a photo was taken or GPS coordinates, images can already be grouped according to year, month, daytime / nighttime or via reverse geocoding[7]. At the border to the next category, one can see classification based on image properties like taking the image height and width to determine the aspect ratio and therefore being able to distinguish between images in landscape and portrait mode, or using the file formats and information like the color palette to distinguish between color, gray level, and black-and-white images; probably even between photographs, paintings, clipart-like images and line art.
- *Automated Classification through Content Analysis:* In contrast to classification based on metadata, this will analyze in addition or solely the content of the images themselves.

The last item on the list is obviously the most challenging and of great interest as this can improve the user experience without spending additional human labor. However, one should not forget and incorporate as well the possibilities that non-automated approaches can provide. And by separating the aspects of the Image Classification and the actual retrieval, one can achieve flexibility in systems to support all kinds of classification.

Automated approaches are of great interest to *Computer Vision* research as this represents basically very well the situation they address: Provide the ability to computers to extract information from images without human assistance. There are two main tasks of interest with regard to this domain:

1. *Object Detection:* Detect whether an instance of a class is present inside the picture. Approaches supporting this task frequently also support localization of the instance inside the picture, e.g., a face detector [Hjelmås and Low, 2001] like the well-known detector described in [Rowley et al., 1998] provides the location and dimension inside an image where it has detected a face. This already can be helpful in some applications, for instance, there are now consumer digital cameras available that have built-in realtime face detection and are therefore able to adjust the autofocus parameters to ensure that faces remain in focus and if flash is needed due to dark environment, add additional short, low-power flashes in order to reduce red eyes. In the retrieval scenario, this can be interesting for example, if one deals with the works of a painter and is only interested in the portraits he did and not in still lifes.

[7] E.g. using publicly available lookup services for countries based on longitude and latitude information like the provided by `http://www.geonames.org`

2. *Object Identification:* As soon as an instance has been detected, frequently there is the desire to identify which instance: If we know there is a face, for retrieval it is commonly very important to be able to say which person's face it is, a task commonly referred to as face recognition. But also for other classes, like the car in the example of the wedding in Chapter 1.4.3 as well as the car in the center of Figure 2.1(i) and behind the cat in Figure 2.1(b) it might be nice to able to identify the make and the model. In Task 2 in Section 2.2 we can also see such a separation: To find all images showing one or more animals, it would be sufficient to perform some animal detection. To satisfy the second part of the task, what kind of animals are shown, this requires additional informations about the instances in the class "Animal". It is hard to draw the line where a class begins and where only individual instances of the class are present. Or rather, in case of approaches based on *Machine Learning* algorithms as they are frequently used for such classification approaches, it commonly depends on the training and may only be limited by the ability of the Machine Learning algorithm to still be able to separate the classes / instances given the features used for classification and the training data.

To assess the quality of different approaches for automated image classification, several benchmarking events have been established like the FERET evaluation of face recognition [Phillips et al., 2000], the PASCAL Visual Object Classes (VOC) challenge [Everingham et al., 2010], the Automatic Medical Image Annotation Task [Deselaers et al., 2008a] and the Visual Concept Detection and Annotation Task [Nowak et al., 2010] of the ImageCLEF workshop series.

Retrieval based on Class

So far, we focused mainly on the *Image Classification*, but not so much on the retrieval itself. As Section 2.2.1 already described what *Known Image Search* is, and the previous paragraphs described what *Object Identification* is, it is probably good to also point out differences between retrieval of a known image and retrieval of images *containing* a known object. In *Known Image Search*, there exist two basic situations:

1. The user is not able to provide the system with the precise image that is being sought: If the user had it, the task would already be over. In such a case, Query by Sketching is therefore a much more appropriate strategy than Query by Example.

2. The user has a representation of the image, e.g., downloaded or printed, but it lacks the linkage to other information that the user is actually interested in. In this case, only the image with the appropriate linkage will be a good result for the user. Therefore the image as a whole is important, but relevance of results is also based on properties which are not part of the image content itself.

In contrast, retrieval of images containing the same object, the user may be able to specify very precisely what the particular relevant object is. The user may be able to provide an example image, so there might be lesser need for methods like Query by Sketching. Rather, the user might appreciate the possibility to highlight the so-called *Region of Interest (ROI)* in the image, as the example image will contain the object, but it may

also contain more than just the relevant object. For instance if somebody is looking for more images of the cat in front of the car in Figure 2.1(b), it may be helpful to be able to provide this information to the system. [Springmann and Schuldt, 2008]

If class membership can be defined or represented through keywords or tags, full-text search might help the user. However, as textual information bears many possibilities of introducing errors, a tool for exploring all available classes might also be helpful, in particular if some domain-specific understanding is represented by them and limited number of choices are available. Relevance feedback might help to refine the search when the initial specification of the class, e.g., by providing a visual example did not map closely enough to the desired results, thus selecting positive and negative example which other images the user expects in the same class can fine-tune the query.

Another aspect is that even if the images have been labeled only with "cat" and "car" in isolation, it is commonly very simple to combine these to form new class definitions like "images which contain cat and car", "images which contain a cat but no car", "images which contain a cat in front of a car", "images which contain a cat sitting on top of a car", "images which contain a cat hiding under a car" and so on that do not have to be materialized in order to be the aim in retrieval: Any new class definition which is based entirely on existing classes might be supported on the fly.[8]

What are other examples of image-related tasks where retrieval based *only* on class membership might be of interest? In literature we find examples, in particular where a user has a visual example and asks for "give me more like this", like in [Cox et al., 2000]:

> Category Search: Users search for images that belong to a prototypical category, e.g., "dogs", "skyscrapers", "kitchens", or "scenes of basketball games;" in some sense, when a user is asked to find an image that is adequately similar to a target image, the user embarks on a category search.

Or as described in [Smeulders et al., 2000, p. 1351]:

> It may be the case that the user has an example and the search is for other elements of the same class. Categories may be derived from labels or emerge from the database [...]. In category search, the user may have available a group of images and the search is for additional images of the same class [...]. A typical application of category search is catalogues of varieties. [Some] systems are designed for classifying trademarks. Systems in this category are usually interactive with a domain specific definition of similarity.

[8] Of course, how efficiently and if effectively any system can support such combined queries depends on the particular implementation of systems. Simple boolean relationships which can be expressed with AND, OR, and NOT can be supported rather easily, e.g., reusing techniques from text retrieval. Full support of spatial requirements might be much harder to provide and probably the task will not be solvable without human assistance, e.g., because it might be too hard to determine automatically from a single 2D image if a cat sits really on top, under or in front of a car. But at least the system may provide some help by filtering out all images which cannot satisfy the criterion, e.g., because they do not contain a cat and a car at the same time. And of course, the quality of the classification of the individual parts is important. However, as [Rabinovich et al., 2007] has shown, the spatial relationship between several objects may actually be used to improve the classification of individual parts.

On web-scale, a common task of retrieval by class might be (near-)duplicate detection, in particular to identify copyright infringements. This can either be done by embedding watermarks [Barni et al., 1998] that make reliable detection easier or use the image content itself.[9]

In practice, there are also applications where the precise opposite is the long-term goal: not searching for items that belong into a particular class shall be achieved, but that *no item for this class* can be found. For instance, a person concerned about his privacy may want that there are no images of him on the web. No matter if the images are good or bad – there just shouldn't be any. Of course, realistically speaking, this will hardly be possible with many surveillance cameras and also webcams installed in public places and and people using digital cameras and cell-phone cameras at almost any public event. But still: if one could at least remove all images in which oneself can be identified easily, this would give back some privacy.[10]

A very similar setting is, that parents may decide to let their kids use their computer, but may be concerned about content that is not appropriate for their age. Images in this context might contain content like violence, offensive gestures or pornographic scenes. [Deselaers et al., 2008b] proposes an approach based on learning such classes of unwanted content from example images and also existing web search engines nowadays commonly provide features to filter search results.[11] Another example of classifying unwanted image content is presented in [Mehta et al., 2008] in the context of E-mail spam detection rather than web searches.

But as there might be little interest in defining searches only by saying what is not wanted, such functionality may just present one out of several criteria to assist the search. This may be incorporated in *faceted search* [Yee et al., 2003, Hearst, 2006], as it has already been mentioned in the context of *Known Image Search* in Section 2.2.1 on page 29, in which case the characteristics of the task, what input the user can provide and what is the aim / which results are satisfactory for the user are dominated by the *Known Image Search* – not the intrinsic properties of the *Classification* task. A similar situation can also be found when the user is searching for images with a particular theme in mind.

2.2.3 Themed Search

So far, there was Known Image Search (Section 2.2.1), where the aim is to find *one* image (or several images, but a known number) which are not new. Then there was Classifi-

[9] There exists a dedicated web search engine TinEye (`http://www.tineye.com`) for what is called "Reverse Image Search" which enables to track the appearance of an image online.

[10] A recent debate related to this at the time of writing was certainly Google Street View, that provides pictures taken automatically by a car passing through streets. As it may take pictures of individuals and cars which can easily get identified via the license plate, Google does automatically blur faces and license plates [Google Inc., 2010]. It also provides facilities to request further blurring of images, but in some countries like Germany, it started a discussion whether there's a need for new laws or at least easier, non-product specific means to let the inhabitants of some area express that they don't want their houses be photographed and made visible in any service like Street View.

[11] For Google Images as well as Microsoft's Bing Image search, this feature is called "SafeSearch" and can be adjusted in the advanced search settings, cf. `http://images.google.com/advanced_image_search` and `http://onlinehelp.microsoft.com/bing/ff808441.aspx`.

cation (Section 2.2.2), where the aim is to find *all* images, that share a common property. The images themselves might be new to the user, but the shared property is not. Aspects aside of that property, e.g., which image of the same class is aesthetically appealing, is not of concern. [Marchionini, 2006] proposed a scheme for search activities in general, not just image search, that consists of three kinds search activities: *Lookup*, *Learn*, and *Investigate*. The first kind, lookup, contains search activities like fact retrieval, known item search, navigation, transaction, verification, and question answering. The user activities that a user wants to perform in Known Image Search belongs without any doubt into this group, but also Classification by itself has no aspects of learning or investigation on the side of the user. If there are such aspects, e.g. if someone with more domain knowledge did the classification and the user retrieves images of one particular class to learn what it is, that made the expert group them together – the learning part is not part of the retrieval.

Learn, in the notion of [Marchionini, 2006] consists of activities like knowledge acquisition, comprehension/interpretation, comparison, aggregation/integration, and socialize; Investigate consists of accretion, analysis, exclusion/negation, synthesis, evaluation, discovery, planning/forecasting, and transformation. The two together, *Learn* and *Investigate*, form the area of *Exploratory Search*.

If we look at examples for image-related search tasks, we may stumble open descriptions like the one in a user study in [Rodden et al., 2001]:

> You have been asked to choose photographs to illustrate a set of "destination guide" articles for a new "independent travel" World Wide Web site. Each article will be an overview of a different location, and is to appear on a separate page. The articles have not yet been written, so all you have are short summaries to indicate the general impression that each will convey. You also have 100 photographs of each location, and your task is to choose 3 of the photos (to be used together) for each article. It is entirely up to you to decide on the criteria you use to make your selections – there are no "right" answers, and you are not bound by the given summaries.

The task clearly requires from the user not just to pick a pre-defined image or a set of images satisfying a common criterion (which would require that there are wrong or right choices), but additionally relate the images and the depicted content to the article summaries, compare and evaluate the available images and finally come up with a decision. Therefore such tasks are exploratory image search tasks. And this leads not necessarily to completely different strategies to employ, but a very different use and importance they have. While *browsing* is just a simple starting point or fallback if anything else fails for known image search, it is an unavoidable part in exploratory image search. If the user is asked to make a choice, it is commonly necessary to see many alternatives and not just pick the first picture – even if the first picture is already good and might end up to be the chosen one. In contrast, in known image search, if the first one is the sought image, the task if over.

The task description of [Rodden et al., 2001] is very much in line with the *illustration task* experienced in real-world use of digital newspaper photography archives [Markkula and Sormunen, 2000], it is therefore not just a synthetic test case.

[Markkula and Sormunen, 2000] also notes that photos with symbolism or photos of themes are often used to illustrate feature articles – as opposed to documentary photos which are predominant for news articles, where there is a certain location and event associated to the article and more time pressure to finish the selection of images faster.

In [Cox et al., 2000], apparently this group of image searches has been recognized and probably due to the necessity to browse significant parts of the collection, has led to naming such tasks *Browsing* as an alternative to *Open-Ended Search*:

> Open-Ended Search/Browsing: Users search through a specialized database with a rather broad, nonspecific goal in mind. In a typical application, a user may start a search for a wallpaper geometric pattern with pastel colors, but the goal may change several times during the search, as the user navigates through the database and is exposed to various options.

In this view, it is simply assumed that the user has no specific goal in mind. But this may not be always the case: maybe the user has a goal in mind – in this case to find a wallpaper that fits a *theme* the user bears in his mind, it may simply not be easy or not be possible to express that theme in a single query, as the task requires to explore the possible choices. [Picard, 1995] mentions the *theme in the user's head* and describes such tasks informally as "I'll know it when I see it". Due to the important role that this theme has for such image-related search tasks, we refer to those tasks as *Themed Search* tasks.

Looking just at the task input the user can provide, the relevant input would be the user's preference. Such preferences can frequently only be matched to images once the user has seen the image. The aim of the task to find one or more satisfactory results.

Partially this can be found in the description of *Search by Association* in [Smeulders et al., 2000, p. 1351]:

> There is a broad class of methods and systems aimed at browsing through a large set of images from unspecified sources. Users of search by association at the start have no specific aim other than find interesting things. Search by association often implies iterative refinement of the search, the similarity or the examples with which the search was started. Systems in this category typically are highly interactive, where the specification may by sketch [...] or by example images. [...]. The result of the search can be manipulated interactively by relevance feedback [...]. To support the quest for relevant results, other sources than images are also employed, for example [...].

Relevance in Themed Searches

The last sentence of the description of *Search by Association* in [Smeulders et al., 2000, p. 1351] leads to an interesting aspect, that for other search tasks was much easier to answer: When is a search result relevant?

For known image search, this is easy to answer: the known image(s) are relevant and nothing else.

For retrieval by class, this is easy to answer as well: all images belonging to the class and nothing else.

In both cases, a binary decision was possible.

For themed search, this is possible if the user finally takes one image and uses it and is satisfied with the result: this image certainly was relevant. But there may have been many other images that have been considered as an alternative – they also were relevant to a certain extend and if the finally-chosen image wouldn't have been in the collection, they might have become even more relevant. This is something new: The relevance of a single image depends not just on itself, but on the (non-)existence of other images in the collection.

The study in [Rodden et al., 2001] about choosing images to illustrate a travel article revealed: "Automatically arranging a set of thumbnail images according to their similarity does indeed seem to be useful to designers, especially when they wish to narrow down their requirement to a particular subset."

[Janiszewski, 1998] also revealed, that the choice of alternatives and their presentation to the user affects the user's behavior w.r.t. one picture which by itself has not been altered.

Such properties lead to a much harder situation in evaluating systems for supporting themed image searches and widely prohibit the use of common evaluation strategies like looking at precision and recall which are based entirely on binary relevance judgements. Nevertheless such themed searches play an important role in the real-world use of image searches. Galleries, Stock Image Photography, and Clipart Collections can always provide only a limited number of images – nevertheless they have shown to satisfy many user requests as they might still provide enough variety to contain at least *something* that matches the customer's preferences good enough.

Aspects to User Preferences

[Jörgensen, 1998] identified a need to consider three kinds of attributes that are very different in nature:

1. *perceptual attributes* like the object or person depicted in an image, the colors used or visual elements like image composition, focal point, perspective, etc.

2. *interpretational attributes* like people-related attributes, for instance, the social status of a depicted person, art historical information like the style and used technique, abstract concepts like depicted atmosphere, content/story like an ongoing activity, or external relation like a reference to another image, etc.

3. *reactive attributes*, in particular the viewers response.

The themes the user bears in mind might follow aspects which are hard to grasp as they relate to emotions towards the image or generated by the image, which frequently may fall in the group of reactive attributes.

The following list gives a small number of examples of terms users might use:

- *"Beautiful"*: Probably ever since human beings started drawing pictures, for instance, caveman paintings, some have been more beautiful than others. At the point where pictures and paintings became a recognized art, people had to start judging which are better, which are even masterpieces. This decision might be

effected among other criterion by the perception of beauty. There are a number of indicators like the rule of thirds for the composition of visual arts. But such indicators *alone* are certainly insufficient to fully define beauty of images or even just the objects depicted in the image, in particular as personal tastes and preferences may vary.

- *"Cute"*: Objects inside an image might be described as cute. It may be possible based on visual properties in some context, for instance, animal babies are commonly attributed as "cute", a puppy has a greater probability as being considered cute than a dog, a kitten has a greater probability than a cat. However, interpretation of an image may reveal exceptions to this "rule of thumb", e.g., if a dog or cat does something "cute".

- *"Funny"* or *"Surprising"*: Both require not only to see the image, but also understand what is happening in the image and –in order to be surprised– know what would have to be expected normally.

For such examples it might never be possible to come to a common agreement between several human evaluators – nevertheless this might be part of the theme the user bears in mind when selecting a particular image. For retrieval purposes, a user is interacting with humans could still ask for recommendations from people which share a similar taste. In order to provide such recommendations it is necessary and may always remain necessary to perform human curation of the content and record ratings for the images. As this curation and collection of ratings is labor intensive, it may be helpful to "crowdsource" such activities to social networks and/or use not only explicit ratings, but also implicit ratings given through the attention an item receives. Such implicit ratings can be captured in logs of websites or via eye tracking in controlled environments – or even read EEG signals to read the brain state while presenting pictures to a human viewer at fairly high rate as performed in [Wang et al., 2009] and [Sajda et al., 2010] as this does not require additional user involvement for annotations except watching the pictures.

In the retrieval phase, such forms of ratings can then be used to present the user exploring the collection at least in a ranked manner. For instance, a website focusing on "funny images" may present the images not only in chronologic order, but also the most funny images first (according to user ratings) and let again the user filter this ranked list to present only images, which contain certain concepts – in other words, combined with faceted search.

Image Manipulation to Generate Good Results

When working with images, very frequently one has to edit and alter the image files. This may be anything from applying just changes to creating new images out of existing ones, e.g., changing the file format as not all applications work with any format, changing the image to appropriate size in terms of pixels used on the screen as well as file size as tradeoff between image quality and time to transfer and load the image, adjusting and correcting colors, cropping the image to adjust the aspect ratio, removing

unwanted elements, merging several pictures into a single one (a process also known as photomontage).

Altering images has not just begun with the use of computers, but has been a long tradition in the history of photography.[12] Therefore, if no single image contains the wanted content in the desired form, it is possible to start with a single or several images and create the desired output. However, this task of manipulating images frequently is very time consuming and requires experience in working with the tools needed to perform the task – even if tools in the digital era like Adobe Photoshop [13] provide assistance.

The theme the user may have in mind might consist of several parts and they might be composed together in a certain way. For this reason we will refer to such particular themes as *Composition*. And probably there is no image available that combines all of these in a nice way. In such a case where no such image exists, probably there will be a picture that provides a good starting point for manual editing or even the possibility to use a tool that already does provide combined search and montage capabilities [Chen et al., 2009] and [Eitz et al., 2009b, Eitz et al., 2011b]. As a subtask, the user might also go back to retrieve a known item to use that within a montage. But the task itself will keep the properties of a themed search: The only specification of what the result should be is inside the user's mind. As it may be part of a creative process, it might change over time and the task will ultimately end successfully at any time the user is pleased with the result – and has to be considered unsuccessful as long as the user is not contended with the pictures that have been found or generated.

2.2.4 Comparison of Task Inputs and Aims

In order to conclude the description of the tasks, it is important to point out some distinctions to clarify the differences.

Composition, Faceted Search and Combination of Classes

Composition as described in Section 2.2.3 can only occur within the context of *Themed Search*.

For the characteristics of *known image search* it is essential that the image itself should be retrieved "as is". Known image search might be used as a strategy / subtask to create a montage to satisfy a theme, but the retrieval of the known image will remain its typical properties and the montage as a whole will be a composition and therefore share the properties of a themed search.

If several classes are combined in a *retrieval by class*, this remains a retrieval by a class – just with a newly defined class consisting of existing ones. The definition of such classes might also consider spatial properties like "an instance of this class is left, right, above, in front of, or behind some instance of that class", but still the criterion whether a task ends successfully or not remains that of retrieval by class.

[12] See `http://www.cs.dartmouth.edu/farid/research/digitaltampering/` for a collection of famous examples since the 19th century.

[13] `http://www.adobe.com/products/photoshop/`

Faceted searches using retrieval by class may be used in both other settings, Known Image Searches as well as Themed Searches. Faceted search is just a strategy to achieve the desired results of the tasks - not a new task of its own and does not alter the task aim. This can be validated by the fact that every task that uses faceted search can also be fulfilled without faceted search through alternative strategies, e.g., the fallback solution to use browsing – which will in many cases be less convenient and very time consuming. But if time needed to successfully finish the task is left aside, it can be used to achieve the aim.

Novelty of Positive Results

Novel items in the result set are images that the user has never seen before. The different task input and aims assess such novel items very differently:

- *Known Image Search:* Novelty is not desired, not even possible.
  The user might see novel results returned by the system, meaning images not (consciously) seen before, but all of these results will not assist in achieving the user's goal. The goal can only be achieved through previously known images. A clear indicator for this can be if the task would be verbally formulated as "Find *the* image in which…" as it states that a) the image exists, b) the number of existing images is known - and it is only one. Of course, one can also search for multiple known images, e.g. "Find the image in which … and find the other image in which …".

- *Retrieval by Class:* Novelty is not relevant.
  The classification phase of novel items must be supported by the system (in other words: systems that only work on the previously seen data, in particular the training data of machine learning approaches, are not appropriate), but the retrieval phase only depends on the class labels the system has assigned on images inside the collection – independent of whether the user has previously seen the image or not. A clear indicator for this can be if the task would be verbally formulated as "Find *all* images which…" as it states that the aim is to find as many of them as long as they do satisfy the criterion.

- *Themed Search:* Novelty is expected, but not always necessary.
  In particular in the context of exploratory searches, the user will expect and appreciate novel items. However, it is not always a requirement that the item is new to the user in order to achieve the user's goal: There might be some item that perfectly fits the theme or is best suited to the requirements, in particular in comparison to all the other items available, so in the end the user might pick an item that she saw before – and probably used before.[14]

[14] An example of this might be the repeated use of cliparts in slide-show presentations, where probably the user would prefer to use some novel item, but the choices are limited and the only ones expressing the theme well enough, are the ones that have been used before. Luckily for the attendants of such presentations, people don't insist on using cliparts on every slide and with the availability of photos in addition to a much richer set of cliparts on the web, presenters are no longer restricted to the same small

Classification of Users

[Datta et al., 2008] added to [Smeulders et al., 2000] a classification of users:

> We broadly characterize a user by clarity of her intent as follows.
>
> - *Browser.* This is a user browsing for pictures with no clear end-goal. A browser's session would consist of a series of unrelated searches. A typical browser would jump across multiple topics during the course of a search session. Her queries would be incoherent and diverse in topic.
> - *Surfer.* A surfer is a user surfing with moderate clarity of an end-goal. A surfer's actions may be somewhat exploratory in the beginning, with the difference that subsequent searches are expected to increase the surfer's clarity of what she wants from the system.
> - *Searcher.* This is a user who is very clear about what she is searching for in the system. A searcher's session would typically be short, with coherent searches leading to an end-result.

This distinction does *not* represent well the findings, that in particular professional users of image archives do need significant amounts of searches for exploration and do use browsing very frequently – which have been published independently in [Jörgensen and Jörgensen, 2005] and [Markkula and Sormunen, 2000]. Therefore part of the classification of user intends as proposed in [Datta et al., 2008] seems to depend on the Task Input and Aim, e.g., Themed Searches will always appear to belong to the category of *Surfer* or even *Browser*, if new images get composed and the individual parts appear unrelated until they are put together into some collage or montage. The problem with this grouping is related to the fact, that it is based on individual queries within a session, of which each follows some information seeking strategy, but the interaction intention itself is not always that clear.

[Ornager, 1995] proposed a different typology of user types by query, derived from the study of requests to newspaper image archives (mediated and recorded through the staff working there):

- *Specific inquirer:* The user asks very "narrow" because (s)he already has a specific photo in mind
- *General inquirer:* The user who asks very "broad" because (s)he wants to make the choice without interference from the archive staff
- *Story teller inquirer:* The user who tells about "the story" and is open to suggestions from the archive staff
- *Story giver inquirer:* The user who hands over "the story" to the staff in the archive wanting them to choose the photo(s), because "they know best"

set of cliparts that was shipped with Microsoft Office and therefore there is now less re-use of common cliparts between different presenters.

- *Fill in space inquirer:* The user who only cares about the size of the photo in order to fill in an empty space on the page

This adds some aspects due to the fact that the search is not performed through an interactive system, but by handing over the requests to staff members. However, it is still very easy to relate this typology to the different types of search tasks, as *Known Item Search* will clearly lead to specific inquirers, *Retrieval by Class* is what a General inquirer would perform and the latter (Story teller / Story giver / Fill in space inquirer) resemble *Themed Searches*.

Nevertheless: Both classifications of users show, that it is not sufficient to focus just on the Task Input and Aim (and therefore the interaction intention), but also on aspects how much "freedom" still exists to pick results for the query: Whether the user has a very detailed understanding of what should be considered a good result or if the mind is not made up yet and broad ranges of results might be beneficial. Strategies in the latter case might therefore be to either rely on the system to provide enough diversity to pick results of the own preference or issuing several queries –not refined, more limiting queries, but queries that add new clues or ask for alternative solutions– in order to add more diversity. Another reason to add more freedom is that users might know what they desire, but don't know how to express that properly. These aspects appear with every kind of Task Input and Aim, and are therefore discussed in the next section.

2.3 Matching Tolerance: Between Exact Match and Invariant Searches

The previous Section 2.2 introduced a view on the tasks as a whole, taking into account the characteristics of the input and aim. In this section we will further elaborate on the input and how to match this input to items in the collection. This will continue the last thought in Section 2.2.4 on the classification of users based on how precisely the user can express what is sought.

[Batley, 1988] introduced a classification for textually expressed visual information needs:

> Four visual information types have been identified. These correspond to the different information needs which may be satisfied by a picture [...and] each of these visual information types has several properties by which it may be identified.
>
> - Specific information needs: can be expressed in keywords; can be expressed in a precise search statement; have no unambiguity; deal with the concrete. An example of a Specific information need could be "Loch Lomond".
> - General/Nameable information needs: can be expressed in keywords; may result in unmanageably high recall (the number of items retrieved); often have to be made more specific. An example of a General/Nameable information need could be "A Steamship on a Loch".
> - General/Abstract information needs: are difficult to express in keywords; may involve abstract concepts rather than concrete objects; can be expressed verbally but not in a precise search statement. An example of a General/Abstract information need could be "Sunlight on Water".
> - General/Subjective information needs: are difficult to express verbally; deal with emotional responses to a stimulus; cannot be expressed in a search statement; are dependent on characteristics of a scene as interpreted by the individual. An example of a General/Abstract information need could be "A Pretty Scene".

This shows some similarity to [Enser, 2008] which describes various levels for text-based indexing:

> A more developed model, which has figured quite prominently in the literature, rests on the formal analysis of Renaissance art images by the art historian Panofsky, who recognized primary subject matter ('pre-iconography') which required no interpretative skill; secondary subject matter ('iconography'), which did call for an interpretation to be placed on the image; and tertiary subject matter, denoted 'iconology', embracing the intrinsic meaning of the image, and demanding of the viewer high-level semantic inferencing. Shatford was instrumental in generalizing Panofsky's analysis, simplifying the first two modes in terms of 'generic' and 'specific', and amplifying these

Table 2.2: Panofsky-Shatford mode/facet matrix

	Iconography (Specifics)	Pre-iconography (Generics)	Iconology (Abstracts)
Who?	individually named person, group, thing **(S1)**	kind of person **(G1)**	mythical or fictitious being **(A1)**
What?	individually named event, action **(S2)**	kind of event, action, condition **(G2)**	emotion or abstraction **(A2)**
Where?	individually named geographical location **(S3)**	kind of place: geographical, architectural **(G3)**	place symbolized **(A3)**
When?	linear time: date or period **(S4)**	cyclical time: season, time of day **(G4)**	emotion, abstraction symbolized by time **(A4)**

> by distinguishing between what a picture is 'of' and what it is 'about'. The notion of 'generic', 'specific' and 'abstract' semantic content has since figured prominently in the literature, the more developed formulations containing multiple levels, comprising both syntactic or pre-conceptual visual content, to which are added semantic layers of interpretive attributes which invoke the viewer's inferential reasoning about the local object and global scenic content of the image.

[Armitage and Enser, 1997] provides a useful illustration in form of Table 2.2. This provided a model to analyze requests and assign them to mode/facet combination, e.g.:

- carnivals: **G2**
- Rio carnivals: **S3 + G2**
- the Rio Carnival, 1986: **S2 + S4**

The most easy way to evaluate needs which can be expressed as keywords and have been expressed as keywords by the user are exact matching of the text. There might be several reasons to enforce exact match:

- Because this is what the user wants. The user does not need or want any other keyword, e.g., no images that have been tagged with a typo like carneval instead of carnival.
- Because this is easy to implement / faster to evaluate.

In this chapter, we will only focus on the first aspect. Implementation aspects and performance considerations will be discussed in Chapter 10.5.

2.3.1 Approximate Matching and Similarity Search

If we consider the last two facets in the Table 2.2, 'Where?' and 'When?', and take into account metadata that may have been gathered automatically, e.g., the geospatial location or the time when the image was taken, we can easily find cases in which retrieval based in these mode/facet combinations need to provide some tolerance. For instance, which geospatial coordinates should be still counted to Rio? Where should the border be drawn if somebody searched for something in Europe (which according to this model is also **S3** and specific as it remains named, with geographic borders in the East and South/East that cut countries like Russia and Turkey into a european and non-european part)? When should a season like winter start (**G4**) - as soon as the (local) definition according to the date is fulfilled or also when the content of the image shows typical winter scenes e.g. snow, even if the snow fall was according to the date already in autumn? Does the problem get smaller if we switch to the specific time, like Winter 2010/2011 (**S4**)?

If we do no longer restrict the user provided input for the task to text and other information that can rather easily be expressed as textual data like time, but also consider images and sketches used as examples, it will get even harder to assist the user because it is yet undefined how closely the input has to be matched in order to be useful to the user. Exact matching may work for keyword-based queries in settings, where a controlled vocabulary assures that the right keywords are used – but is of no use for query by example as the only exact match will be the example that was given by the user.

Instead, for the process of similarity search, usually (visual) features are extracted from the images in the collection and the assumption is, that if the features of two images are similar, then also the images themselves can be considered similar. More formally, if the two images $\mathfrak{Q}, \mathfrak{R} \in \mathcal{I}$ have the corresponding features $f_q, f_r \in \mathcal{F}$ with $\mathcal{F}$ being the space of all features, it is assumed that:

$$\mathfrak{Q} \sim \mathfrak{R} \text{ if } f_q \sim f_r \tag{2.11}$$

The process to compute the similar images from all images in a collection to some query is illustrated in Figure 2.2.

Similarity search is not limited only to visual features. As mentioned before, typos can become an issue in retrieval and using some (dis-)similarity measure for texts like the Levenshtein distance can be used to overcome problems in retrieval. Also a thesaurus can be used and the query expanded using synonyms or a (possibly domain specific) ontology to allow the retrieval of related concepts. All these strategies will move away from exact match.

The opposite extreme to exact match is invariance: The particular attribute will not affect the retrieval process. It is obvious, that complete invariance to all aspects of the input is useless – the search would return the same results no matter if the input is provided or not. Therefore invariance is usually only provided w.r.t. some selected aspects.

Between the extreme positions of exact match and invariance, there exists the approximate match. And for approximate matches, the (dis-)similarity can then be used for ranking.

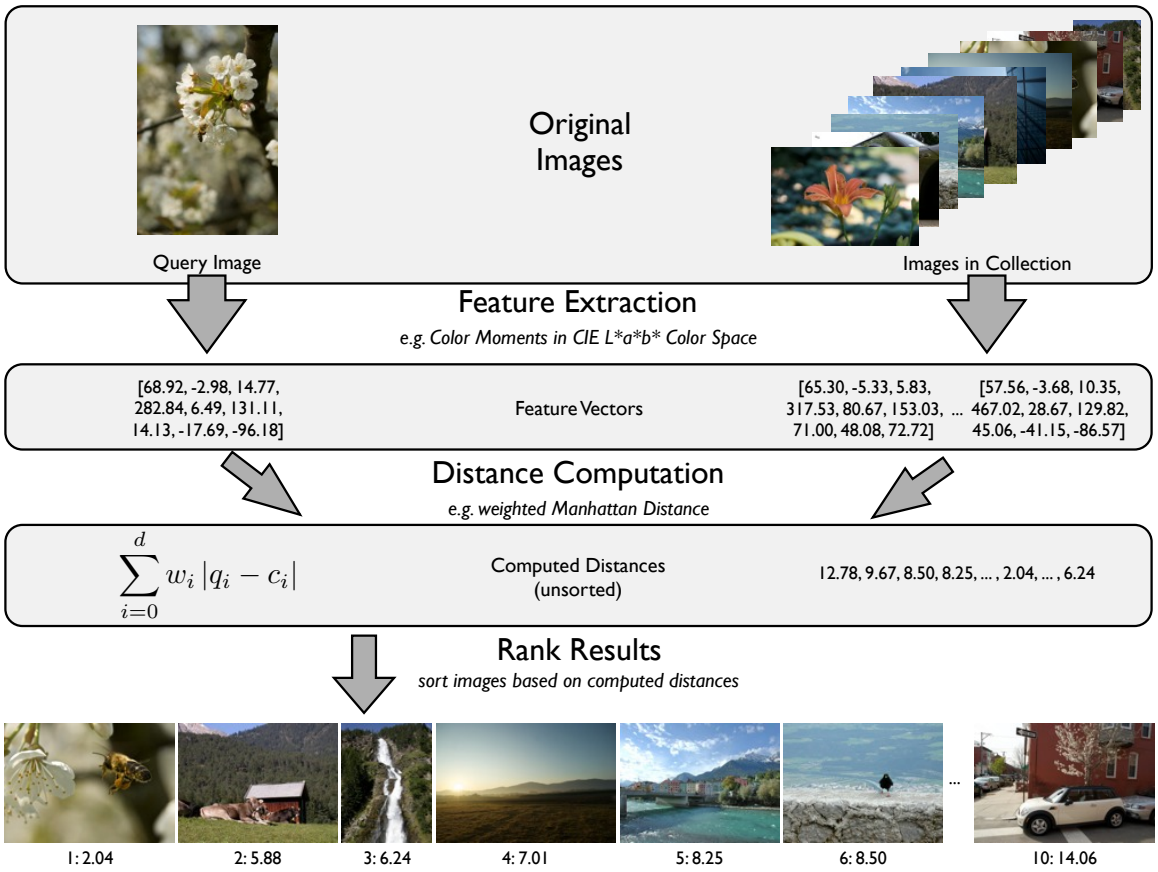


Figure 2.2: Illustration of Steps in Search Process based on Color Moments [Stricker and Orengo, 1995] with a query image very similar to 2.1(h).

2.3.2 Visual Query Input

In particular for visual input as used for Query by Example and Query by Sketching, there are many reasons why the input will always differ from the images in the collection, which imposes quite a number of challenges which lead as a consequence to the necessity of some degree of tolerance / invariance to (at least) small differences:

Input Devices:

For visual input, there is one essential question in the beginning: Does the user have already a good example image in digital format? If not, the image either has to be acquired by a camera / scanner or an example image has to be created by drawing or composing it.

Today, a simple solution to digitize images could be to use a digital still-image camera, for instance just a built-in camera in a modern cellphone which can wirelessly transfer the image to where it will be used. As many computers are now equipped with webcams, it may also be possible to take the digital image directly without the need of any additional device. But the quality of images taken with webcams or cellphones is frequently much worse than what would be possible with dedicated camera equipment

and do webcams and many cellphones also not provide the same capabilities for zoom or lenses that produce very sharp input. This does not mean, that such devices cannot be used at all - they can, and in particular since the whole purpose of the image will most likely be to use it just for the search for better images, the quality of the image itself will not be of major concern. But as a consequence, having very low tolerance in matching would mean, that the system would also return images of similarly low quality. Usually, the user does not desire low quality images as results - therefore the system must be able to ignore deviations which are only caused by the low quality of the input image. This can also be necessary if the user has an existing image, e.g., a rather small thumbnail copied from a website.

For images that exist on paper, in particular when printed with technique that uses a raster, any (re-)digitization like scanning may create Moiré patterns. So also in this case, no matter how good the quality of printed image is, there might be causes of quality degradation - and a need that the picture is not matched too closely within the results.

If no image is available and it is also not easy to shoot a picture or create a rough montage out of existing images, the remaining option is to let the user draw a sketch. In particular in *Known Image Search* tasks, the user will commonly not have the sought image and in many cases also no good example –otherwise the user might not bother and there wouldn't be any need to search in first place– so sketching remains as one of the few options. Currently, the most frequently used input devices for computers that can also be exploited for the creation of sketches are mice or trackpads. They are still not satisfactory for the kind of sketches needed for querying image collections. The task of providing a good sketch as query input is already hard, and the system must be built in a way that is not creating additional difficulty to the expected users. Thus, more user-friendly approaches need to take into account novel input devices that better support even unexperienced users in providing such sketches.

Graphic tablets or digitizers have been used for many years now, in particular in the domain of CAD. Also digital painting depends heavily on pressure sensitive input devices and therefore relies on graphics tablets. However, novel users usually need quite some time to get used to the hand-eye coordination with traditional graphic tablets that only support the digitization of input without display – so the user is not directly able to see the result of her drawing action. There are new tablets available that also provide a LCD display, but they are still far more expensive.

If this functionality is built inside the display of a portable computer, it is now known as a *Tablet PCs* (see Fig. 2.3(a)). Pressure sensitivity of Tablet PCs is usually not as good as on graphic tablets, but for sketching the query input, it still provides a much better user experience than using keyboard and mouse and does not have issues with hand-eye coordination. The vision of a portable computer being able to store user drawings dates certainly back to the late 60's / early 70's, e.g., Alan C. Kay's Dynabook [Kay, 1972] – but it took several decades until the mid 80's digitizers like Pencept appeared [Ward and Blesser, 1986] and in the mid/late 90's that more powerful laptops became available and have been standardized by Microsoft in 2002 [Sharples and Beale, 2003]. The current interest in touchscreen technology for mobile and surface computing has led to an increasing number of devices that can be used to sketch with or without the help of a stylus.

(a) Tablet PC (b) Interactive Paper (c) Digital Pen (www.anoto.com)

Figure 2.3: Novel Input Devices for Sketching Visual Examples

In particular with the availability of less expensive devices supporting such input like some netbooks e.g. ASUS EeePC T91 and tablets like Archos 7/9, Apple iPad and Samsung Galaxy Tab, this technology got much more affordable and available. PDAs have been using stylus input and continued in a current trend to use touch screen interfaces for smart phones. While their screens would be rather small for sketching, the so-called surface computing [Wobbrock et al., 2009] provides now display and digitizer in big sizes.

Another convenient possibility is drawing on paper. Of course, the sketch has to be digitized and using a scanner to do so is commonly inconvenient and interrupting the activity. But since the introduction of *interactive paper and digital pen* interfaces [Guimbretière, 2003] in 2003, such devices allow to capture the input of drawing on paper and transfer them to the PC without the need of scanning the original paper. A special pattern printed on the paper and a small camera inside the digital pen as depicted in Figure 2.3(c) enable the pen to recognize the position on the paper. Via USB or Bluetooth interface, the pen can communicate directly with a PC. Dedicated areas on the paper can also trigger certain actions such that paper becomes an input device that can also issue commands [Norrie et al., 2006]. With these devices, a completely new user experience for sketch-based CBIR becomes possible as they allow the user to draw on regular paper just as they would do if no computer was involved [Springmann et al., 2007b] as depicted in Figure 2.3(b).

Differences in Color:

An important question is whether the image(s) a user searches for contain only black-and-white drawings, full grayscale images, or also color information. Both grayscale and color information are subjective to the drawing and viewing device, and for the very fine details even subject to calibration of the devices. Hence even when using an image taken from one digital camera as input to search for images taken with different cameras or even the same camera with a different white balance, difference will occur. But the problem gets much bigger if different techniques for creating the search input has been used as for the images stored in the collection.

When the user draws a sketch as input, selecting the appropriate colors is a non-trivial task and it may also depend on the use case whether or not this information is needed to successfully fulfill the retrieval task. The survey in [McDonald and Tait, 2003] revealed that users frequently focus very much on color rather than the overall layout of the scene depicted in the image although this does reduce the result quality.

Single Edges vs. Complete Contours:

Another important distinction in sketching is whether the user has to draw complete contours of objects or only simple strokes of edges of objects. In many cases it might be simpler to draw only some of the most prominent edges rather than to sketch the correct contour outline. For instance, if a real world 3D object is drawn, the latter requires that the user mentally projects the object correctly onto 2D.

To be able to automatically fill parts of the images with color, closed outlines are necessary. When color is used with individual strokes, usually small gaps in between two lines remain. If we consider drawing in arts, it is common to use individual strokes that do not necessarily form closed shapes. Several strokes next to each other are used to give an impression about the intensity of color in that area, even though the individual strokes might have a different color. It is also very common to start with a single color (mostly black or grey) to sketch the most prominent edges of the object and then slowly add finer details and colors.

Degree of Detail:

The time the user is willing to invest in creating a query image is limited. When taking pictures with a camera, it may happen that there are unwanted or disturbing elements which would be too cumbersome to remove, e.g., people, vehicles and animals passing by while taking pictures of a building, scenery or landmark. Also if images are taken in more controlled environments e.g. indoor where more control exists when somebody or something steps into the picture, there might be issues with the light casting shadows and reflections, or some other unrelated objects remaining inside the picture. It would require time and effort to remove such elements and the user might frequently not be willing to invest this – especially when as an alternative browsing a little more result thumbnails could also be sufficient to solve the task.

Creating a detailed montage or sketch consumes a lot of time. In *Known Item Search*, the query image will be sketched solely for the purpose of retrieving images and will be discarded after successful termination of the search process. Therefore, the user might rather trade the time for drawing details in query images for more time browsing the result list. Users may also prefer to iteratively add details and skim the result list. They will finally stop adding more details as soon as the result list appears to be 'good enough', that is, likely to contain the desired result image among the number of items the user is willing to browse. Similarly, in *Themed Search*, the user may stop whenever the results are satisfactory – even the sketch by itself is still very rough.

2.3.3 Matching

As soon as the input from the user is acquired, it can be compared to the content of the collection. For this, a number of aspects need to be considered in order to build systems that are helpful to the user and find good results; in particular results that are not misguided by some properties of the examples that turn out to be irrelevant for a user's current task.

Empty, Unknown, Irrelevant, and Unwanted Areas

Related to the degree of detail in the acquisition of the input is an aspect of the matching the input with images from the collection:

For sketches, as already described in [Hirata and Kato, 1992], the sketch of a user is likely to contain large areas left blank. But also in quickly performed image montages, there might remain some blank areas. In traditional keyword-based text retrieval, the user poses a query by choosing the subset of terms that are relevant for the information need. Each document in a collection to be searched, in turn, is expected to contain significantly more words than the query. Therefore, the user already provides a good starting point by defining what he considers relevant. How shall blank areas in similarity search be treated?

Basically, those blank areas in the input can have very different meanings:

1. There is nothing at this spot in the image the user is looking for, just a empty area.

2. The user cannot remember what was at this spot in the searched image, i.e., it is unknown. This is mostly an issue in *Known Image Search*, e.g., for the car at the wedding in Chapter 1.4.3, the user may remember some pictures of the car rather well, but forgot what was next or behind it.

3. The user doesn't care what is at this spot in the result images as long as it is not too disturbing and all the other elements are present. This is most frequently found in *Classification* and *Themed Search*, e.g., when looking for other images of the fountain in Chapter 1.4.1, the user wouldn't mind what is displayed around the fountain, as long as it is the same fountain and reveals further information about it.

4. The user doesn't want what is at this spot in the input image to also appear in the result images as it is disturbing; all the other elements should be present. This is a variant of the "don't care" situation and would commonly be found mostly in *Themed Search*, e.g., pictures of sandy beaches at sunset without any person in it, as in Chapter 1.4.2.

With real images e.g. taken with a camera, there might not be blank areas, but areas that contain clutter or other unwanted elements. For them, the same interpretations exist - it might just be even harder to automatically detect such areas as the user might be the only person being able to say which elements are wanted and which are not. In particular when query by sketching is used, otherwise subtle differences can become important: Considerable parts of the sketch might not be filled as this would require additional time to draw. For the system it is a very valuable information whether some

area that the user left blank should also be empty in the retrieved image. Other areas that the user would prefer to leave empty would be areas that are unknown. The user would usually not draw irrelevant parts except for the reason that it helps to draw, e.g., auxiliary lines. Such drawing aids could be marked by the user as irrelevant. Finally, drawing intentionally unwanted parts could be used to refine the search – if too many results contain such unwanted parts and by explicitly drawing them and marking as unwanted, preference will be given by the system to results that don't contain these parts.

Some of the regions in an image may be more relevant for the user's information need than others (e.g., the foreground or the center of the image might be more relevant than the background or elements towards the boundary of the image). Systems like Blobworld [Carson et al., 2002] allow the user to select the region of interest from the automatically segmented image for formulating a query. However, automatic segmentation does not always give desired results for CBIR, e.g., if the object is partially occluded or consists of several parts. The latter is quite common, for instance in medical CBIR, where the region of interest might be the complete fracture of a bone and thus encompasses two disconnected regions as well as the part in between.

The appropriate treatment of areas in the input will be differ in the individual cases when compared to areas in an image from the collection. Table 2.3 summarizes this. *Similar* in the column of the image means that there is a similar corresponding area to the area in the input; *not similar* likewise indicates that the corresponding area exists, but is not similar. Note that Table 2.3(a) is basically symmetric w.r.t. exchanging input and image. For all other cases, this is not the case – they are asymmetric.

Table 2.3: Meaning of Area Types in Search

(a) empty areas

Input	Image	Meaning
filled	similar	good
filled	not similar	bad
filled	empty	bad
empty	non-empty	bad
empty	empty	good

(b) unknown areas

Input	Image	Meaning
known	similar	good
known	not similar	bad
known	blank	bad
unknown	not blank	neutral
unknown	blank	neutral

(c) don't care areas

Input	Image	Meaning
relevant	similar	good
relevant	not similar	bad
relevant	blank	bad
don't care	not similar	neutral
don't care	similar	neutral
don't care	blank	neutral

(d) unwanted areas

Input	Image	Meaning
relevant	similar	good
relevant	not similar	bad
relevant	blank	bad
unwanted	not similar	good
unwanted	similar	bad
unwanted	blank	neutral

In many cases, the common situation would be that all areas in the input are considered regular, relevant areas, therefore assuming that any blank/empty area in the input

should be empty in the found result images. If this is not the case, additional user input may be required to mark all areas according to whether they are unknown, irrelevant, or even unwanted.

To mark an area, again proper input devices might be helpful – as keyboard an mouse are usually only well suited to either select/deselect predefined area (e.g. automatically segmented areas or inside some regular grid) or do rectangular selections. Performing more detailed selection, e.g., as would be done in image processing with a magnetic lasso tool would commonly take again quite a lot of time when performed with a mouse or trackball/trackpoint with some precision whereas a magic want tool will only achieve acceptable quality for some images. In contrast, rough selections drawn with a digital pen or stylus might deliver a better compromise between precision and additional time required to provide this information. For region selection, even a touch screen without a stylus frequently will provide enough details – which may not always be the case when the same touch screen would be used to sketch the input.

Even when defining such areas, there might be different needs, how closely the areas have to be matched. [Boujemaa and Ferecatu, 2002] describes some scenarios for cultural heritage, which illustrate that both, approximate search as well as very precise search might be needed.

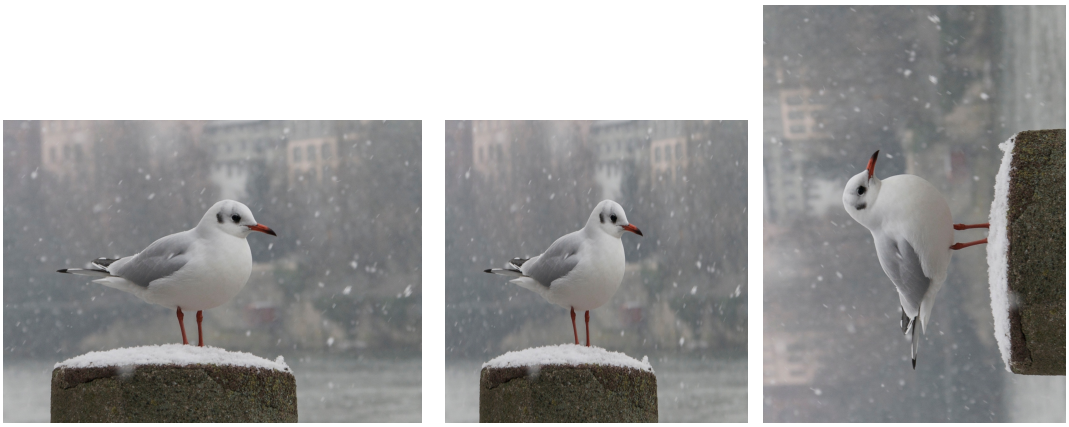
Scale, Rotation, and Translation Invariance

In most cases when the user provides a visual example, the content of the visual example is not identical to the sought image(s). Some exceptions would probably be:

- The extreme case of *Known Image Search* that the user has an image which has lost it's metadata or links by storing on the local filesystem and now simple the user wants to find the original image with all metadata and links intact, or
- Testing the *Classification* by providing one of the examples that were used in training (which is commonly not recommended as it provides the false hope that the classification works well).

In other cases one would only expect that no user would search with an exact copy of what is desired - as the user would have already have what she wants before even starting the search.

Scaling, rotating, translating and shearing form the group *affine transformations* that are commonly used as a basic set of operations in image manipulation. Simple examples are presented in Figure 2.4. Although these distortions might appear quite distracting to the human viewer, it can be quite simple to provide features that are invariant to these changes. In particular the group of so-called *global features* that take the entire image as a source will commonly be not affected a lot. For instance, without going into the precise details, the Color Moments mentioned already in Figure 2.2 have been proposed in [Stricker and Orengo, 1995] as global images that consider the statistical moments of the color distribution over the entire image. The generated features of these images will be almost identical, just changed by the changed number of pixels in 2.4(b) due to the necessary interpolation to change the aspect ratio. Comparing this to what we have analyzed in Section 2.3.3 for empty and unknown regions, this should not be a great



(a) Original Image (b) Scaled Image (c) Rotated Image

Figure 2.4: Examples of simple affine transformations: (b) has been scaled without preserving the aspect ratio to give the image the same height and width (square), (c) has been rotated counterclockwise by 90 degrees.

(a) Original Image (b) 25% zoom/scale out (c) Information Overlay

Figure 2.5: Example of zooming: (b) simulates a different zoom by scaling the cropped selection from a bigger picture, that is, the previous image remains the same but there is added information around it (which is highlighted in (c) with blue horizontal stripes).

surprise: The information in all images 2.4(a)-(c) remained the same – a counterpart can always be found that represents the same pixels.

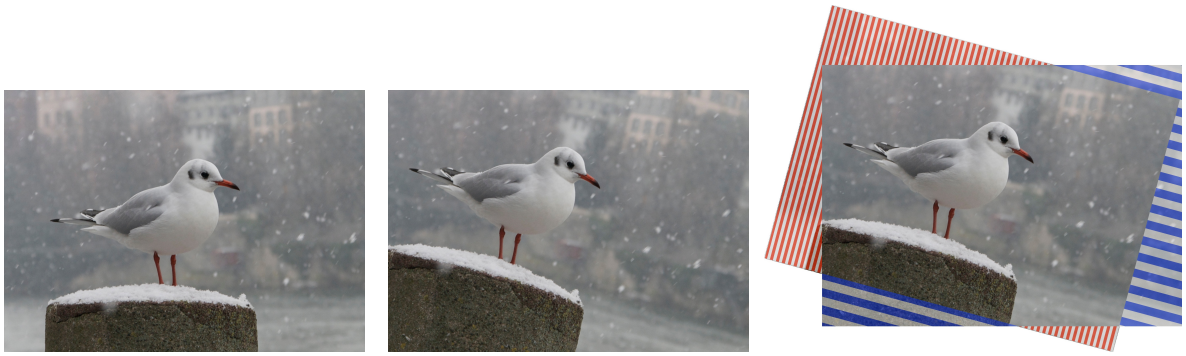
This changes significantly when the image is not maintained "as is" but either altered in different ways or a new picture is taken with not exactly the same parameters: The series of images in Figures 2.5-2.7 show the effect on the information that changes in the pictures.

When the picture would be taken at a different zoom level or from a different distance to the object, it would result in a similar effect as in Figure 2.5. There can be areas with added new information as well as missing information. The latter is the case in Figure 2.6 and Figure 2.7.

In Figure 2.6, the area that is present in both images (and therefore visible without any stripes in 2.6(c)) has been translated to reuse the term from the context of affine transformations. However, to provide ideal support to the user it might be necessary to

(a) Original Image (b) Shifted Area (c) Information Overlay

Figure 2.6: Example of shifted area: (b) simulates a different area when taking the picture by using a different cropped selection of same size from the bigger picture; (c) shows the added information (blue horizontal stripes) and missing information (red vertical stripes).



(a) Original Image (b) Area Rotated by 15 degrees (c) Information Overlay

Figure 2.7: Example of rotated area: (b) simulates different angle when taking the picture by rotating the cropped selection by 15 degrees clockwise from the bigger picture; (c) shows the added information (blue horizontal stripes) and missing information (red vertical stripes).

not only be not affected by translation (or –in other words– *invariant* to the translation) itself, but also to added and missing areas.

The same problem also exists in Figure 2.7 as the rotation also can reveal information that was not present before and loose some information.

If we consider the different task input and aims, we can derive the following requirements w.r.t. scale, rotation, and translation invariance:

- For *Known Item Search*, it is rather common that the user does not have any image closely matching the sought image. The visual example will either be likely to suffer at least form these deformations or the user might provide a sketch. With image examples as well user-drawn sketches, it might happen that the corresponding areas are not placed at exactly the same position – in particular if they contain areas of which the user can't recall the content. Thus, this might lead to (minor) differences in the position of single edges or complete objects. Therefore, the system has

to provide some robustness against parts which have been shifted. However, complete invariance to translation may also be unwanted since the user may wish to search for images where the sketched object is placed exactly in or at least nearby the chosen area and not somewhere else. Again, a user needs to be able to specify during search whether and how misplacement of the sketch should be tolerated by the system.

When the user has to draw a sketch, in particular inexperienced users sometimes have a hard time to estimate proportions when they start drawing. Therefore, even when having a clear picture in mind, they may miss the scale to a certain extent. Hence, a system supporting query by sketching should be able to handle small differences in scale. Very big differences in scale however would usually not occur in *Known Image Search* but rather when searching for known objects inside (arbitrary) images.

For rotations, the situation is similar: small, accidental rotation may be due to the problem of drawing as well as not remembering precisely the orientation of parts of the image. Additionally, images in the collection may not be stored correctly in landscape or portrait orientation, but rotated by 90, 180, or 270 degrees. This problem occurs less for images taken with digital cameras since many of them nowadays have build-in orientation sensors and store their information in the Exif orientation tags. But for images without such information, like old camera pictures, flatbed scans, or images that lost the information during some step in the image processing chain, this may still be an issue. Thus, supporting rotation invariance at least as an option will be an essential requirement for searching using user-drawn sketches.

- For the *Classification of the image content* it is frequently very important that the system provides robustness against all kinds of affine transformations. In many *Classification* tasks where there is no controlled environment to take the input images, this might even be the rule and not an exception. E.g. if we want to classify all birds in pictures, the system must handle much greater deviations as the ones presented in Figures 2.5-2.7 reliably. This is essential to allow for *Retreival by Class*, like searching for known objects in (arbitrary) images.

  But there might also be instances where too much invariance is unwanted in classification tasks: Imagine medical images where the fracture of a bone somewhere in the hand might be considered as a different class than a visually somehow similar fracture in a bone of the leg or the chest. In this case, too much translation invariance would be unwanted.

- Also for montages that are done to express a *Theme*, it might depend on the theme if the placement of individual parts is really important or not.

Invariance to Changes in Perspective, Pose, and Viewpoint

For images of real world objects (i.e., in 3D), the appearance in any 2D projection may change significantly depending on the viewing perspective that was taken when taking

(a) Bird: Pose and Perspective 1

(b) Bird: Pose and Perspective 2

(c) Building: Perspective 1

(d) Building: Perspective 2

Figure 2.8: Examples of changes in pose, perspective, and viewpoint: (a) and (b) are different, but depending on the needs still manageable without too much support for changes in perspective and viewpoint. This won't be possible for the next two pictures: (c) and (d) actually do show the same building, the "Haus der Begegnung" in Innsbruck, Austria. (c) was shot from a road called "Rennweg", 2.8(d) from a different side towards the street "Tschurtschenthalerstraße" and the only thing that really remains similar inside the images are some windows on the upper floor of the building and the sky above it.

or drawing the picture of the scene. When searching for known images, the user may well remember the individual objects depicted in the scene and the image composition, but will probably not be able to remember the perspective in which the objects occurred — or the user is simply not able to sketch them properly (e.g., by choosing a frontal pose even if the object was turned to a certain degree to the side). Figure 2.8 presents some examples.

In general, it can be expected to be a very hard or even unsolvable problem to provide a system that is completely invariant to such changes when all the input it has is one 2D sketch or image provided by the user and another 2D image inside the image collection. There are approaches for estimating 3D poses and even complete 3D models from certain 2D images, e.g., [Romdhani et al., 2006], but these usually rely on prior information about the images and/or objects and thus cannot be directly applied to a generic image retrieval setting. There are also existing approaches to generate 3D models automatically out of many images which might not necessarily have to be taken under controlled settings [Pollefeys and Van Gool, 2002, Pollefeys et al., 2004].

For instance, [Snavely et al., 2008a] reconstructed several public places just from images found on the Internet.

If the differences in pose and perspective between sketch and searched image are rather small, the ability that the system will have anyways to cope with artifacts caused by the input device as well as the drawing skills of the user and invariance to small translations, rotations and changes in scale may also partially cover the needs of changes in perspective and pose. However, one should also bear in mind that due to the differences in perspective and viewpoint, some parts of the image might get occluded and others revealed, thus the information in the image might change. For instance comparing Figure 2.8(a) to Figure 2.8(b), different parts of the buildings in the background are covered by the bird in the foreground.

Invariance to Illumination

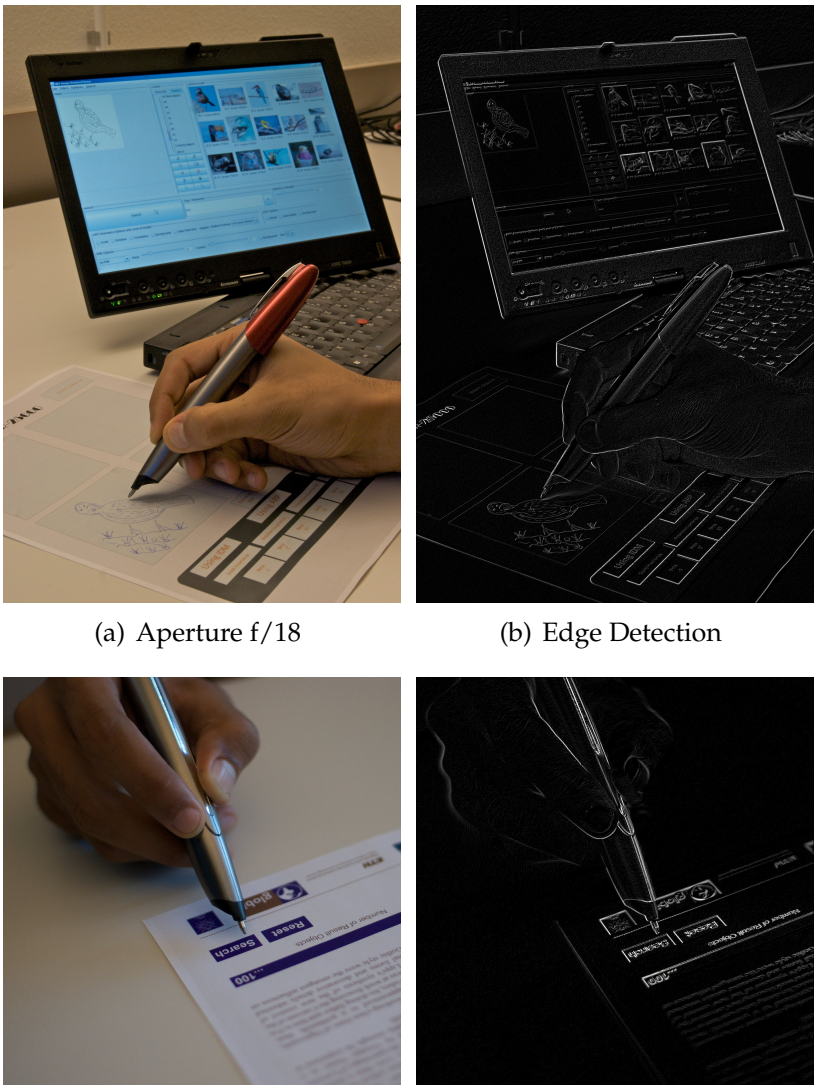
As already mentioned for input devices, there might be differences in color caused by the particular devices and their calibration. But even when the same equipment was used and all possible settings on the device were the same, the environment might have changed. In particular the illumination may change significantly during the day and when considering different times of the year and different weather conditions. Usually the appearance of color is more affected by illumination than shape or texture information like edges, so this is mainly an issue when color is used in search.

Invariance to Focal Length and Depth of Field

As long as no light conditions like strong drop shadows or dimly lit or overexposed areas degrade the quality of the edge detection, approaches not using color and relying only on edge information are inherently robust to issues caused by illumination. However, edge detection may be affected by blurred parts of the image, in particular by the choice of the depth of field, one of the most important stylistic devices in photography. This is caused by the aperture stop selected by the photographer that adjusts the focal length. The area in focus is therefore the parameter that may differ most when attempting to reproduce one image with the same equipment without knowing the precise parameters for the camera or image processing for sharpening/blurring that have been used in production of the original image.

This may also affect the drawing of edge information in sketches for *Known Item Search* and *Themed Searches*: The user may not be able to guess whether or not the border of an object will still be recognized by the system as an edge. Therefore, she might require a certain robustness against the artifacts caused by edge detection. For instance, it would be hard to guess which parts of the blurred text in Figure 2.9(c) would still be detected as some edges in Figure 2.9(d) and which are just different shades of gray to the system.[15]

[15]The Sobel filter is a fairly simple convolution filter, but as it does not have additional parameters to tweak results, it's outcome might be even more predictable than other filters. For instance, the Canny edge detector [Canny, 1986] uses two thresholds, *low* to reduce the number of falsely detected edges caused by noise in the image and *high* to prevent streaking for detecting step edges. Without knowing these additional parameters, one cannot estimate which edges will be detected or suppressed as noise.



(a) Aperture f/18 (b) Edge Detection

(c) Aperture f/1.4 (d) Edge Detection

Figure 2.9: Examples of Depth of Field: While in (a) all parts of the image are rather sharp (e.g. the text on the paper and the images and logo on the Tablet PC in the back), the depth of field is very limited in (c) and only area directly in the focused area of the digital pen over the search field is sharp while text in the front and the thumb of the hand quickly get blurred the further one gets away from the focus. Impact on edge detection is visible in (b) and (d) which both have been processed with a horizontal and vertical Sobel filter on the image in grayscale to detect the edges: Only the areas in focus show strong response to edge detection (white lines).

However, Figure 2.9(b) probably illustrates well why such additional parameters are needed: If not just looking at the white lines that do correspond to edges, but also at the finer gray lines one can see quite a number of supposedly fine edges in the frame of the Tablet PC's display and the cap of the digital pen as well as on the skin of the hand. These are partially caused by the fine structure of the surface of the plastic/skin but also by noise on the camera sensor which has stronger impact with long exposure times or high ISO sensitivity on digital SLR cameras to take a picture with a very small opening for the light –

Sensory Gap

[Smeulders et al., 2000, p. 1352] uses the term *sensory gap* to characterize the uncertainty in what is known about the state of an object shown in a picture.

> The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.

This is particularly the case for considering 3D scenes and 2D pictures as described on 58, where depending on the viewpoint part of the object may be occluded and for other information in the image , it may not be identifiable from the image whether this still belongs to one particular object or another. But the *sensory gap* is not limited only to aspects of viewpoint, but also all the other mentioned aspects in which two different pictures of essentially the same scene can differ – not on the basis of changes in the scene / real world objects, but due to changes in the sensing devices and the processing of the resulting images.

2.3.4 Extreme Cases of Matching: Exact Match and Randomness

The tolerance allowed for approximate matching. The two possible two extremes therefore would be:

- *No Tolerance:* This requires an exact match and as pointed out, there are many reasons why this is rarely desired in image search tasks, in particular when a visual example is provided. However, for certain aspects, in particular facets defined by metadata or highly accurate classification, such little tolerance might be desired.
- *Complete Invariance:* This leads to a situation in which the result of the search does not depend at all on the input provided by the user. Such results are of no use if the user expected the system to respect the provided input – however, they can be seen as the lowest baseline to compare to and are what the user expects when asking for random results.

  Random results are not always completely useless: Due to serendipity it may contain some interesting images. Instead of starting to browse images always in the same order, presenting random results may help to get some sort of overview what is inside the collection. In particular for exploratory approaches to the collection, this initial diversity presented to the user might help.[16]

Most search task will have some aspects that require little tolerance while requiring invariance on other aspects. As already pointed out: Different tasks will differ with respect to which aspects are requiring what level of matching tolerance. Therefore if a system is invariant to some aspect, this by itself is neither good nor bad: It might be a

which is a necessary consequence of the aperture being set to f/18 in Figure 2.9(a) to increase the depth of field without using strong artificial light.

[16] Of course, it only can be considered useful if the user is aware that these results are random. Otherwise the user may think that there is some criterion behind the particular arrangement and try to learn how this collection is structured – which cannot succeed.

requirement for some tasks, e.g., in order to deal with the sensory gap, where in some other tasks this invariance will actually prohibit the user to state clearly what the desired images look like; in other words: Invariance and expressiveness in queries are basically two sides of the same coin. The only thing that appears to be certain is: If a system does not support any invariances and therefore only allows to search for exact matches, it will be of very limited use for image-related search tasks. In contrast, for some search tasks, the users seem to appreciate a fairly high degree of matching tolerance *after* they where able to successfully finish their aim.

Variant of Extreme Case of Matching: Just Not Ending Search

There are situations, where the task could end successfully, but the user doesn't stop using the system. In what one could call *Leisure Tasks*, the user voluntarily continues to explore the collection out of curiosity and enjoyment rather than to achieve an aim.

For instance, if the user was searching for a particular known image from one event and reached it, she may get interested in having a look at other images of that event as memories may pop up. For satisfying the tasks aim, it is not necessary to do so, but the user decides to do it anyway and therefore explores more of the collection because she relaxes the requirements of how closely the results should match the inputs she provided in order to be of interest to her.

Similar situation may occur for retrieval by class: Instead of remaining in the class that was the reason to start using the system, the user might also explore other classes. And in particular in themed searches, where already very broad themes like "a funny picture to cheer up my mood" might exist, the user can go on even if there was a result that the user was very happy with. So the user might just continue as long as she has time and enjoys using the system. To measure how well the user is supported in leisure activities, *User Engagement* as proposed in [O'Brien and Toms, 2008] might be a good candidate [Wilson and Elsweiler, 2010, Elsweiler et al., 2011].

In all of these cases, this neither changed the input the user could provide to the task nor did the original aim change. The user might just end the task at any time as the criterion to successfully end the task was already reached. But what did change is how close results have to match the provided input in order to be of interest. And therefore leisure tasks can be characterized with tolerating very high deviation.

As one of the strategies for exploring the collection with much tolerance is browsing, this may have lead to the term *Browser* in [Datta et al., 2008] as already mentioned as one potential user class in Section 2.2.4. However, modeling this behavior as matching tolerance for a particular task rather than a property of the user herself seems to better represent finding in literature and might give a possibility to express why a certain search session was continued much longer than needed to finish the task.

(a) Image Content provides desired information: Emblem of car make visible

(b) Image Content does not provide the information: Name of fountain etc. not contained

Figure 2.10: Images from Examples in Chapter 1.4.3 (a) and Chapter 1.4.1 (b)

2.4 Result Usage: Content or Representation

As soon as results are presented to the user, depending on what the user is trying to do with the image, another major distinction may occur.

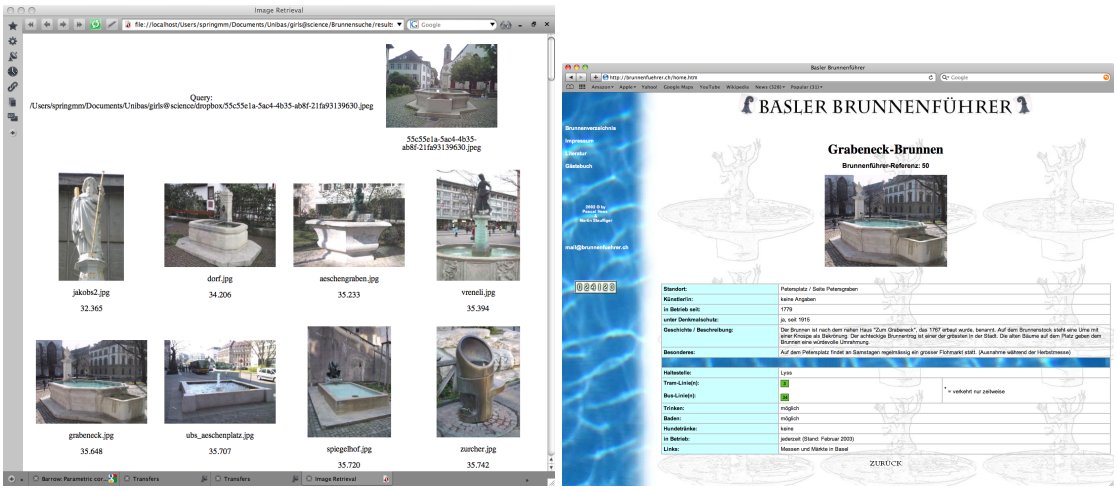
On one hand, there are certainly use cases where the information displayed in the image is used exclusively. Some examples of this have been mentioned in [Fidel, 1997, p. 189]:

> Systems for the retrieval of cartographic material, medical slides, or chemical structures are examples of systems that store images which are commonly retrieved for the information they embody. A user may want to retrieve a map to see how to get from one place to another; a physician may need a slide of a normal foot to help decide if a patient's foot is flat; and a chemist may use a diagram of a chemical structure to examine the molecular structure of the elements involved. It makes no difference to these users who created the image, its history, or how it relates to other images - as long as the image provides them the information they need.

In [Fidel, 1997], this use of the content itself has been named *Data Pole*. Another example of this usage of the information inside the image content was to determine the make of the car at the wedding presented in the example in Chapter 1.4.3 on page 16: Looking closely at the images (once they were found) provided enough information to determine the make based on the emblem (cf. Figure 2.10(a)).

The opposite extreme when probably not so much the information in the image itself, but associated with the particular representation of the image was named the *Object Pole* [Fidel, 1997, p. 189]:

> Stock photo agencies are a clear example of organizations that provide retrieval of images as objects. [...] Such agencies may be asked for concrete or abstract images, for a very specific kind of image of a person or an event, or for any image that represents a specific idea or object. What is common



(a) Filename reveals fountain names (b) Image embedded in page with information

Figure 2.11: Images from Examples in Chapter 1.4.1

> to all these requests is the future use of the retrieved images: They all will be used as objects in the products of the requesters, whether as pictures in a history book, as part of an advertisement about the Internet, or on the cover of the next issue of a magazine.

As both form extremes, there were also cases defined in between [Fidel, 1997, p. 189]:

> Graphic artists, medical instructors, and art historians are examples of users who may retrieve images both as information sources and objects.

The article was written in 1997, so several highly hyperlinked knowledge sources like Wikipedia did not exist or were in their infancy. For this reason, the data pole might have been restricted to images that include the information, ignoring all the information it may link to / serve as an anchor. [Enser, 2008] noticed that there are particular application areas, in which the way the image is embedded and associated to information gains great importance:

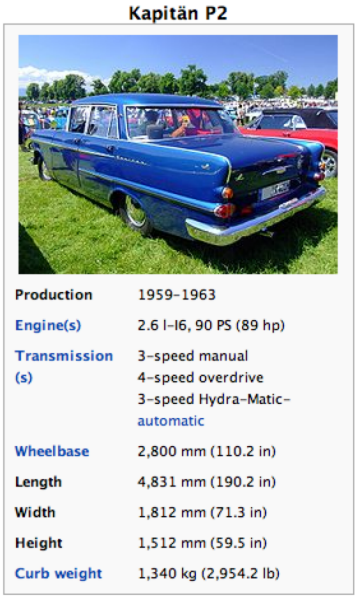
> [There are] classes of images, notably in the medical, architectural and engineering domains, [that] tend to occur as adjunct to parent records, and it is these parent records which are usually the object of retrieval rather than the images themselves.

But also in any other domain, depending on tasks in which the image-related search tasks occurs, we might find examples where the image itself is not the target of the search, but the associated information.

This was the case in retrieving more information about a particular fountain in the example in Chapter 1.4.1: The image itself did not add new helpful information, but when restricting the search to images that are used in a website that provides additional information to any of the images, it becomes the navigational help to get to the information.

(a) Car not embedded

(b) Same model embedded in Wikipedia Infobox



(c) Different model embedded in Wikipedia Infobox

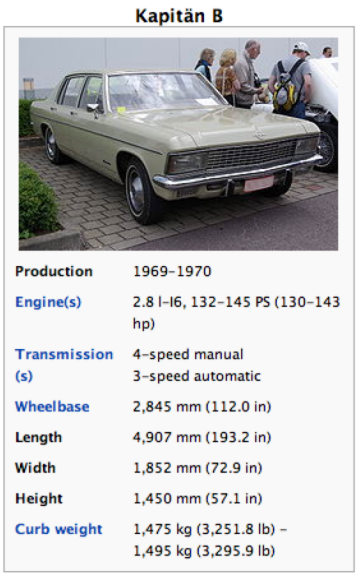


Figure 2.12: Images from Examples in Chapter 1.4.3 (a)

Another example can also be found in when not only determining the make, but also the concrete model of the car at the wedding presented in the example in Chapter 1.4.3 on page 16: the image of the car is compared to a list of images of different models on the page in Wikipedia (cf. Figure 2.12). It is not the image itself that is of great interest - it is just a means to get to the information associated to it.

2.4.1 Content- vs. Representation-Orientated Usage

As an extension to the notion of *Data Pole* and *Object Pole* we want to use the following terms to characterize the image usage:

- *Emphasize on Image Content / Content-oriented*: The user is interested in the information displayed in the image. Any other image with the same content, e.g., showing the same person, physical object, same medical condition has the same value to the user.
- *Emphasize on Particular Representation / Representation-oriented*: The user is interested in how the information in the image is presented, in which form the image is provided, and how the image is embedded or associated. Any other image or even different representation of the same image will not provide the same value to the user, e.g., using even a bit-wise identical image file that has been downloaded from the net and stored locally might be of little value if the association to the website from which it was downloaded is not maintained.

We prefer this notion as the term "object" can be misleading - it may either refer to the objects displayed in the image (content-oriented, as it frequently occurs in Object

Detection-related tasks) or the use of the "image as an object" (reprensentation-oriented, as it was intended in [Fidel, 1997]).

"Representation" on the other hand may either refer to the artistic aspects of how to display the content as well as how to technically represent the image as file or sequence of bytes – both of which would be found towards the representation-oriented side / *Object Pole*, and therefore not misleading.

2.4.2 Interactions with Search Tasks

[Fidel, 1997, p. 191] also derived some searching behavior characteristics based only on the distinction between *Data Pole* and *Object Pole*, e.g., about the relevance criteria and whether browsing can be performed rapidly. This shifts or enlarges the focus from the *usage* of the result to stretch also over the search process itself. Although it is certainly possible to also characterize the search tasks based on whether the search strategies employed by the user focus more on the actual content (data) than the representation (object), they can be very different from the usage of the result itself – as the following examples will show.

- *Known Image Search* for the search itself is always very representation-oriented as the user knows that this particular image exists and searches for exactly that image, not another image showing the same content in a different form. The usage, however, may be focussing just on the content itself. For instance: Imagine there was an event and the user wants to know who was present at that event. If the user remembers that a group picture was taken, the task can be solved by finding this image again and looking at its content: The pictures of the people who attended the event. Just because the user was searching for a particular image does not mean, that there is a need to use any of the visual aspects, metadata, or links from the image – it just means that the user preferred this interaction over an alternative. E.g., in this example, the user could also have a look at each individual image taken at the event to see if there is another person that was not seen on the previous images. This approach would also work if there wasn't a group picture – however, if there is, the approach of analyzing just the group picture will be much more efficient as finding the group picture even through browsing can be performed significantly faster than analyzing each individual picture.

- *Classification* is for the part of image classification in most cases focussing on the content itself. If this would determine already how the image has to be used, there would be no need at all for *Retrieval by Class* as all images within the class contain the same content (w.r.t. the classification criteria) and therefore returning any of the images from the class would have the same value as returning all of them – which is exactly the opposite of what has been described in Section 2.2.2 on page 35. If we recall any of the scenario in Chapter 1.4 where classification is used as a subtask, for instance, the fountain search in Chapter 1.4.1 obviously not any image showing the same fountain was of same value. Instead, the retrieved images were only used to exploit the linkage to websites – which is the most radical usage towards representation-orientation (or Object Pole).

- *Themed Search* for the search is commonly never ignoring the representation. In exploratory searches, it may however be the case that just the content is analyzed and aggregated. For instance, a medical doctor may prepare a talk for which he collects various images showing fractures of the same bone and select from these some examples to show the variability of the fracture that one has to deal with in common clinical practice. The aim in this task would not be to collect as many images of fractures, but to satisfy the theme to give a good-enough overview. The usage of the selected images is therefore very content-oriented as it will be dominated by the questions: Is the image showing the fracture? Is it a different variation of the fracture from the ones found in other selected images?

Of course, these examples have been selected to show where the search tasks with employed strategies differ significantly from the usage of the result images. But as these cases do exist and can better be described in the context of task input and aim, the notion of *Content-oriented* and *Representation-oriented Usage* as defined in Section 2.4.1 shall only refer to how the results are being used. And this is yet another reason why we prefer these newly introduced terms over the existing definition of *Data Pole* and *Object Pole* found in [Fidel, 1997].

2.5 Image Task Model (ITM)

This chapter introduced a number of aspects related to image search tasks. Those aspects were:

1. Task Input and Aim (Section 2.2) in the form of:
   - *Known Image Search:* Image is known to the user and this particular image shall be retrieved.
   - *Classification:* User may have an image that belongs to a particular class and/or wants to retrieve all images belonging to this class.
   - *Themed Search:* The user has a theme in mind that should be matched as good as possible.
2. Matching Tolerance (Section 2.3) with the two extremes:
   - *Exact Match:* No deviation from the given input is allowed.
   - *Complete Invariance to the input:* Result is independent from the input and therefore indistinguishable from random results.
3. Result Usage (Section 2.4) with the two extremes:
   - *(complete) Content-orientation:* importance exclusively on what the result images contains for usage, independent of artistic/aesthetic value or metadata.
   - *(complete) Representation-orientation:* what is shown in the image irrelevant; artistic/aesthetic value, metadata, and linking/embedding are what matters for the use.

These aspects together form the *Image Task Model*. Figure 2.13 shows a graphical representation of the model. Notice that only for *Task Input and Aim* there are markers for each of the three types; for *Matching Tolerance* and *Result Usage*, there are only indications of the extreme values.[17] Furthermore, as described in Section 2.3 the *Matching Tolerance* in itself has many aspects, such as the degree of invariance to changes in

[17] A different model using Venn diagrams as a visual representation has been proposed in [Lux et al., 2010, p. 3916] for classifying user intentions in image search into the four classes "Knowledge Orientation (ET1)", "Mental Image (ET2)", "Navigation (ET3)", "Transaction (ET4)"; allowing a particular task to belong in the overlapping areas ET5–ET8. From these classes, "Knowledge Orientation" corresponds closely to the content-oriented usage of an image (described in [Lux et al., 2010, p. 3917] as a user's need to "extract knowledge from the picture(s) to answer a question") and "Transaction" corresponds to representation-oriented usage (described as user's need prior to search as "the user wants to find a picture for further use"); whereas "Mental Image" can be seen as more closely related to themed searches and known image search as the user's need prior to search is described as "the user knows how the image looks like" with the user's need being "match the retrieved images to mental image". The classification in [Lux et al., 2010] does provide only a very coarse-grained view on matching tolerance by describing for each of the four classes whether the user's need prior to search is based on "Text", "Concepts", "Mental Image", and/or "Non-viusal concepts" and the user's need during search is based on "Image semantics", "Visual analysis", and/or "Text semantics" – where the emphasize on "text" and "text semantics" already refers to particular strategies used in search as there certainly is no "text semantics" inherent in an image itself.

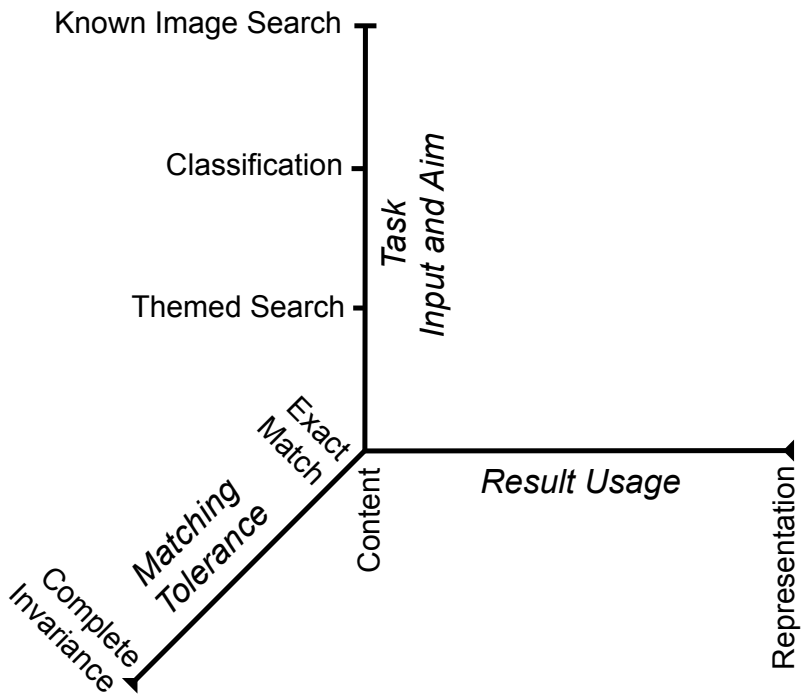


Figure 2.13: Illustration of the Image Task Model to give a graphical representation to the 3 aspects.

scale, rotation, translation, change of perspective, pose, viewpoint, spatial arrangement of individual objects, illumination, color, blurring, or even semantic similarity when different, but conceptually similar objects are tolerated. It is impossible to represent all aspects of matching tolerance on a single axis; the graphical representation therefore is not introduced to encode all these aspects, but to visually compare different tasks.[18] Notice also, that the orientation of the axes in the graphical representation has been chosen for reasons of illustration, not in order to express strict orthogonality in a mathematical sense: As discussed in Section 2.4.2, particular tasks can have a result usages that are more likely than others.

[18] In a mathematical sense, the single axis for *Matching Tolerance* would therefore require an embedding of a much higher dimensional space in which each aspect of matching tolerance would be an dimension of its own. However, as the matching tolerance might even be different for different areas or objects in the image, in particular depending on the distinction between *empty, unknown, irrelevant, and unwanted ares* mentioned in Section 2.3.3, to fully represent all requirements the matching tolerance would even have to de defined for any of these areas independently – which would make it even harder to compare the requirements on matching tolerance for different tasks. Therefore the illustration here should just allow to easily indicate whether the matching tolerance of one task compared to another allows more or less deviations – with the extremes still being exact matching and complete randomness. The task-depending needs in image retrieval for either supporting *selective* or *imprecise* queries has also been explicitly mentioned in [Del Bimbo et al., 1998].
To a lesser extent, the same is also true for *Result Usage*, as also there are various possibilities to use the content or the representation of the image and the main intention of the illustration is not to fully characterize the particular usage, but provide a graphical representation to compare tasks.

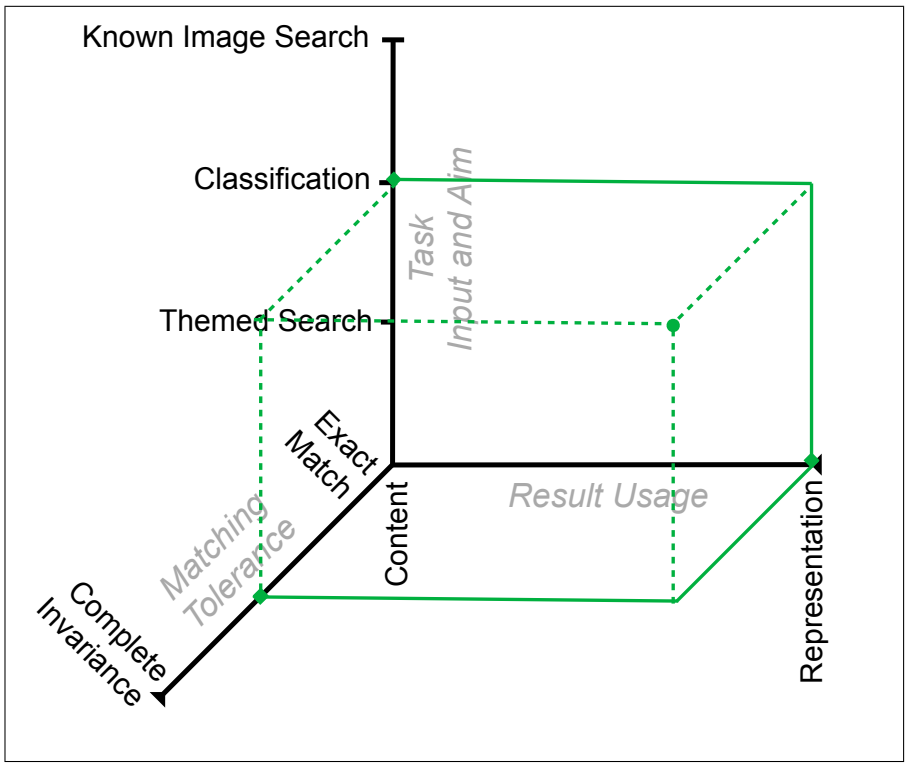


Figure 2.14: Scenario 1 in Chapter 1.4.1: Fountain classification

2.5.1 Task Characterization using the ITM

The Image Task Model can be used to characterize the tasks users may wish to perform. For instance, we can now use the model and its graphical representation to illustrate the Scenarios from Chapter 1.4 and to compare them among each other. This relative comparison will certainly be easier than trying to attach absolute values for the *Matching Tolerance* and *Result Usage*.

The search for the fountain in Chapter 1.4.1 is a classification task, where the user can provide a visual example. The result will only be used to exploit the associated information, not the image itself. Therefore the result usage will be completely representation-oriented. In order to be helpful to the user, the matching has to very accurate with respect to semantics: It is not sufficient that the image shows a fountain in Basel, it has to be the same fountain. However, in the visual appearance, there are quite a lot of degree of freedom: it doesn't matter whether the fountain is placed in the center of the image or anywhere else, it doesn't matter from which side the image was taken or whether it was a landscape or portrait format or whether parts of the fountain are missing because they didn't fit in or where occluded by people or objects – as long one can still identify it as the same fountain. It is hard to define for all those different requirements where to place this w.r.t. the matching tolerance, but probably something like Figure 2.14.

In comparison, the search for romantic beach in sunset in Chapter 1.4.2 gives more tolerance in matching as there are less constraints as long as the image still matches the theme. The result usage does no longer exploit metadata or linkage, but the particular representation for instance w.r.t. image composition and color is still of great importance

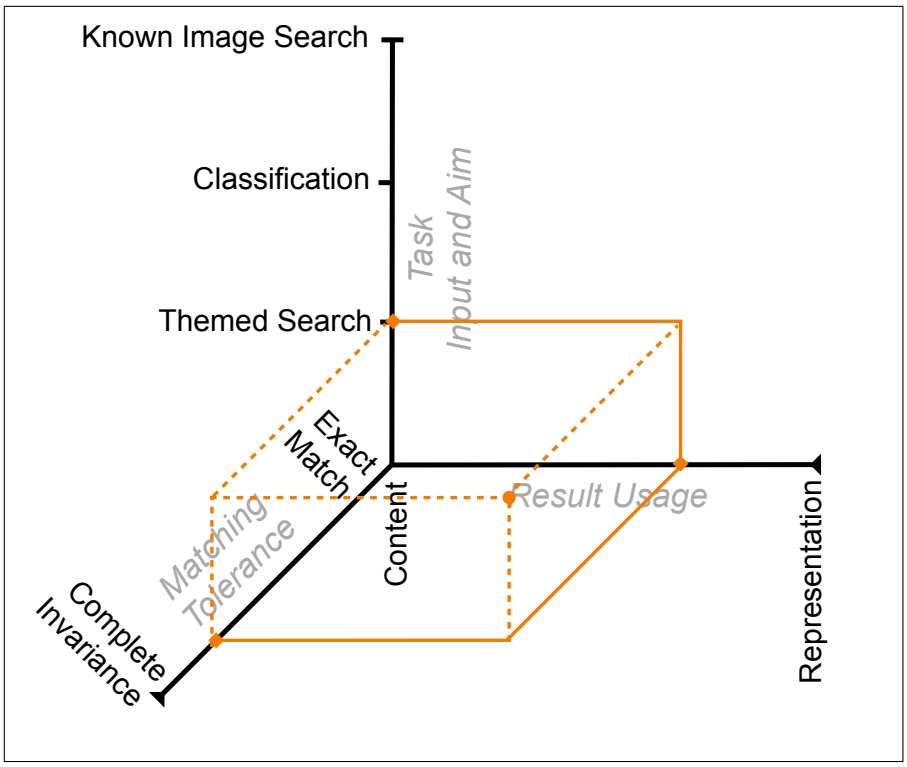


Figure 2.15: Scenario 2 in Chapter 1.4.2: Beach at Sunset theme

as this defines what is appealing about the picture and reduces the amount of manual work required later to use it in the design of the gift voucher. This is illustrated in Figure 2.15.

Finally, the search for known image of the car at the wedding in Chapter 1.4.3 will be used to identify the make of the car from the emblem, so for the usage, the representation is basically irrelevant, what matters is the content of this particular image. In order to ensure that it is really the image Steve had in mind, the image has to match very closely the input Steve provides – here it would be a problem if a different image would be presented as the result and in this image, part of the car is occluded by people or if it was shot from a different side such that the emblem is no longer visible. But as Steve might not recall all details perfectly, his input will still differ significantly from the actual image. Therefore also in this case no exact match is desired as this might result in the answer, that there is no such image in the collection satisfying all the input Steve gave. Figure 2.16 shows an illustration of the characteristics of this task.

What followed in Scenario 3 was, that once the image of the car with the emblem was, that Steve and Chris were looking for a different picture where the car is shown from the back while driving away (a theme) to derive again some hint from the readable content and finally classify the car by comparing it to other images of Opel Kapitän on Wikipedia – which is very similar to the classification of the fountain in Scenario 1, even in regard to using the information associated with the car; but with a little more freedom as the matched car may be a different car of same the model, not necessarily the same car and not even the same color, engine, placed in a different environment, etc.

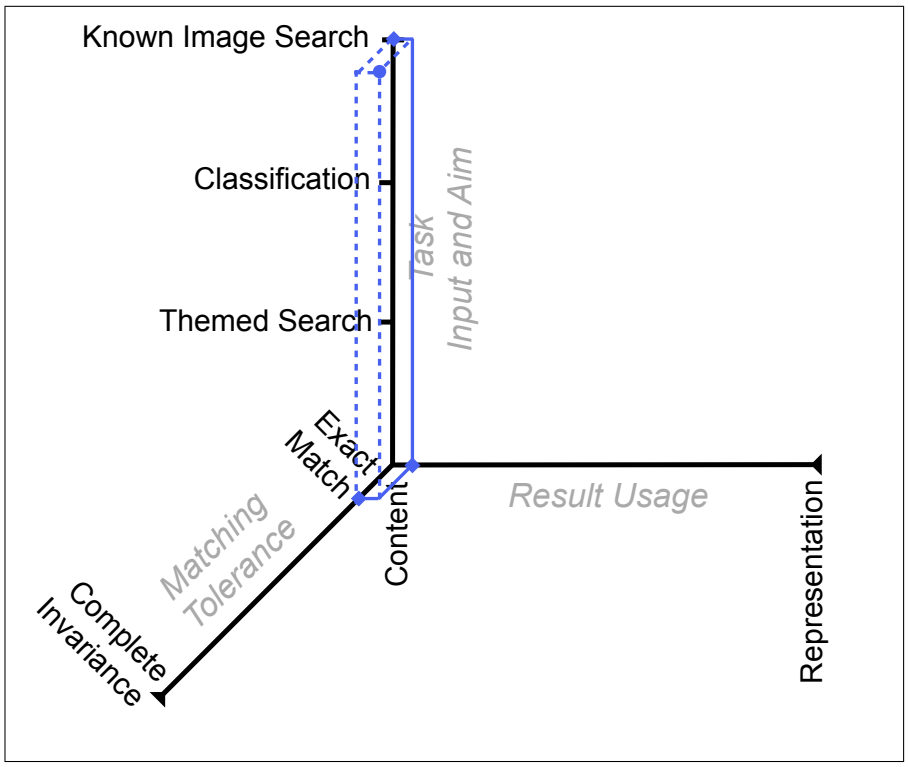


Figure 2.16: Scenario 3 in Chapter 1.4.3: Finding the known image of Car at Wedding

2.5.2 Building systems to cover the aspects described in ITM

At this point is very apparent that different tasks may have very different requirements. Building a single system to support all possible aspects of all possible tasks will therefore be very challenging or even impossible. If a particular search problem is recurring frequently, it might be possible to build a dedicated system to support the user well for instances of this task. And it is also possible to build a single system that somewhat satisfies many of the users' needs – but probably only very few of them in a very satisfactory way.

But there is no need to design every new system for every new task from scratch: now that we have identified aspects of importance to the search, we can identify also building blocks that provide the required functionality. Of course, for any task there still might be many strategies that lead to (hopefully) successful search results – but if the building blocks are not only reusable, but also interchangeable and (re-)combinable, it becomes much easier to ultimately implement individual pieces of software that can match the user requirements very closely.

Therefore the next step will be to identify the functionality that is common to handle some of the aspects and group them into building blocks. This will be the content of the next part of this thesis.

Part II

Functional Building Blocks

3

Overview on Functional Building Blocks

In following chapters we will analyze how the functional requirements of digital libraries to provide content-based access for images can be organized into conceptual *Building Blocks*. As stated in Chapter 1.2 on page 7 we use the term *Building Blocks* to refer to shared concepts that are independent of the concrete language or platform used for the implementation.

It will therefore focus on the conceptual properties of building blocks rather than individual implementations. But as we will see, there are quite a number of interactions visible between the individual building blocks that need to be considered independently of any particular approach and technology of implementing such a system.

Thus, one of the main contributions consists in the description of the provided functionality of the building blocks and the mapping to the *Image Task Model (ITM)*, which has been developed and described in the previous chapter. Another contribution is an extensive review of existing work that can provide the functionality needed for the building blocks. This includes the discussion, for which tasks they can be used and where there are still shortcomings in the state-of-the-art approaches.

To start with the analysis, we should recall the general *Similarity Search Process* as it was quickly introduced in Chapter 2.3.1 on page 48 and depicted in Figure 2.2.

Figure 3.1 shows again this general overview of how a similarity search is performed and overlays the main conceptual building blocks that provide the required functionality.

Content Management which is described in Chapter 4 is responsible for all artifacts stored in the system.

Query Formulation and Execution which is described in Chapter 5 provides all functionality to evaluate the user requests to generate ranked results based on (dis-)similarity. For this, any image has to be transferred into a feature representation.

This is one of the main aspects where *Query Formulation and Execution* has to interact with *Content Management* (illustrated by the overlap of the blue and orange area). But not only *Content Management* is touched by that, it causes also interaction with the third main building block, the *User Interaction* as the features of any query image have to be

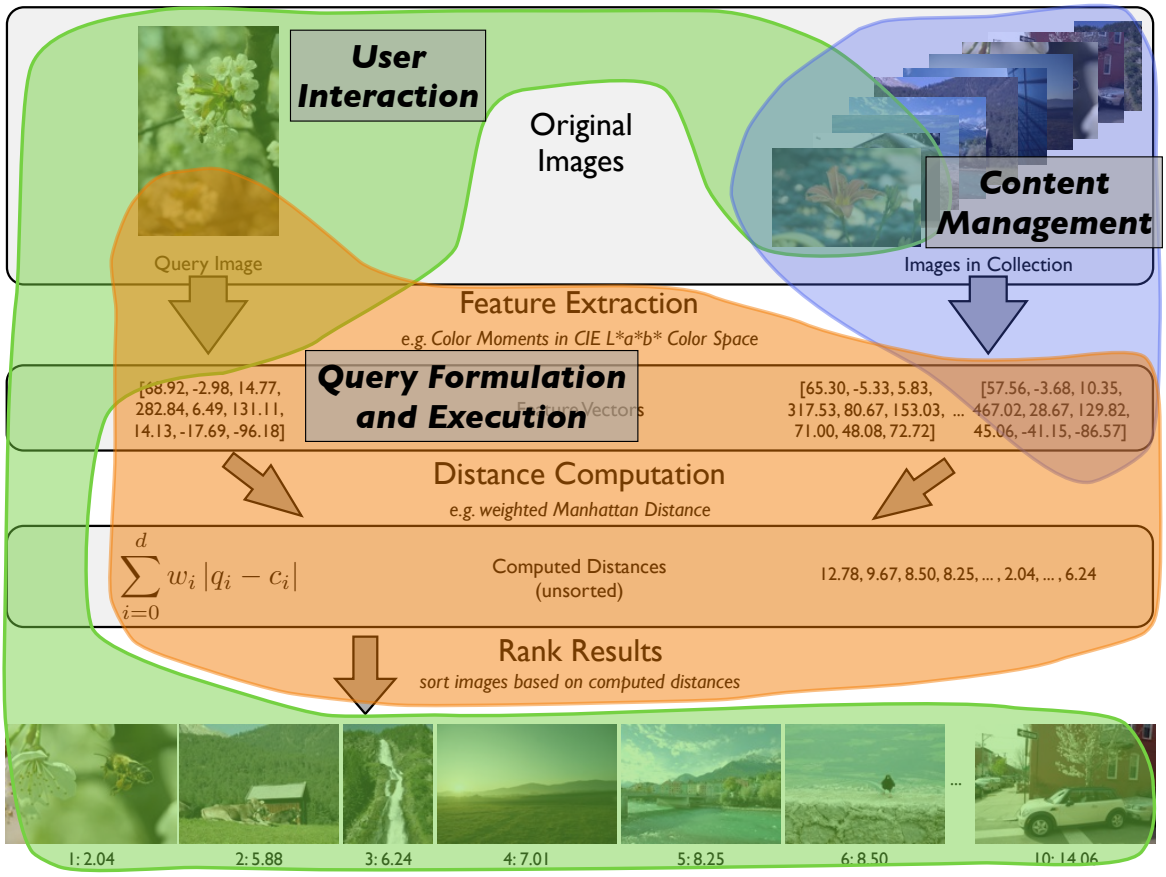


Figure 3.1: Illustration of Building Blocks involved in Search Process: Content Management (blue), Query Formulation and Execution (orange), and User Interaction (green).

extracted and all parameters defined in the user's query have to be considered in the execution (illustrated by the overlap of the orange and green area).

The functionality related to *User Interaction* is described in Chapter 6 and will also take over the ranked result list from the Query Formulation and Execution and present the results to the user.

It may also provide functionality manage the collections and certainly will need to visualize the results; therefore it also interacts with Content Management (as illustrated by the overlap between the green and the blue area).

4

Content Management

Digital libraries are built to store content and provide access to it. In the context of this thesis, the content will be digital images and the metadata associated with them. Considering the result usage, both –the content itself and the metadata– can be of great importance for the tasks the users want to perform. Figure 4.1 visualizes this aspect. In addition to the result usage, also the task input and aim affects which kind of information needs to be stored with the images and how the content should be modeled: Only the information preserved in content management will be available for search and navigation through collections for browsing.

4.1 Structure of Content

With the growing number of images that are stored, it frequently becomes necessary to group items together. On a traditional hierarchical filesystem, such a grouping would usually correspond to folders / directories. In the context of digital libraries, one would rather refer to *collections*. In general, collections should be able to contain arbitrarily many items and –in order to allow hierarchical structuring– should also allow other collections to be contained as a sub-collection. Depending on the digital library, its structure and its users, it can be desirable to be able to place one item in several collections, e.g., in case of images to create particular "exhibitions" with overlapping content. Also this is nothing completely new and can be found in filesystems as links – either hard links or soft links. In order to navigate through the collections, it is important to allow *browsing* – which is also a very important strategy that needs to be supported for known item and themed searches.

The content itself may also have a structure: The so-called *complex documents* consist of several parts and an image may be one part of such a document. In addition, annotations of the content may exist and these might refer to the document, part of a document, or even just a part of a part - e.g., a region inside an image.

A very rudimentary kind of annotations are *tags*. For *tags*, it is common that there is no logical bound to how many tags can be assigned to one single item. A similar concept are *class labels*; with commonly only the logical difference that for class labels, there

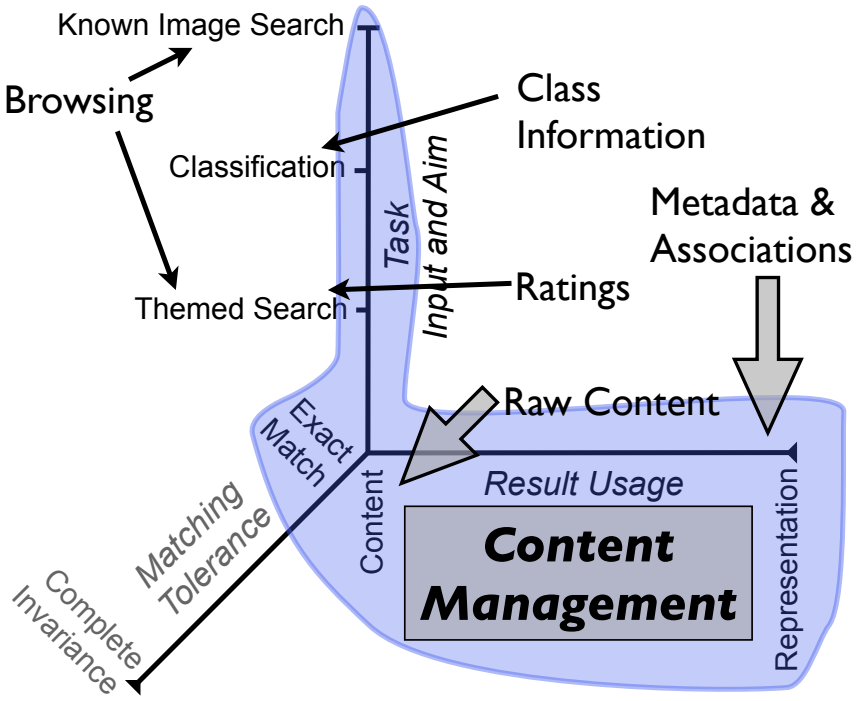


Figure 4.1: Aspects in ITM covered by Content Management: Raw Content delivery and Metadata and Associations affect the entire Result Usage aspects, particular types of Metadata like Class Information and Ratings are relevant for Classification and Themed Search tasks. And browsing frequently becomes an important strategy for Known Image Search and Themed Search.

might exist a general guideline that no item might get assigned more than one class label. No matter whether multi-class assignments are allowed or not: the availability of class annotations is essential to support the retrieval by class. And yet another type of annotations are *ratings* or marking images as *favorites*, which have the particularity that they are commonly handled on a per-user basis: Each user has the chance to express his or her preference. Such ratings can become a very important input for assisting themed searches.

Abstracting from the actual semantics, all this added structure and annotations can be seen as particular *relationships*. Relationships between a collection and the images in it, between collections and sub-collections, between classes and images, between tags and images that have this tag, between users and rated or otherwise annotated content, et cetera. So *relationships* are one of the most important concepts to support in content management.

More for elegance in the model and potential performance improvements than out of conceptual necessity, one may want to introduce another concept to model certain properties or attributes of information objects. A *property* can be seen as a simple key–value pair that belongs to another entity like an information object, therefore cannot exist without it. For instance, one particular property that is of great importance for any document in any digital library is the (unique) document ID. Other properties may be

derived from the file in which the image is stored, e.g., the image file format and image dimension in pixels, the date of its last modification, or the filename. Exposing them directly in the model will make it easier to use them in retrieval.

Modeling such information as information objects of their own that are in a relationship with an image – but it may be more convenient to just see and represent them as properties of the information object, in particular if there is a common set of metadata attribute that any object of a certain type has to provide.

For filenames, it might actually very well be that the case that one would prefer to assign different names to the same item in several collections; just like hard and soft links in the filesystem may give the same file different names. For images, this can easily be the case if the filenames are user-assignable and images may belong to more than one collection: as soon as one creates a new collection that gathers files from several pre-existing collections, the filenames may no longer be unique and –if used as a criterion to order the content on browsing– of little help. Modeling the filename as an attribute of the image rather than per collection membership would not allow to resolve the situation by renaming the image in one collection, but not in any other to which it belongs.

4.1.1 Generic Storage Model

As Section 4.1 has shown, there are a number of logical aspects that content management must be able to express to structure the content. The needed concepts and their semantics for particular digital libraries may differ. However, for providing this functionality, a *generic storage model* can provide the essential elements.

The generic storage model consist of four parts:

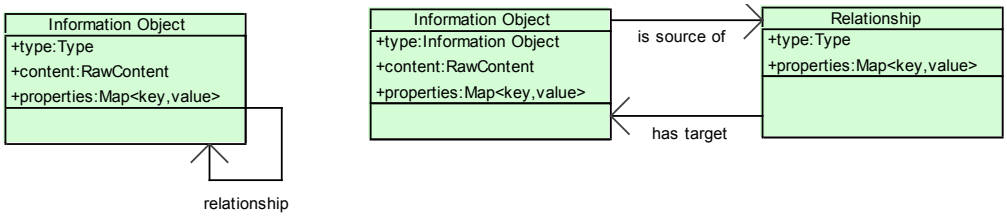
1. *Information Objects:* Any entity that holds information and shall have an ID.

2. *Raw Content:* Content of an information object that is best represented as a binary file or BLOB (binary large object) in a database, in particular images and feature vectors. Not every information object needs to have raw content associated with it.

3. *Relationships:* The relationships between two information objects with distinct roles of the source and the target of the relationship, so there is a particular "reading direction" for each relationship.

4. *Properties:* Any information object and any relationship may have arbitrarily many properties in the form of key–value pairs.[1]

[1] *Attributes* is an almost perfect synonym for *properties*. However, it is quite common in programming languages that support properties that the value of a property might be derived and not necessarily materialized (as it is for *derived attributes*) and changing the value of such a property can be supported by triggering some action that results in the desired effect (which is commonly not the case for *derived attributes* – they are considered read-only).



(a) Simplified Version (b) Generic Storage Model

Figure 4.2: UML Class Diagram of a simplified version of the Generic Storage Model (a) in which relationships are simply expressed by means of UML itself which is easier to be used in subsequent, more complex examples of content models and the direct representation in (b) which expresses relationships as a class of its own and therefore can associate arbitrary properties also to relationships.

So basically, this model defines a directed graph in which the nodes are information objects and the edges are relationships with the added "convenience" that nodes can have properties and raw content.[2]

Figure 4.2(b) presents class diagrams of the Generic Storage Model in UML to illustrate the difference between the simplified and the non-simplified version and ease comparison with other models.

On top of this storage model, domain and application specific models can be built. The two most important entities to deal with in the context of digital libraries certainly are documents and collections.

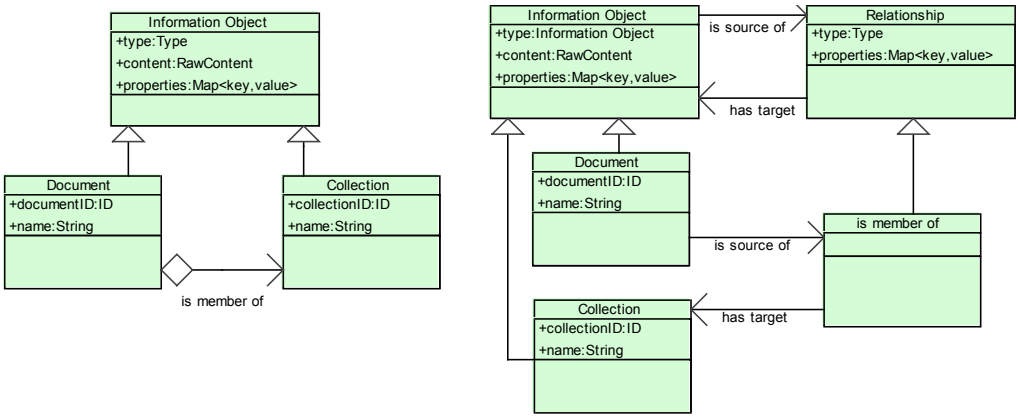
For a rather simple model for documents and collections, the following generic concepts may be enriched with semantics and rules:

- Every document is an information object with the attributes *Document ID* and document name and may have raw content.
- Every document has to participate in the relationship *is member of* collection exactly once.
- Collections are information objects without any raw content. They have the two attributes collection ID and name. They are always the target of the *is member of* relationship, in which they can participate arbitrarily many times.
- Metadata about the document is expressed as properties of the document.

Figure 4.3 shows UML Class Diagrams of this simple content model.
But the generic model allows also for much richer models, e.g.:

- Each and every information object as well as reference has a property "*type*" to make them easily distinguishable. For instance, a document will have the *type* property being "document", the value of the *type* property of a collection will be "collection", and so on.

[2]Commonly it has advantages to not allow relationships to create cycles, therefore the model becomes a directed acyclic graph (DAG).



(a) Simplified Version (b) Simple Content Model

Figure 4.3: UML Class Diagram based on the simplified version of the Generic Storage Model 4.2(a) is shown in (a); the same model using the Generic Storage Model as presented in 4.2(b) is shown in (b).

- Every document is an information object with a property *Document ID* and does not have raw content by itself, but rather has relationships *has manifestation* with manifestations in which it acts as the source.
- Manifestations are information objects with raw content. They have properties that are derived from the raw content like file format, file size, dimension in width $\times$ height for bitmap graphic file formats.
- Manifestations participate at most one time in the relationship *is derived from* some other manifestation, e.g., to express that the manifestation was generated by a format conversion. The relationship will preserve the parameters used for conversion as properties.
- Every manifestation can be the source of the relationship *has replica* which points to replicas.
- A replica is a information object which has to be exactly once the target in a relationship *has replica* and has exactly the same raw content as the source of this relationship.
- Features are information objects that are the source of the relationship *is extracted from* a manifestation.
- A document may also participate in the relationship *has part* as either the source in the role as the parent document or the target in the role of the document part.
- The relationship *has part* has a property *placement* to define where and how the part is located in the overall document. This part may overlap with an existing part of the parent document, e.g., it could describe just a region of an image. As an example, for a huge satellite image, it might be good to model smaller parts

sometimes called tiles, which may have the same resolution as the original image but might be easier to handle due to the smaller dimensions. The manifestations of these tiles can be generated from the original full image, therefore participate in the is derived from relationship with a manifestation of the parent document.

- Relationships may not contain cycles: Any document that is part of another document may not be the parent document of its parent – or any document that is directly or transitively the parent of its parent. The same is true for *is derived from*.

- Every document can participate several times in the relationship *is member of* collection.

- Collections are information objects without any raw content. They have the the properties collection ID and name and can have the collection subtype "materialized" or "virtual". Materialized collections are always the target of the *is member of* relationship, in which they can participate arbitrarily many times. The relationship has two additional properties *at position* to define a (default) order in the collection as well as *with name* and *with description* to define a name and description to be used within the collection. The source of the relationship may either be a document or a collection, in the latter case, cycles are not permitted. Virtual collections do not have explicit members, the membership is defined by membership criterion and evaluated by performing a query. The query itself is an information object which has the query definition as its raw content and is the target of a *is defined by* relationship which has the virtual collection as the source.

- Every collection or document may be the source in a *is annotated by* relationship, in which the target is a document. To limit visibility, these documents that are used as annotations have to participate also in a *is annotated in* relationship with the collection for which it should become visible. As annotations are documents by themselves, they can be textual or images and can have annotations, therefore allow for instance to allow comments on comments to form annotation threads.

- In order to trace modifications, *versions/editions* of documents can be added by establishing a relationship *is successor of* the previous version. The same kind of relationship may also exist for collections to trace their modifications, e.g., to represent annual updates.

Figure 4.4 shows a slightly modified version of this content model by using the simplified Generic Storage Model from Figure 4.2(a), such that regular UML constructs can be used to express relationships as associations, aggregations, and compositions.[3]

[3]The corresponding UML Class Diagram based on the non-simplified Generic Storage Model 4.2(b) is omitted here as it gets much less readable due to necessity to also introduce every relationship as a new class extending the Relationship class and each of them has two directed associations: one with the class or interface that represents source, one with the class or interface that represents the target.

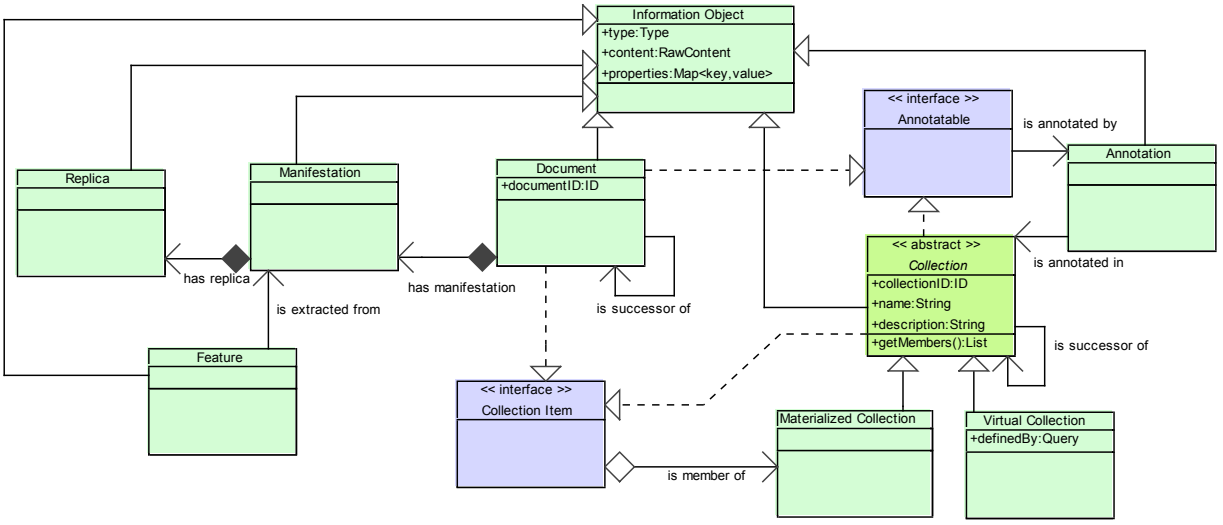


Figure 4.4: UML Class Diagram of rich Content Model which is based on the simplified Generic Storage Model from Figure 4.2(a). Notice that every class in this model is a specialization of the Information Object.

4.1.2 Extensions to Support Image-Related Search Tasks

Even this fairly complex model in Figure 4.4 is not yet sufficient to support all desired aspects for handling user tasks. For this, the following extensions can be added:

- To model also users as creators of the annotations and also to trace who inserted or altered images to the system or which groups of users shall have which access to what collection, *users accounts* and *groups* can be represented as information objects; *annotated by*, *inserted by* relationships with a timestamp as property and *can access* a relationship in which properties can define the precise permissions granted.
- With the particular tasks in mind, the model might benefit from the following extensions:
  - To support *Known Image Search* in home photo collections, it is generally helpful to preserve the original order in which images where taken to allow to find images based on the temporal context (cf. [Mulhem and Lim, 2003]). To enable such a feature, representing the pictures as an "photo stream" and/or using the "photo roll" metaphor for bulk imports can be used, which may later be split up in "events" / "project" / "folders" / "sets" / "albums" / "galleries". From how they act and how they can be implemented, these concepts are much like *Collections*. But to ease retrieval, "events" have to preserve chronologic order to sort images while "albums" should allow the user to arrange the images purely based on the user's preference – but an album should never be the only "collection" the picture is a member of. An "image set" could be selection of images with a common topic rather than a common origin with the intention to be shared with or presented to other people.

"Galleries" and "folders" give additional possibilities to structure the images, where "folders" are dedicated to personal preferences for the use by oneself, "galleries" again more targeted towards how to structure and present images and image sets to other people and aggregate images from and image sets from several people.[4]

- To support *Retrieval by Class*, the class to which an image (or image region) belongs must be assignable. As there might be several independent classifications and a hierarchy of classes (e.g., a complete ontology), each class can be represented by an information object with a detailed textual description added as a a property. Images and image regions participate through an *is instance of* relationship with the class and the hierarchy or other relations between individual classes can be expressed through appropriate relationships.
- To support *Themed Search*, user-assignable tags and/or keywords might be very helpful. They are similar as for classes, but as they frequently have not a definition as clear as for classes, hence they might not have a description property. In addition, to introduce subjective measures, an image might participate in an *is rated by* relationship with a user which has as an property "rating". Out of all ratings for one image, the system may derive the average rating for the image.

- With regard to the representation of images that are imported into collections, the model might benefit from the following extensions:
  - To better support *Representation-oriented usage* of retrieved, crawled content, preserve at least the *URL* from which the content was downloaded. Through this, the user can be pointed to the original source and the associated information presented in that location accessed.
  - In order to support also rich queries, convert as much of the context into the Generic Storage Model introducing new types of information objects when needed.

To make this model even more versatile, the *type* of an information object or relationship may not be a simple property, but expressed as a relationship *is of type* itself. For this, it would be necessary to allow relationships to be the source of a relationship – rather than allowing only information objects to be the source or target of a relationship. The flexibility that this adds is the definition of type hierarchies: For instance, the

[4]Real world examples in practical use: The image sharing website flickr (`http://www.flickr.com`) uses the concepts "photostream" and "sets" to let users organize their pictures (`http://www.flickr.com/photos/organize/`) and "galleries" (`http://www.flickr.com/galleries/`) to aggregate / order / present pictures of several photographers. The social network facebook (`http://www.facebook.com`) has a flat structure of "albums" in which the users can organize their photos. The desktop application Google Picasa has a photo stream, "folders" and "albums" (http://picasa.google.com). Apple iPhoto has a "photo stream", "events", "albums" and "folders" to organize "albums" (`http://www.apple.com/ilife/iphoto/`). Apple Aperture has "libraries" as a top level with "photo stream", "projects", "albums", "folder" (`http://www.apple.com/aperture/`). Adobe Photoshop Lightroom has "catalog" as a top level with "photo stream", "folders" and "collections" (`http://www.adobe.com/products/photoshoplightroom/`).

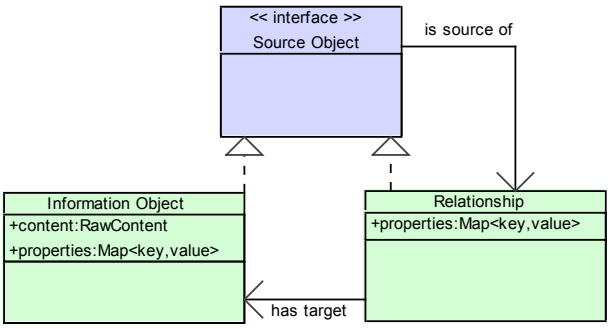


Figure 4.5: UML Class Diagram of an extended Generic Storage Model that allows also relationships to be the source of a relationship. By modeling the type information of Information Objects and Relationships in this model as information Objects themselves, this allows for content models with a type hierarchy.

type "image" could be a specialization of the type "document", expressed by the corresponding information object image to be the source of the relationship *is specialization of* the information object "document" and "text document" another specialization. And for relationships, there could be the generic type *is derived from* which could have the two specialization of a feature *is extracted from* and *is metadata materialized from* for the metadata as described in the following small section on interoperability.

4.1.3 Storage of Metadata

As interoperability with other digital libraries can be of great importance, not only the availability of the raw content in a well-understood format is important, but also the metadata. There can be a need to support a multitude of particular formats for instance:

- Common metadata formats for images and/or media in general like Exif [JETIA, 2010] , IPTC Information Interchange Model [Tesic, 2005], DIG35 [Digital Imaging Group, 2001], Extensible Metadata Platform (XMP) [Troncy, 2008], MPEG-7 [Chang et al., 2001], etc.

- Metadata formats known from library catalogs that are commonly used for digital libraries like Dublin Core (DC) [Dublin Core Metadata Initiative, 2011], Machine-Readable Cataloging (MARC) [Furrie, 2009], Metadata Object Description Schema (MODS) [Guenther, 2003], etc.

- More domain specific formats like Digital Imaging and Communications in Medicine (DICOM) [Lee and Hu, 2003], Digital Image Map (Dimap) [Bally et al., 1999], MPEG-21 [Burnett et al., 2003, Bekaert et al., 2003] and Sharable Content Object Reference Model (SCORM) [Kazi, 2004] etc. of which some of them do not only describe metadata, but also provide a container format for the content.

The generic storage model itself is flexible enough to represent the metadata entries as relationships and properties[5], however it might be desirable to also store the metadata as a file, e.g., the binary or XML representation as raw content such that it doesn't have to be regenerated whenever this metadata representation is requested, e.g., to respond to requests in OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) [Van de Sompel et al., 2004].

4.2 Storing Raw Content Inside or Outside the DL

In general, images in digital formats are commonly stored in binary formats with vector formats being the most prominent exception. The binary content can easily be stored in files in the filesystem or as BLOBs in a database or just link to an appearance on a network resource by a URL (Uniform Resource Locator). In the most simple case, Content Management might limit handling of the raw content to providing the link to the file on disk or on the net.

This simplistic approach however becomes problematic if the content on the filesystem or at the specified location on the net can be altered. This would allow inconsistencies. Comparing this to traditional libraries, it would mean that books can be taken out of the library, replaced by different books or new books added without the library information system being aware of these changes – which could be seen as a disaster.

The organizational approach to resolve this problem is: Any access to the content has to happen under supervision and with the acknowledgement of the responsible authority. For the traditional library, this usually requires the users to obey certain rules, in particular for checking out books. For digital libraries, there are two technical solutions:

1. *Store Content inside/under exclusive control of Content Management:* Only Content Management will be allowed to add, remove, alter files, e.g., by the means of file permissions. Even further: If particular access rights should be enforced, Content Management will be in charge to grant access to the files.

2. *Store Content outside Content Management, but monitor changes:* Files are stored in places, where they can be accessed directly – but Content Management will either have hooks to get notified of changes or will have to periodically check for changes. In such a setup, it will be almost impossible to enforce particular access rights defined in Content Management.

Users will expect that the more the content is curated, that is, carefully selected by humans rather than automatically crawled, the less likely it will be that some content that is returned as a search result is actually not available – as it can be the case not only

[5]Many of the simple formats are centered around a key/value pair model which can be expressed as properties directly. More complex models are commonly now expressed as XML, which can always be represented as a tree, in particular the common Node tree used in the Document Object Model (DOM). Such an XML tree again can be transformed by turning any node into an information object and any edge into a relationship – or what might be more efficient, represent just DOM elements as information objects and there text content as raw content, any attribute into a property and only the links to other elements as relationships, e.g., the grouping.

due to failures in the system but also due to inconsistencies if the second approach is used. Or the other way round: If the content acquisition is performed by crawling the web, the user might be annoyed if results are unavailable – but may nevertheless accept them as it is a consequence of the dynamic nature of the web. But as soon as the content is managed more carefully, the tolerance for inconsistencies will certainly decrease.

4.3 Maintaining Consistency

To limit the problem of inconsistencies, there are two simple technical approaches:

- *Increase frequency of consistency checks:* For crawled content, increase the frequency of crawling. If not just simple crawlers are used, but hooks can be established, e.g., in a Publish–Subscribe fashion, ensure timely reaction to notifications.

- *Local Caching:* Cache the content locally and allow to access the content from the cache.

There are two other reasons to keep some content local, which also have great impact on the user satisfaction because of speed for the result delivery. For providing similarity search, as described in Section 2.3.1 / Figure 2.2, it is necessary to extract features of all content. Of course, theoretically, they could get extracted on the fly from the content at the time a search is performed. But this would require that all content is accessed at the time of retrieval and search cannot be finished before all content has been processed and evaluated. Opening a single image usually takes a few milliseconds, depending on the size of the image and how fast it can be read from the storage media, whether a compression algorithm is used and how complex it is to decode, and the libraries that are used generate an in-memory representation of the image that can be processed. Usually the extraction of features take additional time in the range of milliseconds to seconds per image, depending on the resolution at which the image is processed and the used algorithms and the resulting amount of the processing needed to extract a feature. Thus, the minimal time to process all images of the collection sequentially grows linearly with the size of the collection. As a result, it is impossible to extract the features for search on the fly for collections of significant size in a small number of seconds that the user might be willing to wait for a result. It is therefore essential to extract features ahead of time. This can be done as soon as the image is inserted to the collection and reused for any subsequent search over the collection.[6]

For this, the extracted feature has to be stored somewhere. The features are frequently just a sequence of real numbers and therefore commonly represented as an array of floating point values. Therefore, for simplicity, it might be treated as other binary content – and stored like the raw image content.

[6]Of course, instead of processing all images of the collection sequentially, this step can be parallelized on multiple computing nodes to reduce the overall time of processing. However, this approach would create additional coordination overhead that can still slow down the response time. And parallelization and performing feature extraction ahead of time are not mutually excluding: They are frequently combined and Chapter 10.5 will provide some more details on that.

Another kind of raw content needed for acceptable user experience are thumbnails generated from the images in the collection. Just like for feature extraction, opening just the result images to display them on a result screen might take significant time, depending on the size of the real images. Creating a version of the images on the fly with reduced resolution and a dimension that fits to the area available for each result in the result screen of the user interface can also cause unacceptable delays; therefore should be performed when or directly after content is inserted or altered rather than at the time of a user waiting for it.

These two kinds of binary content are necessary due to requirements outside Content Management – features to achieve acceptable time for Query Execution, thumbnails to achieve acceptable time for result presentation in the User Interaction – but they are closely related to Content Management[7], in particular if one considers updates of the content. In this case, any updated content should result in a corresponding update of the features and thumbnails. By storing not only binary content, but also *relationships* between the content and derived data, these relationships can be used to trigger the required updates to enforce consistency.

4.4 Relation to Existing Models and Technology

Of course, the Generic Storage Model described on page 81 is not the first generic model and the two example content models built on top of it are not the only possible content models. In fact, there do exist a number of existing models for digital libraries and the next paragraphs will compare some prominent examples to the models presented in Section 4.1.1. In addition, aspects of implementation of the models will be discussed.

The DILIGENT Storage Model as described in [Candela et al., 2007, pp. 67f] and a little more detailed in Chapter 9.1 provides a three layered approach in which the Storage Layer provides a stripped-down version of the generic storage model: Information objects, raw content, properties are handled as described in Section 4.1.1 on page 81 and it also supports (directed) relationships between two information objects, but does not support arbitrary properties for relationships. There are some pre-defined properties available for relationships and in the Content Layer which is build on top of the Storage Layer, also a model for documents and collections suitable for digital libraries for Earth observation research and other applications. The third layer is the so-called Base Layer which provides the functionality to store relationships and properties in a relational database and the raw content in either in BLOBs inside the relational databases or files in the filesystem or on a Storage Element (SE) in the Grid.

The DELOS Reference Model [Candela et al., 2008a, pp. 36–39 and pp. 77–79] describes a reference model for digital libraries of which the Content Domain is centered around Information Objects with pre-defined relationships like *hasPart*, *hasManifestation*, *hasEdition*, *hasMetadata*, *hasAnnotation*, etc. It therefore provides in addition to many other important aspects of digital libraries one particular content model similar to the one presented on page 82. However, there are some differences in the detailed terminology and degree of detail, e.g., it does not consider replicas or extracted features

[7] In the illustration in Figure 3.1 , they are issues of the overlapping areas with Content Management.

and does not name particular properties. It does also not consider any of the extensions to support particular image search tasks, what may be explained by the fact that it was focusing on a reference model for any digital library – therefore not providing very detailed support for particular types of content and potential use cases for them.

Another model for content in digital libraries is 5S [Gonçalves et al., 2004] where 5S stands for *streams*, *structures*, *spaces*, *scenarios*, *societies*. 5S provides a formal model for entire digital libraries, the content itself therefore is covered by the first two concepts: *Streams* represent the raw content. A *digital object* [Gonçalves et al., 2004, p. 294f] has a universally unique handle and a set of streams which form it's consisting parts. Furthermore, the digital object has structure, between the streams (how the parts are arranged) as well as within the streams (structure to segments of a stream). For digital objects, the content model of 5S remains very generic and thus, when mapping the digital objects of 5S to the Generic Storage Model, every stream becomes raw content of an information object of it's own. The digital object itself becomes an information object which has relationships to its streams to represent the internal structure. On the next levels of granularity however, 5S provides much less flexibility:[Gonçalves et al., 2004, 295] defines *Collections* as sets of digital objects, so formally it does not define any order for it and does not allow for sub-collections within a collection. A *metadata catalog* provides descriptive metadata specifications for each digital object in the collection. *Repositories* encapsulate a set of collections and the specific services to manage and access the collections. This defines a fairly simply content model for collections and repositories which would be directly translated into two kinds of information objects and relationships: Collections with an *is member of* relationship to the digital objects and repository with which the collections are bound by an *is managed by* relationship.

For the context of interoperability of Enterprise Content Management systems, the Organization for the Advancement of Structured Information Standards (OASIS) has releases the Content Management Interoperability Services (CMIS) standard [OASIS, 2010] which defines a data model as well as services and bindings to SOAP[8] and Representational State Transfer (REST) using the Atom convention. The data model describes four base types of objects, Document Objects, Folder Objects, Relationship Objects, and Policy Objects. Each object has a set of named properties of specified type. Therefore an object in CMIS is very similar to Information Objects in the Generic Storage Model. Relationship Objects in CMIS can act like relationships of the Generic Storage Model, however they are left optional in the standard. Instead, the content model defined in the CMIS explicitly mentions Folder Objects as a logical container for collections of the two file-able object types Document Objects and Folder Objects. It also defines that document objects are allowed to have a Content Stream –or in other words, raw content– and that Document Objects can have *Redentions* to provide thumbnails or alternate representations. The content model described in the CMIS specification is therefore an extension of the simple content model described on page 82 by allowing subcollection/subfolders and alternate representations/redentions, which would be modeled in the rich content model on page 82 as membership to collections and as manifestations, respectively. CMIS also allows versioning of documents, but does not support versioning of any other kind of information objects, in particular

[8]SOAP stands for Simple Object Access Protocol, cf. [W3C, 2000]

Folder and Relationship Objects. Due to its focus on Enterprise Content Management, another optional feature mentioned explicitly in the CMIS standard is the possibility to use Policy Objects to control apply CMIS implementation specific policies on controlled objects and to define Access Control Lists (ACLs) to set permissions on objects in a repository.

The International Committee for Documentation (CIDOC) of International Council of Museums (ICOM) proposes also a very rich model that has a very wide overlap with digital libraries: The Conceptual Reference Model (CRM) is a standard for cultural heritage that was recently adopted by the International Standardization Organization (ISO) as ISO 21127:2006 and provides very rich structuring mechanisms for metadata descriptions [Ntousias et al., 2008]. It also uses the term *Information Object* to describe identifiable immaterial items similar to the DELOS Reference Model [Candela et al., 2008a, p. 62], but as it is intended for museums, the main artifacts of interest are physical objects, and therefore also information like when they where created and modified, where they have been kept. The model also provides the possibility to describe physical objects not only textually, but also visually through images. As it is a common need in archeology and related areas to express in detail which historical person or event is depicted and how this relates to other findings and known facts, the model provides a rich ontology to express such aspects. The entire model can be expressed as a graph and expressed in the Resource Description Framework (RDF) [Nussbaumer and Haslhofer, 2007].

In RDF [Lassila and Swick, 1999] the basic data model consists of *resources*, *properties* and *statements*.

- *Resources* are the counterparts to *information objects* in the Generic Storage Model and everything that is being described by RDF expressions is a resource.

- A *property* in RDF is a specific aspect, characteristic, attribute, or relation used to describe a *resource*. Compared to the Generic Storage Model, RDF does not distinguish on this level between relationships and properties, they are all called "property". However, for any property the specific meaning and permitted values have to specified in line with the RDF Schema specification. Therefore the different semantics between property and relationship as it exists in the Generic Storage Model is not lost.

- A specific resource together with a named property and the value for that resource is called an RDF *Statement*. The three different parts of a statement are called *subject*, *predicate*, and *object*. The subject is the resource, the predicate is the property, the object is the property value for the described resource – where the value can either be another resource or a *literal*, e.g., a primitive datatype. As a statement always consists of these three parts, it is frequently also referred to as an RDF triple.

To compare RDF to the Generic Storage Model, one has to consider that RDF is tied to a particular syntax which can be expressed within the Extensible Markup Language (XML), while the Generic Storage Model is still left abstract with regard to how it should be preserved and exchanged. As a consequence, for RDF any resource is identified by a Uniform Resource Identifier (URI) and any literal must be expressible in XML.

As XML itself is very versatile, this does not restrict which information can be expressed in XML and therefore RDF; it only restricts how efficiently this can be done.

In effect, RDF can be used to implement the Generic Storage Model when every information object gets assigned a URI and the raw content of information objects is either maintained inside the XML, e.g., embedded in Base64 encoding or outside the XML but accessible and addressable through a URI[9].

Vice versa, everything that can be expressed in RDF statements can also be represented in the Generic Storage Model, e.g., by turning the subject into an information object, every predicate into a relationship and any object into another information object, where references simply point to the information object of that resource and for literals, the value becomes the raw content of a new information object. Other, in practice more efficient transformation rules may prefer storing statements where the object is a literal in the Generic Storage Model directly as properties of the information object.

As RDF can be mapped to the Generic Storage Model, also models on top of it can be mapped. This is in particular the case for the Abstract Data Model which is part of the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) standard [Open Archives Initiative, 2008]. By providing the resources via appropriate URIs and mapping to aggregates as this standard defines, interoperability between digital repositories can be achieved. [Tarrant et al., 2009].

Fedora[10] is a popular digital library software package that relies on RDF for modeling complex objects and their relationships [Lagoze et al., 2006]. This model has been modified a bit towards the Fedora Digital Object Model [Davis and Wilper, 2011], in which each object has a persistent, unique ID (PID), system-defined descriptive properties necessary to manage and track the object (object properties), and multiple content representations (datastreams). The Fedora Digital Object Model is therefore even more similar to the generic content model, however, there are still some noticeable differences:

- Any user- or application-defined property has to be encapsulated in a datastream.
- Any relationship to another internal or external object has to be encapsulated in a datastream.

Datastreams [Davis and Wilper, 2011] itself have a similar structure to the Fedora Digital Object Model, but may contain relations and raw content (bytestream content), which may be XML content for which the control group can identify particular types that are understood for internal use. The Content Model Architecture (CMA) can be used to define structural models within a Content Model Object. Compared to the Generic Storage Model, the Fedora Digital Object Model adds more predefined fields, but still requires to encode non-predefined properties and relationships in datastreams – making it harder to traverse the model as more elements serialized to datastreams have to parsed to be understood as structural primitives.

[Candela et al., 2009a] introduces a typed compound object data model and a digital library management system named Doroty (Digital Object RepOsitory with Types) that provides an implementation of the typed model. Similar to the Generic Storage Model, it known three elementary types:

[9] Which therefore has to be a Uniform Resource Locator (URL) at the same time.

[10] `http://fedora-commons.org/`

1. *Atom Type* is used to store files and corresponds to raw content.
2. *Structure Type* in a simple form may contain sequences of attribute-value-pairs, which would correspond to properties. Additionally, Structure Type may also represent more complex structures such as trees of records.
3. *Relation Type* represents binary relationships between objects, therefore correspond to relationships.

In the Generic Storage Model, any Information Object and Relationship has an attribute type. On top of the Generic Storage Model, arbitrary domain-specific content models can be defined which correspond to "types" in [Candela et al., 2009a]. The typed compound object data model is not more expressive than the Generic Storage Model, rather, it describes a variant with a particular proposed implementation aspect: [Candela et al., 2009a] emphasizes on the ability of the digital library administrator to define types with a particular structure and any object in a digital library will be an instance of a type that will always be member of a set, that aggregates the instance of types. This allows the enforcement of type-correctness as well as optimized storage of sets as it contains only instances of the same type. This implementation is similar to object-oriented databases following the ODMG 3.0 standard in which a type can be defined as a *class* and all instances will be aggregated in an *extent* [Cattell and Barry, 2000]. For an implementation based on the Generic Storage Model and dedicated content models, type-consistency is usually enforced at the level of content models, e.g., in services and libraries that provide the content model while rudimentary checks and optimizations on the storage level remain possible in the implementation of storage backends that evaluate the type attribute.

Of course, RDF is not the only possibility to implement the Generic Storage Model or a particular content model developed on top of it. In order to show alternatives, we will briefly illustrate some other solutions of which probably the simplest solution can be to use the filesystem directly: This is particularly appealing if the content is generated or imported and later used primarily locally as it is common for personal image collection management. In this case, and if the simple model described on page 82 is mostly sufficient, the main important relationship is *is member of* a collection and collections can be represented by folders in the filesystem. Information objects and their raw content are represented as image files inside the folders, any properties can be stored inside the file (if the image format allows to be used as a container for such information, e.g., as the Tagged Image Format (TIFF) or JPEG allow for Exif metadata) or in a separate file with same base name, but different extension to distinguish from the actual image (as it is done for so-called XMP sidecar files). Extracted features and alternative representations like thumbnails can be stored in subdirectories that follow a similar naming convention.

The use of the plain filesystem gets more problematic when multiple hierarchies for organizing the content must be available concurrently or the history of changes must be preserved. Of course, such problems can be coped with by adding yet more conventions about naming and where to keep information – just as Revision Control System (RCS)[11]

[11]RCS was the predecessor of was Concurrent Versions System (CVS) which is still used as a version control system for a significant number of software projects – and still was and is based on a number of concepts from RCS, but adds for instance branching.

also able to provide versioning of files basically inside the filesystem. However, the used filesystem might easily become the bottleneck for such "abuses" and it might be more elegant to rely on a more versatile basic. For instance, the distributed version control system Git[12] is in its core much closer to the Generic Storage Model: As described in [Vilain, 2006], it distinguishes between the four essential types *blobs*, *trees*, *commits*, and *tags*. Commits can hold several properties like author, subject, and description. Blobs can be used to store arbitrary binary content. Trees can relate individual files therefore blobs with each other. Common operations of a version control system are usually performed very fast in Git compared to other state-of-the-art version control systems. And due to the similarity to the Generic Storage Model, it should require little changes to modify any implementation of Git[13] to turn it into a basic implementation of the Generic Storage Model on top of which arbitrarily complex content models are possible – and due to the origin of Git as a distributed version control system, may get transactional isolation of changes as well as distributed use and conflict resolution.

However, the latter properties are certainly where traditionally database systems excel. The Generic Storage Model can, of course, also be implemented on top of a database, e.g., using a Relational Database Management System (RDBMS) in which one table holds the identity of information object, one table holds the properties of the information objects, one table holds the relationships, and another table holds the properties of relationships. Raw content can either be stored in the database directly as BLOBs or outside and just store a URL in the database how to access it.

For Object-Oriented Database Management Systems (OODBMS), the mapping is even more trivial as their data model is identical to object-oriented programming languages and therefore providing persistence and query strategies for objects with attributes and references between objects directly. Most OODBMS accept binary data like any other datatype, therefore they are also able to handle the raw content – for which there remains also the option to store it outside the DBMS and just keep the URL in the DBMS. This also applies to systems dedicated to *Graph Databases*, e.g., as proposed in [Iordanov, 2010] and proposed as a basis for semantic image retrieval in [Ganea and Brezovan, 2010]. Support for storing raw content is the core functionality of the Content Repository API for Java (JCR) based on JSR-170 (Version 1) [Java Community Process, 2006] and as JSR-283 (version 2) [Java Community Process, 2009], uses as a content model a hierarchical graph consisting of nodes and child nodes. These nodes have properties.

Another last option that can be mentioned just briefly becomes possible when the content is stored outside the system boundary and just referenced by a URL, is to use existing text-retrieval systems. For instance, Apache Lucene[14] allows to store individual *fields* for documents inside its indexes and also allows to retrieve all values of all fields for a particular document. This makes the mapping of properties from the Generic

[12]Which has been developed by Linux Kernel developers as a replacement for the proprietary BitKeeper system and has now (at the time of writing) replaced in many other Open Source projects as well as closed source projects previously existing version control systems.

[13]There are several implementations of Git available, in particular the original C implementation which was started by Linus Torvalds available at `http://www.kernel.org/pub/software/scm/git/docs/` as well as for instance a Java implementation JGit available at `http://eclipse.org/jgit/`.

[14]`http://lucene.apache.org`

Storage Model to fields straight forward. The URL to the content can be stored as a field as well. To represent direct relationships, one can employ the fact that text retrieval commonly uses inverted lists for efficient lookups. And as fields cannot have only a single value but arbitrarily many, relationships can be stored there as well. E.g., the names or IDs of collections in which one particular document is member can be stored in a field named "is member of". To retrieve all documents inside a particular collection, it is sufficient to issue a query with the name/ID of the collection for the field "is member of". However, this approach does usually not work elegantly for deep structures as the inverted lists are commonly designed to handle efficient lookups and scan but don't support construction of "subqueries", that is, do not by itself provide the ability to use the result of a first query as the input for another query. This can make traversing a tree structure a tedious task.[15]

All these examples have shown that there are many ways to implement the Generic Storage Model. Some of them have some limitations, so cannot handle the entire model, but may be limited to certain relationship structures – but on the other hand may have benefits like robustness, simplicity, or ease of integration with pre-existing infrastructure. On the other hand we have seen several existing content models for digital libraries which may be very elaborate. And we have also seen that all of them can be represented within the Generic Storage Model. So one question could be, whether one should use just the Generic Storage Model directly or even a particular implementation, e.g., RDF to manage the content? Or shall one rather decide or come up with one reference model that is detailed enough to handle all requirements of digital libraries?

4.5 Is there a "One size fits all" Model for Digital Libraries?

On one hand, the development of a higher level reference model capable of handling all needs of digital libraries is very challenging. The example content model presented on page 82 is already very complex, but it still does not express all possible aspects, e.g., multi-linguality (which is one of the aspects that are commonly of great interest in the use of CIDOC CRM). On the other hand, such higher level models make communication between stakeholders much easier as for instance, developers and users will be able to understand "collection" and "document" better than information objects and relationships.

By using the Generic Storage Model on a conceptual level, a layer of abstraction between the detailed content model and the low-level implementation of storage to files or databases has several advantages:

- It allows to replace one implementation, e.g., filesystem-based implementation with an RDBMS to handle relationships and properties when the latency of the filesystem becomes a bottleneck.

[15] An alternative would be to store inside a field the entire path instead of just the last layer, or in other words: the entire hierarchy from the root to the leaf and query for partial matches if needed.

- It allows to add new types of information objects, properties, and relationships as the requirements of the digital library evolve over time.
- It also forms a better basis for reusability as the low-level implementation can be reused even with a different higher level content model for a different digital library without carrying around the added complexity necessary to integrate all different requirements from the beginning.

To illustrate briefly these aspects, it may help to present some examples: Personal image archives have different needs than public galleries. For instance, public galleries need to provide different permissions to different groups of users. Public galleries may need to understand different kinds of metadata but also expose the metadata in various formats. On the other hand, private image archives may have less "curation" of the content – users may simply want to maintain all images they have gathered over years and it will be impossible for them to provide manually much additional metadata on the individual items simply due to the amount of images and lack of time and memory. Where in general it seems to be a simple task for system to deal with *less information*, so here less metadata, it has the side-effect that tools that worked well for perfectly annotated material become unusable and with them, the entire system loses value.

If we want to apply generic digital library systems to particular domains, e.g., medical images, we will commonly experience that some concepts are lacking in the model. For instance, for medical images the common access hierarchy is patient, study, image, region of interest. One could try to map patients to collections and study to subcollections, however, the access permissions on patient information in medical environments opposed to generic collections is much more dynamic and based on the relationship between the user of the system and role towards the patient: A doctor currently treating a patient should have more viewing permissions than an unrelated doctor. The ability to add such domain-specific concepts to the high level model without having to re-implement may therefore be highly beneficial.

4.6 What is needed to support images as content?

The Generic Storage Model is not only limited to digital libraries that contain images – it is generic enough to hold also other content. What is of particular importance for supporting images is the support for raw content as a first-class citizen: In more traditional, text-oriented digital libraries, searches are usually metadata-driven or fulltext-searches. The raw content is commonly needed only when the user accesses individual items; in the list of search results it may be sufficient or even better to present the metadata prominently and relevant text snippets – as a placeholder, a link to the raw content in a particular format can be presented.

For image content and content-based access, textual representation is not sufficient: As described in Section 4.3 on page 89, features and thumbnails are needed. Both are needed *before* the user selects individual results. More details on the features and the execution of similarity search will be given in Chapter 5; the presentation of results which will make use of thumbnails is subject to Chapter 6.

For Content Management, this imposes the need to:

- Store binary content: the image files, extracted features, generated thumbnails.
- Maintain consistency between the image and derived data.
- Be prepared to deliver multiple items in short time, in particular the thumbnails of all results of a search.

4.7 Handling Queries for Exact Matches on top of Generic Storage Model

The Generic Storage Model can also assist in implementing some core functionality of digital libraries. In order to support facetted search and also plenty of internal functionality, the system must be able to provide exact match and range queries. For instance:

- Retrieve all images which are member of some collection which have been uploaded by a particular user.
- If features are extracted in some background task, it is very helpful to be able to ask for all content that has been added or updated after some particular date.

Given the Generic Storage Model, it is possible to define also generic query primitives to traverse the graph expressed by relationships and match values of properties. For the implementations of the Generic Storage Model this may also allow to convert such primitives into particular languages that are available for the used technique, e.g., SQL for relational databases, SPARQL for RDF, common text query primitives for text-retrieval engines, or at least listing subdirectories and checking the existence of a file with particular name for the filesystem.

Of course, the generic translation from such a search primitive might not be as efficient as highly optimized statements – but it may provide a starting point and wherever performance is crucial, there might anyway be the need to add auxiliary data structures to speed up the execution. This is in particular necessary for retrieval by similarity rather than only exact match. The execution of searches including similarity is the subject of the following chapter.

5

Query Formulation and Execution

The actual search functionality is dominated by two very different aims:

- *Expressivity and quality of results:* Providing the ability to the user to formulate what are desired results, support various input methods, define the matching tolerance that the user is willing to accept or needs given the input that can be provided. (This aim will be subject of the discussion within this chapter.)

- *Deliver results quickly:* Make the evaluation of the query fast such that the users don't have to wait – and will probably reformulate their queries if they are not happy with the results and re-query. (This aim will be subject of the discussion within Chapter 10.5.)

These aims usually are at least partially conflicting as commonly richer queries using more inputs need more time to evaluate than simple queries with a single input.

The first item depends highly on providing a query model that serves the needs of the particular task that the user wants to perform and enables the information seeking strategies that are appropriate. In order to handle the input that the user provides and support invariances the user input and task needs, suitable features have to be used.

Suitability of any feature depends heavily on the following two aspects:

- *What is the entity of the feature?*

  Is it the image as a whole or does the feature correspond only to parts, for instance, real world objects. This is important as we have seen in the previous chapter that many tasks may be driven by particular objects, people, animals, buildings... If a feature does not correspond to just this object, the search results will be of little help to the user. For instance, if the user is performing a task like finding all images in collection that contain an animal (Task 2 on in Chapter 2.2 on page 24), features should correspond to the animals inside the images in order to be helpful in detecting them. In contrast, for Task 1 to find a known image, it might be much more helpful if the feature corresponds to the entire image – as this would avoid retrieving potentially many images that contain the same objects, but are not the particular image the user was looking for. And in some cases, it might be even the

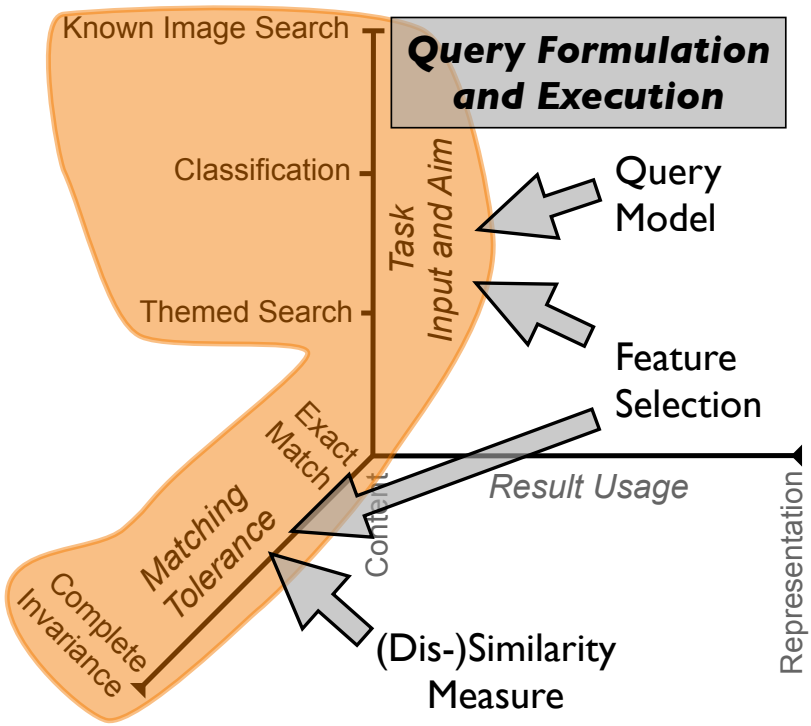


Figure 5.1: Aspects in ITM covered by Query Formulation and Execution: Main importance is handling the Matching Tolerance, but this wouldn't be possible without an appropriate query model for the task and selecting the appropriate features for the task.

scenery that is of greatest importance to the user, like in the themed search for a beach at sunset in Scenario 2 in Chapter 1.4.2 on page 15.

- *What is the (visual) attribute captured by the feature?*

  In some cases, the user may be interested in matching the input as close as possible. In other cases, there can be good reasons why there should be some deviations allowed as described in detail in Chapter 2.3. In Task 1 in Chapter 2.2 on page 23, the input for known image search was only available in an uncolored version; the sought image has colors - therefore any suitable feature should ignore differences in in color in the search. To further be able to control the matching tolerance, not only the feature, but also the measure of (dis-)similarity needs to be adjusted to the users input and desired results.

These aspects are illustrated in Figure 5.1.

(a) Original Image (b) Blossom Segment (c) Bee Segment

Figure 5.2: Segments of an image: Entire image (a), semi-automatically segmented images of the blossom (b) and the bee (c).

5.1 Regions for Extraction of Features

The main distinction for regions for image retrieval is between global features, that are computed from the image as a whole and local features, that are extracted from parts of the image. During the matching process, local features allow a very broad range of invariances as listed in Chapter 2.3.3, e.g., to allow invariance to translation by identifying the same parts in different locations in two different images. If this ability exists, also scale and rotation invariance can get supported much easier and any background can be ignored (if desired). Due to this added benefit, the discussion will start with the more versatile local features.

5.1.1 Segmented Regions

If feasible, the preferred result of the image analysis would be a proper segmentation of the image. That is, each segment identified corresponds to a real world object that is of interest to the user.

If we take one image from our small collection in Figure 2.1 on page 23, for instance 2.1(h) which shows a blossom and a bee, we would expect a segmentation like the one presented in Figure 5.2 (b) and (c). Early retrieval systems that provided such segmentation were VisualSEEk [Smith and Chang, 1996], Blobworld [Carson et al., 1999, Carson et al., 2002], SIMPLIcity [Wang et al., 2001] and Windsurf [Ardizzoni et al., 1999].

However, in general it is still very hard to achieve good segmentation results just from generic images as many real-world objects consist of several visually different regions. For instance, the bee in Figure 5.2(a) consists of dark brown/black as well as yellow/light brown regions, thus leading purely visual approaches frequently to oversegmentation. Another related problem exists in the separation of objects: In particular in nature the appearance with color and shape may not be coincidental but evolutionary beneficial when it is particularly hard to distinguish individuals from their common environment. For instance, many animals have developed fur and skin color and pat-

terns that provides some camouflage and therefore protects against predators or helps attacking their prey.

The problem may even increase when many instances of the same kind appear in the same image and overlap each other like the blossoms in 5.3(b). In order to achieve better segmentation results, manual segmentation through users or semi-automatic segmentation like GrabCut [Rother et al., 2004] can be used and by this exploit the understanding the user has of the image.

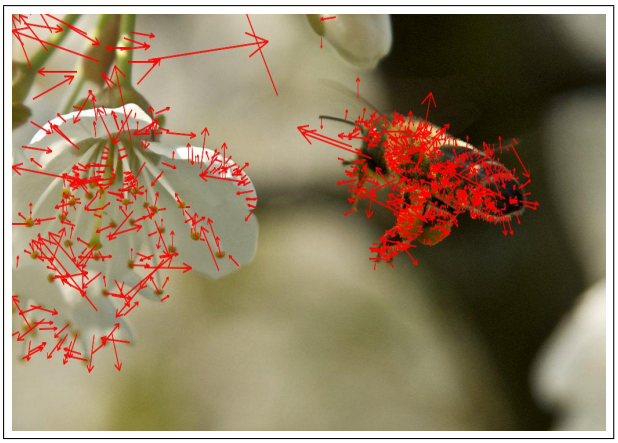
Another approach for domains in which the expected objects in the image and the desired use is predetermined can exploit models of the objects and fit them into the images as it is very common in medical image segmentation (e.g., cf. [Heimann and Meinzer, 2009]) or in face detection (e.g., [Hjelmås and Low, 2001, Romdhani et al., 2006]). [Flickner et al., 1995, pp. 26f] describes also tools to perform either fully automated unsupervised segmentation for a restricted class of images like museum or retail catalogs, in which a small number of foreground objects is presented on a generally separable background, or semiautomatic flood-fill tools and so-called spline snakes methods. If tasks can be anticipated from the nature of the image collection, both approaches are very suitable; they are far less applicable to large-scale generic image collections for which there exists little a priori knowledge about what users might query for. And even when such approaches are used, artifacts may still remain in segmentation, e.g., the semi-automatic segmentation of the bee shown in Figure 5.2(c) does not only miss part of the sprawled feet and antenna, but also the wings are problematic as they are translucent and therefore have the color of the background.

5.1.2 Salient Point Detection

Instead of identifying regions that correspond to entire objects, to better support affine invariance only small regions or points are used which can be easily found again in different images showing the same objects. The term "salient points" was coined in [Loupias et al., 2000] to differentiate from the established term "interest point", that has been understood / used as a synonym for corner point – even though interesting regions may not necessarily lie on corners of objects, but also inside. "Key" [Lowe, 1999] or "keypoint" [Lowe, 2004] are other terms used to describe the concept of points for which detection is highly repeatable and provides some robustness to common variations, in particular invariance to affine transformations and noise.

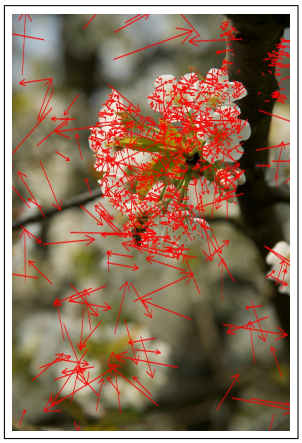
For the purpose of retrieval, the interest is actually not limited in an individual point and the value of the pixel at this point, but small regions around the salient point. Therefore [Kadir and Brady, 2001] uses the term "salient regions" and the "scale" determines the size of the region of visual saliency. Some examples of detectors for salient points / regions are described in [Lowe, 1999, Van Gool et al., 2001, Gouet and Boujemaa, 2001, Mikolajczyk and Schmid, 2001, Matas et al., 2004, Mikolajczyk and Schmid, 2005, Bay et al., 2006].[1] Figure 5.3(a) shows the Scale-invariant Feature Transform (SIFT) keypoints [Lowe, 2004] using the Difference-of-Gaussian (DoG) keypoint detector derived from the image in Figure 5.2(a). As can be seen in this example, many regions or

[1] `http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html` collects a number of implementations for the salient region / keypoint detectors.



(a) Salient Keypoints

(b) Another Image



(c) Keypoints

(d) Matching Result

Figure 5.3: Salient Keypoints using the DoG SIFT keypoint detector in (a) and on another image (c). The second image itself is shown in (b) and contains the same blossom and bee, but also more blossoms. (d) shows a matching to illustrate that the corresponding area of the original picture 5.2(a) was found based on the SIFT features in (a) and (c).

points can be found per object. As correspondence to objects is not as direct as it would be required for segments, [Smeulders et al., 2000] uses the terms "strong segmentation" for segmentation and "weak segmentation" when salient points and features are used.

To illustrate the ability to retrieve the same salient keypoints in different images, a different image is shown in Figure 5.3(c) and the SIFT keypoints derived from it in Figure 5.3(c), using the same extraction parameters. Figure 5.3(d) shows the matching that follows mainly the approach proposed in [Lowe, 2004] for matching, and performs RANSAC filtering to be able to determine a affine transformation from the found matches – even if some matches are misleading. The red triangle illustrates the matched area after filtering. We have described the details of the processing

in [Springmann and Schuldt, 2008] and more aspects of the matching will be discussed in Section 5.4. For now, it shall just be seen as demonstration of the general feasibility of the matching based on keypoints within some bounds. As already visible the image, only the blossom was matched – the bee itself has not been detected properly using SIFT and the fixed particular parameters without any adaptaion.

The easiest form of a feature taken at a particular point can be considered to take the area as a patch. Such an approach is proposed in [Deselaers et al., 2005], together with discriminative training.

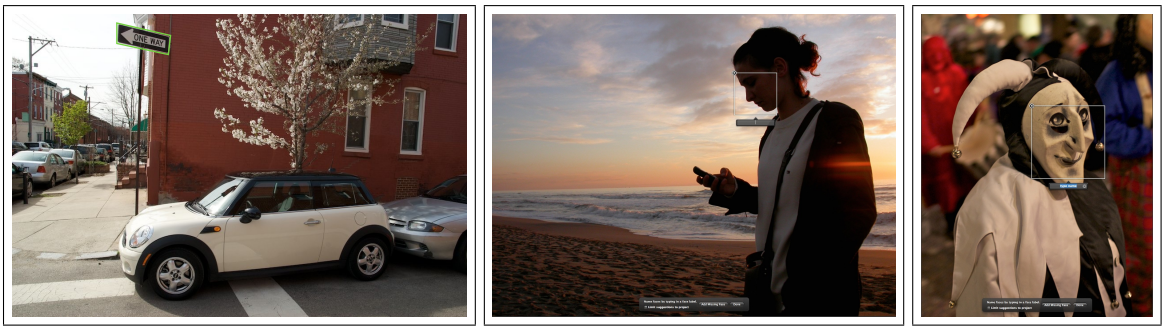
Training is commonly combined with keypoint-based approaches as the keypoint itself does have little semantic meaning - it does not correspond directly to a real world object. By training, keypoints are selected that carry information about the object or class it belongs to: Analyzing several images of the same object or several instances of the same class taken under various conditions and probably also from different angles, the system learns to identify what is are characteristic properties of the object / class, and which keypoints refer rather to artifacts or clutter in the image as it was only found in one example but not in any other – and therefore even if it wouldn't be clutter, could not act as a reliable point anyway.

This allows to detect objects even if they are partially occluded just based on the remaining visible keypoints and the trained model of the entire object.[2]

In contrast to these salient points without direct correspondence to (meaningful) parts of real world objects, [Smeulders et al., 2000] mentions also objects of a fixed shape, like signs or an eye, for which localization can be sufficient. Figure 5.4(a) illustrates a sign and faces highlighted inside images. For retrieval like in the Scenario 2 in Chapter 1.4.2 on page 15 for a beach without a person, it might already be helpful to be able to perform a facetted search in which images with faces are excluded. For other tasks, in particular when particular people or objects shall be retrieved, this will certainly not be sufficient. However, the location of the region –even if not properly segmented– can be used to initialize dedicated building blocks for automatic identification or asking users for manual label. The latter may always be needed for optimal results, as automatic approaches will always fail to identify properly if a person is present in a picture, but not fully visible; this will certainly be the case if the person is in disguise like in Figure 5.4(c).[3]

[2]ClassCut [Alexe et al., 2010] provides an improved, unsupervised segmentation algorithm for trained classes. Therefore it does no longer require input from the user as GrabCut [Rother et al., 2004] does, but it can still not provide a complete alternative approach for generic segmenting regions for feature extraction: ClassCut requires training on all classes that it should segment, therefore all classes would need to known at indexing time and enough training examples have to be available. However, for frequently used classes and domain-specific applications where only some classes are relevant for retrieval, this approach might be used and therefore uniting strong and weak segmentation.
[Rabinovich et al., 2007, Galleguillos et al., 2008] use multiple stable segmentation from normalized cuts and trained class information combined with context information to improve accuracy of class labels. [Roth and Ommer, 2006] describes another approach that uses ensembles of segmentations to achieve stable segmentation for object recognition.

[3]"Tagging" people in images can be done manually or automatically. In Figure 5.4(b), manual tagging was used – as it is common in particular in social networks (cf. [Odio, 2010, Camp, 2009]). To reduce the amount of manual work required for tagging, systems provide now frequently automatic face detection (so detecting if there is a face in the image and where; cf. *Object Detection* in Chapter 1 on page 34) and



(a) Traffic Sign (b) Manual Face Label (c) Detected "Face"

Figure 5.4: Regions based on signs and faces: In (a), the sign of the one-way street has been highlighted and the face in (b) manually "tagged" as it is common for shared pictures in social network. The "face" of in (c) has been automatically detected using a commercial photo management software. In this case, it is not a human face, but a mask used at the carnival in Basel.

5.1.3 Static Image Regions

Basic spatial orientation inside an image can get analyzed, even when the content is not taken into account for defining the regions. For instance, to express that the top left part of an image should be similar, it is sufficient to make sure that the top left part of both images is compared. Such simple regions are used in many popular features, e.g., for the MPEG-7 Edge Histogram Descriptor (EHD) as a first step the image is sub-divided into 4×4 sub-images from each of which local edge histograms are computed [Manjunath et al., 2001, p. 714] and for the MPEG-7 Color Layout Descriptor (CLD) the input picture is divided in into 8×8 blocks [Manjunath et al., 2001, p. 710f]. Technically, this is easy to perform and already assists in providing some basic invariances: By dividing the image always into the same number of regions and analyzing them independently, some effects of scaling the image as a whole can get compensated. And through the same number of regions along the vertical and horizontal axis, any rotation by multiple of 90 degrees can also be expressed as a permutation of the regions.

Figure 5.5(a)-(d) illustrates those regions on the images from Figure 5.2(a) and Figure 5.3(b). What can be seen as suboptimal is, that due to the static partitioning into quite a significant number of regions –16 and 64, respectively– real world objects might get split up into many several regions and even fairly small translations may cause the objects to fall (partially) into a different region, which will have impact on the derived features and similarity based on regions.

---

may even suggest who's face it might be (by comparing the detected face to previously tagged faces; cf. *Object Identification* in Chapter 2 on page 35) – so leaving the user just to confirm or correct the suggested name. For Figure 5.4(c), the face detection of the commercial software Apple Aperture 3.1 was used – suggestions were not provided as there was no person with such a mask previously tagged. Similar features are also available in other products, e.g., Google Picasa [Google Picasa Help, 2011a] and Picasa Web Albums [Google Picasa Help, 2011b], Facebook Photos [Mitchell, 2010], and Aperture's smaller brother Apple iPhoto since the 2009 release.

(a) 4x4 Regions (b) 4x4 Regions (c) 8x8 Regions (d) 8x8 Regions

(e) Fuzzy 5 (f) Fuzzy 5 (g) 8/4 ARP (h) 8/4 ARP

Figure 5.5: Static Regions that are defined independent of the image content using 4×4 non-overlapping rectangular regions in (a) and (b), 8×8 in (c) and (d), five fuzzy regions that blend from one to the other at the border in (e) and (f), and ARP with in 8 angular and 4 radial partitions in (g) and (h).

To reduce this effect, a partitioning scheme as proposed in [Stricker and Dimai, 1996, Stricker and Dimai, 1997] can be used: Figure (e) and (f) use five "fuzzy" regions. Due to the smaller overall number of regions, the impact of small translations are already reduced. Furthermore, as the outer regions are distributed towards the edges, small changes in the scale of (parts of) the image as caused zooming as described in 2.3.3 on page 56 will only shift pixels between the outer and the center region. "Fuzzy" means in this context, that the borders between the regions are not absolute, but that the impact of a pixel is slowly decaying in the border area before ultimately becoming zero. In the illustrations, this is visible as colors blending into another instead of sharp edges. For the computation of features this also means, that pixels along the border do have less impact on individual regions – thus if "misplaced" do not have the same negative effect. Such small misplacements can be caused by any small change of scale, rotation, or translation. Finally, the elliptic center region follows the heuristic approach that frequently, objects towards the center have greater importance to the viewer than other regions of the image – and that following the rules for image composition like the

rule of thirds common static region scheme with rectangles of equal size will be likely to split up important objects not considerably better. Of course, any impact of this aspect depends on the particular images in the collection. But at least, scaling the image as a whole and rotations by 90°can be handled as easily as with rectangular regions.

To support also rotations that are not multiple of 90°, ARP [Chalechale et al., 2004] cuts images into angular and radial partitions. This is illustrated in Figure (g) and (h). With 8 angular partitions, rotations by 45°can be achieved / compensated by permutations of the regions. More angular partitions would allow for even smaller degrees of rotation. If features derived from the partitions are stored in a feature vector of appropriate layout, applying one dimensional Fast Fourier Transform (FFT) can be applied to make the feature rotation invariant [Chalechale et al., 2005], thus avoiding the necessity to compute permutations for robustness against rotation.[4] Like in the case of the five fuzzy regions, the regions in ARP towards the center cover a smaller area of pixels than the ones more outside. Therefore the individual pixels within the center regions will have more impact if each feature derived from each region is considered with the same weights. Different weights can usually be incorporated at the time of query execution. This can be the case to implement differentiation based on the users' input whether some areas should be considered more relevant than other – or even irrelevant or unwanted as described in Chapter 2.3.3 on page 53.

If the regions are very small, this can be used to extract features at points without particular "saliency", but rather sampled from a regular grid or even chosen randomly. This allows to use "keypoint descriptors" even when no good detector for prominent points in the image exists – and in combination with machine learning approaches, then use only features derived from these sampling points which were able to perform well on the training data. For instance, [van de Sande et al., 2010] compares not only the results of different key point descriptors, but also the compares results of key point detectors with strategies like dense sampling.

5.1.4 Global Features

If the image is considered and represented as a whole in a feature, this is called a "global" feature. This has two major advantages:

1. It is the most simple case to implement.

2. With respect to the supported invariances, if the features derived on a global scale have no inherent limitations, global features will always be invariant to rotation (as long as not too many new pixels appear or disappear as mentioned in Chapter 2.3.3 on page 57) and changes of the scale of the image as a whole.

[4] Another approach basically exploiting the definition of invariance as presented in Equation (5.2) is proposed to create *invariant features* in [Siggelkow et al., 2001, Siggelkow, 2002] by integrating over a compact transformation group (a.k.a. Haar integral); to achieve rotation and translation invariance, integrating over the group of translation and rotation. *Fuzzy Histograms* are then used to build feature vectors with continuous bin assignments [Siggelkow and Burkhardt, 2002, Siggelkow, 2002]. As integration in practice is performed over each pixel in the image grid and normalized by the number of pixels in the image, this approach uses the most fine-grained static segmentation possible as an intermediate step to derive a global image feature.

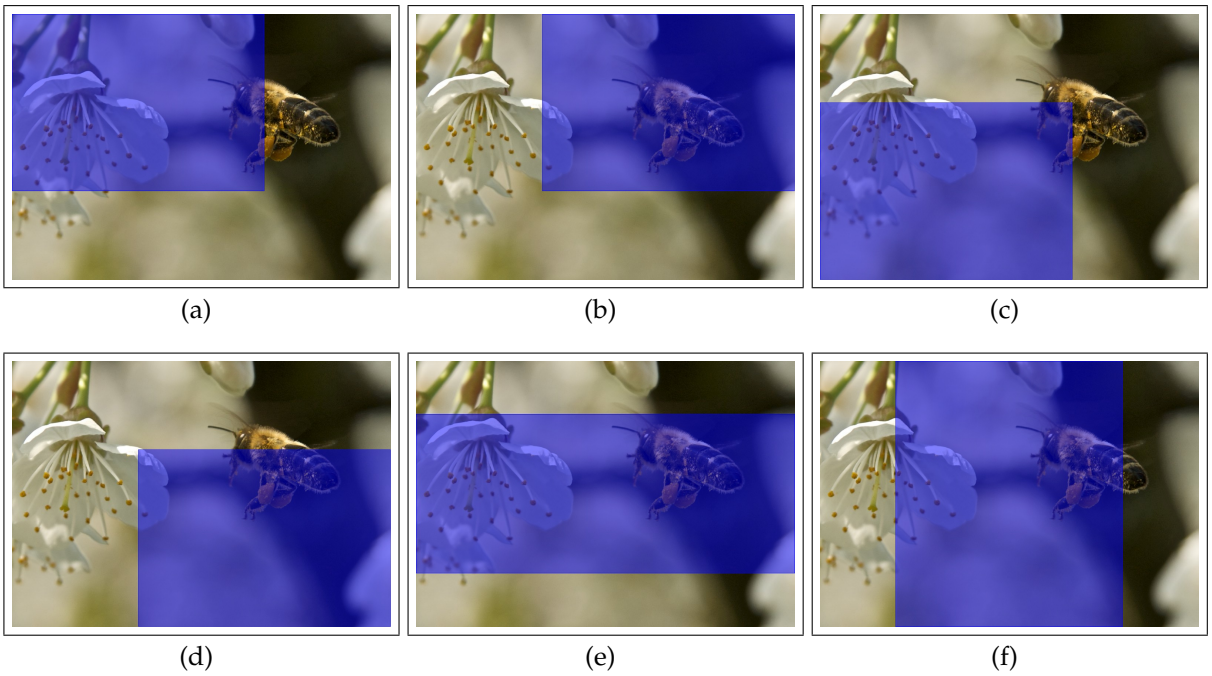


Figure 5.6: Static sub-images (highlighted in blue) as a heuristic approach to add invariance. The sub-images (a)-(d) can be used to compute global features or features based on static regions that handle to a certain amount translation. The sub-images in (e) and (f) add situations in which in which the aspect ratio is not preserved. Although there is no general guarantee that these regions are particularly important and may not miss important parts, they have been helpful in the context of Query by Sketching.

However, invariance to translation are not supported and changes in scale caused by zooming (and therefore adding or removing some information) cannot be handled either. Additional, in contrast to keypoints or static regions, invariance to rotation is not optional – it comes at the cost of limiting expressivity (as described in Chapter 2.3.4 on page 63).

To add optional support to translation, a heuristics can be applied either to the global image or also combined with static regions: By defining sub-images that cover some parts of the image and deriving the features from these, overlapping rectangular regions, some additional invariance can be provided at the time of executing the query by comparing the input to any of the sub-images and choosing the score based on the best match as we have used in [Springmann et al., 2010c, Springmann et al., 2010a].

Figure 5.6 shows six examples of such static sub-images. They have been used together with one additional, dynamic region in the context of query by sketching. The dynamic region is based on a bounding box that covers all non-empty areas in the sketch drawn by the user as well as the images in the collection that have been processed to contain only edges. This processing is performed with an edge detection filter as will be described in more details in Section 5.2.3 on page 120 and illustrated in Figure 5.9. Figure 5.9(i) shows the bounding box. This does assume that blank areas in the outer part of the sketch shall be considered as unknown rather than empty (cf. Chapter 2.3.3 on

page 53). We conducted a small evaluation with 54 sketches collected from seven users on a Tablet PC to search for known images used this scheme for providing (partial) scale and translation invariance [Springmann et al., 2010c, Springmann et al., 2010a]. It showed that depending on the target image and the user sketch, this simple heuristics improved the retrieval quality for up to more than half of all sketches provided by the users for one image.

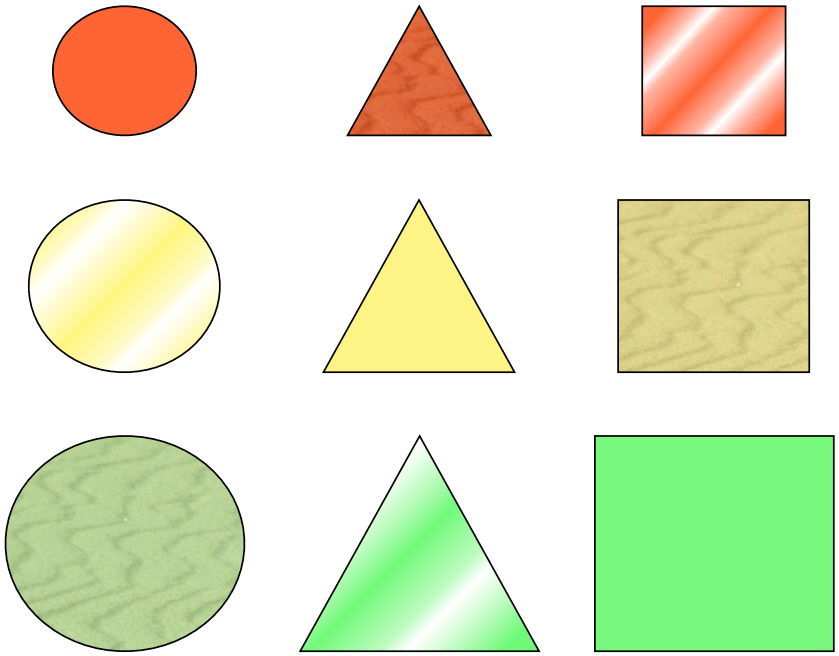


Figure 5.7: Examples of Color, Texture, and Shape: Each of the three lines contains three shapes of similar color. Throughout each column, the shape is the same - just slightly different in size. Along the mid-diagonal, the images have similar texture of plain color, some "waves", or color gradient. (Illustration inspired by [Dimai, 1999b, p. 13] and [Popper, 2002, p. 375].)

5.2 Perceptual Features

Features are commonly grouped by how they measure or correspond to human perception [Jörgensen, 2003, pp. 14–18,146–156].[5] Therefore they are grouped into *Color*, *Texture*, and *Shape*.

Figure 5.7 illustrates different colors, textures and shapes.[6] [Popper, 1959, p. 412] states:

> Generally, similarity, and with it repetition, always presuppose the adoption of a *point of view*: some similarities or repetitions will strike us if we are interested in one problem, and others if we are interested in another problem.

Therefore depending on the *point of view* or problem/task the user is working on, different objects in Figure 5.7 will be considered similar by the user. When focusing on

[5] See also [Smeulders et al., 2000, pp. 6–8] or [Li et al., 2002, pp. 263–267].

[6] The illustration is heavily inspired by an illustration in [Dimai, 1999b, p. 13] of 12 objects with no emphasize on color – for which credits were given to [Popper, 1959]. [Dimai, 1999b, p. 14] further states that "Popper uses this diagram to illustrate that terms such as 'likeness' and 'similarity' are without value unless someone explains in what respect two things are supposed to be compared". In fact, [Dimai, 1999b, p. 13] uses a slight variation of the original diagram from [Popper, 1959, p. 412] which had two rows of same size and different color and two rows with same color but different sizes (and provides a second example diagram) – while in [Dimai, 1999b, p. 14] every row has a different shading and size.

color, objects in same rows will be considered most similar; when shape is important it will be the columns; and in case of texture, it will be yet another scheme following the diagonal. Conceptually it is therefore important to support features in a way that is generic enough to replace features as needed to adapt systems to individual applications, for which there might be a need for a particular feature or combinations of features. In addition, much progress has been achieved and is still achieved in developing new feature descriptors that can achieve better retrieval results – therefore systems should allow to replace features with novel, better features to remain in line with state of the art approaches.

Before discussing aspects of particular features, an abstract notation is as follows: $\phi(\mathfrak{I})$ is a function to extract a feature from an image $\mathfrak{I} \in \mathcal{I}$, therefore transforms an element form the image space into the feature space:

$$\phi : \mathcal{I} \to \mathcal{F} \tag{5.1}$$

Such a feature space can basically anything, however a very important one will be the n-dimensional space of real numbers to define feature vectors in $\mathbb{R}^n$.

For instance, if ϕ would be GRAY-HIST256 that is building a histogram of every possible gray–value in 8-bit depth – so counting each pixel occurrence of a pixel with values from $0, 1, 2, \ldots, 255$ – will result in a 256-dimensional feature vector $\vec{v}$. In this case, the values at the individual positions in the vector will only be natural numbers (including zero), $\vec{v}_{1\ldots256} \in \mathbb{N}_0$ and $\sum_{i=1}^{256} v_i$ equals the number of pixels in the image. Since $\mathbb{N}_0 \in \mathbb{R}$ the entire vector belongs to the previously mentioned set of n-dimensional real numbers, or more precisely: $\vec{v} \in \mathbb{R}^{256}$.

If a feature is completely invariant to some image transformation t, the following equation must be satisfied:[7]

$$\phi(\mathfrak{I}) = \phi(t(\mathfrak{I})) \tag{5.2}$$

For instance, the GRAY-HIST256 takes into account only the gray-level value of individual pixels and does not rely on any position of a pixel, hence GRAY-HIST256 would be invariant to rotations by 90°.[8]

The example of the GRAY-HIST256 probably already shows that the grouping into *Color*, *Texture*, and *Shape features* is not always easy or completely possible as derived features from pixel information may not exclusively capture information of a single category.[9] A global GRAY-HIST256 certainly does not capture any shape information. It does capture information about the brightness of pixels in the image, in particular when

[7] Cf. [Smeulders et al., 2000, p. 1354]

[8] Which can be easily performed without having to interpolate any pixels. Any interpolation could slightly change the value of individual pixels and as a result, if transformation t causes interpolation, Equation (5.2) might no longer be satisfied, only

$$\phi(\mathfrak{I}) \sim \phi(t(\mathfrak{I}))$$

– which is less strict, but still able to fulfill the general assumption of Equation (2.11) on page 48.

[9] Cf. for instance the "Color and Edge Directivity Descriptor (CEDD)" [Chatzichristofis and Boutalis, 2008] and "Fuzzy color and texture histogram (FCTH)" [Chatzichristofis and Boutalis, 2010] intentionally contain both, color and texture information for content-based image retrieval.

for color images, gray values are extracted from the brightness channel in HSB/HSV color model or luminance channel in CIE L*a*b*/L*u*v* color space is used. But it does not capture other important properties of color (which would be expressed in the other channels and are commonly much more associated with the term "color"). And it does capture as well part of the information how homogenous the distribution is — an aspect which is commonly attributed to texture features. If we switch also from a global histogram to fine-grained (static) segmentation, the feature will not remain completely independent of the shape.

Notice that GRAY-HIST256 has been selected as an example feature because of its simplicity, not because of the quality of the results; much better features have been proposed in literature and many of them will be mentioned later inside this chapter. However, for many of the features found in literature –in particular the more recent approaches– the main goal is to achieve good retrieval results for the intended usage scenario and not to correspond exclusively to a single aspect of human perception. Therefore any attempt to group a feature into either *Color*, *Texture*, or *Shape* may not always be possible or ideal. Nevertheless it is common practice and is a helpful starting point to identify alternative features and differences between features.

With the availability of many features to choose from and situations in which a single feature may not satisfy completely the user requirements, Φ_{MF} will denote a method that can extract several features and therefore perform the transformation into the power set $\mathcal{P}(\mathcal{F})$:[10]

$$\Phi_{MF} : \mathcal{I} \rightarrow \mathcal{P}(\mathcal{F}) \tag{5.3}$$

In some cases there might also be more than one image that should be transferred into feature space, e.g., if more than a single image is included in query like $\mathcal{Q} = \{\mathfrak{Q}_1, \mathfrak{Q}_2, \ldots, \mathfrak{Q}_n\}$ with $\mathfrak{Q}_1, \mathfrak{Q}_2, \ldots, \mathfrak{Q}_n \in \mathcal{I}$, then Φ will denote a method that can perform the transformation for each image, hence mapping to $\mathcal{P}(\mathcal{F})$ like Φ_{MF} did for a single image:

$$\Phi : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{F}) \tag{5.4}$$

If there is a guarantee that each image results in exactly one feature and therefore $|\Phi(\mathcal{Q})| = |\mathcal{Q}|$, this can be indicated with Φ_{MO}.[11]

Semantic Gap

Any feature that is extracted automatically by a machine from a picture by processing its pixels[12] or even by human beings with different understanding of the scene will lead to some information from the image not being available for retrieval. This is due to the

[10] MF is simply used as an abbreviation for *multiple features*.

[11] MO is simply used as an abbreviation for *multiple objects*, but single features. Therefore Φ without any index represents the generic form of a Multi–Object Multi–Feature transformation (cf. [Böhm et al., 2001a] and [Weber, 2001, pp. 15,28]).

[12] Such features for which the approaches operate on the pixels directly are frequently referred to as *low-level features*, for instance in [Jörgensen, 2003, p. 146].

fact that as different interpretations of the same image may occur from different people and/or at different points in time: [Smeulders et al., 2000, p. 1353]

> The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.

For features which are derived from the image as whole rather or static regions any correlation to the image content, this gap can even widen compared to features derived only from particular objects in the image as the latter are at least able to aggregate only visual data that corresponds to (real world) objects into the numeric feature rather than pixels which potentially have to be considered unrelated on the semantic level.[13]

General Correspondence to Regions

Since features can correspond only to some region of the image, there can be several ways to incorporate region information:

- *Segmentation/Keypoint detection as Preprocessing*: Any image can be preprocessed to create new (sub-)images. By extracting features from every sub-image independently, a set of features can be generated from a single image. For keypoint-based approaches, this is very common and therefore they are frequently described in two stages: *keypoint detection* to identify the point and surrounding region, and *keypoint descriptor* which forms the actual feature extracted for each keypoint. In this case, Φ would emit one feature per region/keypoint for which usually the information is stored to which region/keypoint it belongs. The latter can be used to apply spatial constraints in search.

- *Blocks/Static Regions as core part of the feature*: Many features have been enriched by taking into account the spatial distribution of values. This can be achieved in many cases by extracting a simple features from static regions and concatenating the resulting feature vectors into one single, long vector. In this case, Φ would emit one feature per image which in itself carries the correspondence to regions. Spatial constraints can be applied if this internal structure is respected.

[13] The *Semantic Gap* has frequently been mentioned as a reason why CBIR systems cannot deliver answers similar to what would be expected from manually generated / curated lists – and sometimes probably even been used to excuse disappointing results. As the gap origins from the problem that the information that can be derived just from the visual data is not sufficient for understanding the situation completely, additional sources of knowledge are needed (cf. [Smeulders et al., 2000, p. 1375]) – either at query time or when training a system to learn at least some basic concepts which have a visual correspondence. The latter is frequently performed with keypoint descriptors to restrict the use to descriptors that can be found in several instances of a class and are discriminative between several classes, thus can be used as a "visual words" or "visual codebook", e.g., in [Sivic and Zisserman, 2003, Agarwal et al., 2004, Bosch et al., 2007, Rabinovich et al., 2007, Yang et al., 2007, Opelt et al., 2008, Deselaers et al., 2008b, Yang et al., 2008, Tirilly et al., 2010, Kogler and Lux, 2010, Gavves and Snoek, 2010, Tian et al., 2011, Eitz et al., 2011a]. There has been some debate whether or not *ontologies* and the *semantic web* can help to narrow the semantic gap (cf. [Santini, 2008, Santini and Gupta, 2010]).

- *Global Features*: For global features it is not necessary to consider regions. However, if needed, both aforementioned strategies can be applied to construct a feature / region combination.

Thus, treating Φ as the generic base case provides most flexibility whether multiple individual features are emitted for reasons of regions or not.

In general, any feature could be combined with any scheme of regions. In practice, many features have been proposed in combination with some particular scheme and may even not work with others. The latter is mainly true for shape features based on the object contour which commonly do require appropriate segmentation. The following subsections 5.2.1 – 5.2.3 will show some examples of applied techniques and combinations with regions. They therefore review mainly the state-of-the art of perceptual features used in content-based image retrieval.[14]

5.2.1 Color

Color is considered important as both, a visual and semantic retrieval tool: Common objects that are similar often have similar color and in addition, overall composition can indicate such semantic factors as whether an image depicts an indoor or outdoor scene [Jörgensen, 2003, p. 146]. A simple approach to measure what colors are present in an image and to which extent, is to count the pixels of individual colors to compute color histograms – just as described for gray values on page 111 and not uncommon also for image processing operations, e.g., to adjust color/brightness levels. However, in a common RGB color image with 8 bit depth for each of the channels red, green, and blue, this leads to $256 \times 256 \times 256 = 16.7\ (= 2^{24})$ million unique color values. This means that images would need a resolution of approximately 5000×3400 pixels if a single pixel of each unique color shall be present; for significantly smaller resolutions, e.g., of webcams or common sizes used to illustrate web pages, for most colors there would not be single pixel in the entire image.[15] In image processing, it is common to either consider just the (computed) luminance channel to represent the brightness or on the individual R, G, B channels in isolation to still have a meaningful histogram. For analyzing the overall color impression of an image, this seems not ideal. Therefore the easiest alternative approach would be to reduce the color resolution by quantizing the found colors into

[14] Impatient readers may therefore skip to Section 5.2.4 on page 123 directly. The list of approaches is certainly far from exhaustive and as mentioned in Section 5.2 on page 111, the grouping into Color, Texture, and Shape may not always be ideal.

[15] Modern digital cameras have reached resolutions of 16 megapixels (MP) and above between the years 2008 and 2011. For instance professional full-frame sensor DSLR have been announced in that time the like Canon 5D Mark II with 21.1 MP in Sep. 2008 and a little more affordable cameras with sensors in APCS-C size like Canon 7D/550D in Sep. 2009/Feb. 2010. In 2011, Casio, Fujifilm, Sony, Panasonic, Samsung announced also a number of compact cameras from the Exilim / FinePix / Cybershot / Lumix DMC-F/ ST/PL line with 16.0 - 16.2 MP.
However, 1. by stitching together several digital images or using good scanners to digitize prints, even higher resolution images have been technically possible and certainly in practical use for satellite images since years and 2. most of the cameras mentioned do not only support JPEG as an image file format, but RAW formats with 12 or 14 bits per color channel. As a consequence, such images will have 2^{3*12} to 2^{3*14} unique colors – and a full histogram will again consist mainly of zeros and ones when pixels colors would follow a uniform distribution.

bins. [Flickner et al., 1995, p. 26] uses a color histogram with 256 bins in total (which can be represented with a total of 8 bits instead to the previously mentioned 8 bit per channel). Color histograms from images with same overall resolution are invariant to translation and rotation about the viewing axis and change only slowly under change of angle of view, change in scale in the scene, and occlusion [Swain and Ballard, 1991]. The approach was successfully used for identifying objects in images that have characteristic colors. [Stricker and Swain, 1994] estimated that a color histogram with 64 bins could store more than 25'000 distinguishable color histograms. They are still sensitive to changes in illumination, which was addressed in [Funt and Finlayson, 1995] by using histograms of color ratios of neighboring pixels rather than absolute color values.

Color Coherence Vectors (CCV) is a histogram-based method that incorporates spatial information by classifying each pixel in a given bin as either coherent or incoherent, based on whether or not it is part of a large similarly colored region [Pass et al., 1996]. For this, the image preprocessed with a 3×3 mean filter to blur and colors discretized to 64 values. In this reduced color space, the image is essentially segmented by searching for connected regions. For each pixel, it is determined whether or not it belongs to a region of the same color bin and whether this region exceeds a threshold on the number of connected pixels – if so, the pixel is considered coherent. For each color bin, a coherence pair is stored: the total number of coherent and incoherent pixels. Thus, the final vector is not using the segments that have been computed as an intermediate step. In contrast, Blobworld [Carson et al., 2002] uses Expectation-Maximization (EM) for segmenting the image. Each segmented region is described by a color histogram in L*a*b color space with 5, 10, and 10 bins for the L, a, and b channel; therefore the regions are preserved and available for retrieval.

Color moments [Stricker and Orengo, 1995] were proposed to have a more compact representation for the color distribution, e.g., for the first 3 moments (mean, variance, covariance) for an image in HSV color space only 9 floating point numbers per image, and therefore achieve faster retrieval times while at the same time producing better results as it avoids a number of problems where color histograms do not match human perception. [Stricker and Dimai, 1996] proposed to use five fuzzy regions as described on 105 and illustrated in Figure 5.5(e) and (f) to support for spatial distribution in retrieval. CCV and Color moments have also been successfully used for classification of images to semantic classes like indoor/outdoor, night/day, sunset in [Vailaya et al., 2001, Ciocca et al., 2011], color moments with 10×10 / $9 \times$ respectively sub-blocks in *LUV* color space. [Vu et al., 2003] basically extends color moments for the use in searches with a particular region-of-interest (ROI) by computing a signature of seven statistical average-variance pairs from sampled blocks in 85 sub images of each image in the database and evaluating the distance only of blocks that lie inside the ROI, treating all blocks outside the ROI as irrelevant (cf. Chapter 2.3.3 on page 53).

The MPEG7 standard [Chang et al., 2001] introduces a number of audio and visual descriptors. The visual descriptors use a variety from global, static, and segmented regions. The Scalable Color Descriptor (SCD) uses a global histogram in HSV color space for which a Haar transform is used for controlling the color quantization [Manjunath et al., 2001, pp. 706ff]. Color Space Descriptor (CSD) adds spatial information by using a 8×8 pixel structure element to compute on the (sub-sampled)

image at how many positions of the structure element a value with quantization of 184, 120, 64, and 32 bins in HMMV color space is found [Manjunath et al., 2001, pp. 708f]. The structuring element is moved over the image, resulting in 64 sample points. Dominant Color Descriptor clusters regions and determines up to eight dominant colors for each region, including the percentage of region having this color, coherency, and (optionally) the color variance [Manjunath et al., 2001, pp. 709f]. The Color Layout Descriptor (CLD) [Manjunath et al., 2001, pp. 710f] derives the average color for the image divided into 8×8 blocks as depicted in Figure 5.5(c)-(d) and quantized using a DCT in YCrCb color space [Manjunath et al., 2001, pp. 710f]. As CLD captures the spatial distribution of color, it can be as an descriptor for sketch-based image retrieval where the sketch mainly consist of defining the color of particular areas. It has been applied in [Westman et al., 2008] for sketch-based retrieval, but results have not been satisfactory. Another descriptor using even a more coarse split into 4×4 static regions as in Figure 5.5(a)-(b) is proposed in [Chatzichristofis et al., 2010]. For each region, a histogram for a custom palette of 8 colors in HSV color space is derived based on fuzzy membership function. The resulting feature vector for the 16 regions $\times$ 8 colors gets quantized with 3 bits per bin into the Spatial Color Distribution Descriptor (SpCD), thus forming a compact descriptor of 48 bytes that has been evaluated with image retrieval based on hand-drawn sketches. A different approach for a similar setting has been proposed in [Jacobs et al., 1995], which uses multiresolution wavelet decompositions of the color channels for sketch-based retrieval of images at a resolution of 128×128 pixels.

[Gouet and Boujemaa, 2001] proposes to use interest points found with a Harris color points extractor to generate color differential invariants. Affinely invariant regions [Van Gool et al., 2001] uses Generalized Color Moments that are detected either with geometry-based regions using corner points and edges, or intensity-based regions starting from local extrema. Due to the popularity of SIFT [Lowe, 2004], a significant number of variants that do not only use intensities but color information has been proposed. Many of these have been evaluated in [van de Sande et al., 2008, van de Sande et al., 2010].

5.2.2 Texture

Texture is an ubiquitous attribute of visual objects referring to the statistical, or syntactic, property of surface features which confer visual homogeneity [Jörgensen, 2003, pp. 148f]. [Tamura et al., 1978] identified and confirmed in psychological experiments six textural features that correspond to visual perception: coarseness, contrast, directionality, line-likeness, regularity, and roughness. Each of this texture feature can be seen as a axis with two extreme poles and in their proposed descriptor therefore represented by a single numeric value to identify where an image would be located on the between the two poles. Using and refining methods to compute these values from the (gray-level) information of images, three of them (coarseness, contrast, and directionality) have shown successful results and strong significance; while still simple combinations of the features were not able to simulate human similarity measurements. Within the MPEG-7 standard, the Texture Browsing Descriptor provides a compressed/quan-

tized and therefore even more compact descriptor of at most 12 bits to characterize a texture regularity, directionality, and coarseness [Manjunath et al., 2001, pp. 711f].

In [Manjunath and Ma, 1996], Gabor wavelet-based texture analysis are proposed with a number of different orientations and scales to basically detect edges in frequency space. The response to Gabor filters per scale and orientation is analyzed for the statistical moments mean and variance, therefore measuring high strong the image contains repeated edge patterns in a particular direction (orientation of filter) and frequency (scale of filter). For the proposed four scales and six orientations, this results in a $4 \times 6 \times 2 = 48$ dimensional feature descriptor. Due to the orientation of the filter, these texture filters are sensitive to rotation of the image. [Dimai, 1999c] therefore proposed a approach based on Gabor filters which is rotation invariant through the use of General Moment Invariants. Gabor filters have been used in [Weber et al., 1999, Mlivoncic et al., 2004a, Brettlecker et al., 2007] not only as a global image descriptor, but also in static regions like those depicted in Figure 5.5. With particular emphasize on scene recognition, global features are generated by determining the principal components of the response to multi scale oriented filters on the luminance channel to capture the *gist* of a scene [Oliva and Torralba, 2006].

In addition to the Texture Browsing Descriptor, the MPEG-7 standard also defines the Homogeneous Texture Descriptor (HTD) [Manjunath et al., 2001, pp. 712f], which is also using Gabor filters to derive a global feature and the Edge Histogram Descriptor (EHD) [Park et al., 2000, Manjunath et al., 2001, pp. 713f]. For the latter the image is divided in 4×4 sub-images as in Figure 5.5(a) and (b). Each of these 16 sub-images are essentially rescaled to achieve a constant number of blocks per sub-image. A block consists of 2×2 (rescaled) pixels. For the each of 16 sub-images a local edge histogram is computed by counting every block of whether it contains edges from one of the five categories: horizontal, vertical, 45 degree diagonal, 135 degree diagonal, undirected – which results in $16 \times 5 = 80$ bins, which are nonuniformly quantized using 3 bits/bin, resulting in a descriptor of size 240 bits. An extension derives from the 16 sub-images also a global and 13 semi-global histograms, thus leading to 150 instead of 80 bins in total, which has shown to improve the retrieval quality. [Pinheiro, 2007] proposed the Edge Pixel Direction Histogram (EPDH), which is based edge directionality with histograms also computed from 16 sub-images (4×4 as in EHD), but varying the scale-space for edge pixels, uses only 4 directions (no undirected edges), and adds global histogram only for higher scale. [Jain and Vailaya, 1996] showed the applicability of edge direction histograms and edge direction coherence vectors to classify some semantically meaningful classes. To be able to use query by sketching, it is very difficult to use texture directly – therefore [Pala and Santini, 1999, p. 518] propose in their approach to allow the user to select the texture from different images and use it in their query. Other approaches use edge histograms where the edges in the sketch are taken directly, while they are derived in the case of the images in the collection. Under this variation, EHD can be used for sketch-based retrieval. [Eitz et al., 2010] proposes an histogram of oriented gradients (HOG) descriptor which basically extends EHD by storing the sum of squared gradient magnitudes falling into one of six discrete orientation bins while still maintaining the same regular grid for defining static regions / sub-images. In this eval-

uation, HOG has significantly outperformed the EHD as a texture-based descriptor for sketch-based image retrieval.

Texture information has also been used for segmenting images, for instance, the Windsurf system [Ardizzoni et al., 1999] uses wavelet features and Edge-Flow [Ma and Manjunath, 2000] uses Gabor filters together with Gaussian derivatives. From and for such segmented regions, it can be beneficial to construct texture thesaurus for efficient lookup based on visual codewords, such as applied in [Ma and Manjunath, 1998] for browsing and finding areas in large-scale aerial photographs. Classifying images and the objects depicted in them can also be based on texture information, in particular in images that do not contain color . Local Binary Patterns (LBP) [Ojala et al., 2002] use the neighborhood of individual points to provide invariant to changes in gray-scale and rotation. For this, a binary code is generated that describes the local texture pattern in a circular region thresholding each neighborhood on the circle by the gray value of its center. After choosing the dimension of the radius and the number of points to be considered on each circle, the images are scanned with the LBP operator pixel by pixel and the outputs are accumulated into a discrete histogram. LBP have been successfully used in video surveillance [Zhang et al., 2007] and for medical image classification [Unay et al., 2007, Oliver et al., 2007, Tommasi et al., 2008b].

Many descriptors used with salient points are based on the homogeneity around the detected keypoints. Scale-invariant Feature Transform (SIFT) keypoints [Lowe, 1999, Lowe, 2004] uses Differences of Gaussian (DoG) for detecting keypoint locations and generates descriptors from 4×4 local gradient histograms with 8 bins, thus leading to a 128-dimensional feature vector per keypoint. Gradient location and orientation histogram (GLOH) [Mikolajczyk and Schmid, 2005] is a descriptor which extends SIFT by changing the location grid and –similar to [Ke and Sukthankar, 2004]– uses PCA to reduce the dimensionality and therefore the size of the descriptor. [Mikolajczyk and Schmid, 2001] generates 12 dimensional descriptors for scale-invariant interest points by computing Gaussian derivatives at the characteristic scale detected with a Harris-Laplacian detector. The peak in a histogram of local gradient orientations is used to control the direction for the computation of the derivatives to achieve invariance to rotation. Speeded-Up Robust Features (SURF) [Bay et al., 2006, Bay et al., 2008] uses a Hessian matrix to detect interest points much faster than DoG or Harris-Laplacian detectors and construct a descriptor from sums of Haar wavelet responses in horizontal and vertical direction. Descriptors like DAISY [Tola et al., 2010], Binary Robust Independent Elementary Features (BRIEF) [Calonder et al., 2010], and Binary Robust Invariant Scalable Keypoints (BRISK) [Leutenegger et al., 2011] are optimized to reduce the computational complexity during extraction of keypoint descriptors.

5.2.3 Shape

Shapes can be detected and represented in two basic ways (cf. [Jörgensen, 2003, p. 150],[Safar and Shahabi, 2002, pp. 4–7],[Kimia, 2002, pp. 350–352]):

- Through shape *boundaries* which are found by edge detection or

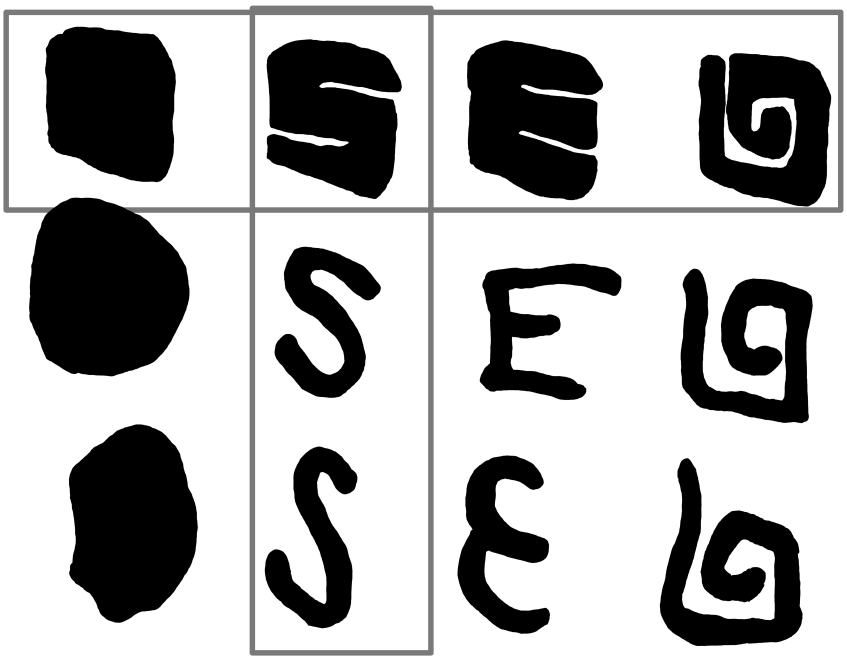


Figure 5.8: Retrieval of Shapes using Contour and Region (Illustration drawn to mimic illustration in [Bober, 2001, p. 716]): First row shows retrieval for region-based similarity, columns show retrieval for contour-based similarity.

- the region inside the shape (*interior representation / region-based methods*) which is frequently achieved with some region growing algorithm.

Figure 5.8 illustrates the difference between retrieval by region and contour.

When edges are used, in contrast to Texture feature which as we have seen also frequently process edges, less emphasize is given to homogeneity and more emphasize on how edges are connected to form shapes.[16] Shape information is frequently used when information is taken directly from the user (Query by Sketching) or due to their ability to match well the application domain like for trademarks and logos, medical retrieval, technical drawings / CAD, fingerprints in law enforcement, etc. (cf. [Kimia, 2002, pp. 346f]).

The QVE system [Hirata and Kato, 1992] uses a reduced resolution of the image and edge detection to generate the so-called abstract images in a size of 64×64 pixels that are compared to a rough sketch provided by the user by aligning blocks of 8×8 pixels

[16]The separation gets harder when regions are used in texture-based approaches: For instance, if edge directionality is captured like by the texture feature, e.g., Tamura features [Tamura et al., 1978], Gabor Texture Moments [Manjunath and Ma, 1996], or EHD [Park et al., 2000] and we extract the feature, e.g., with 8×8 non-overlapping regions as depicted in Figure 5.5(a), there are only limited choices of shapes that could satisfy generate similar texture features. If we do not consider only one type of non-overlapping regions, but add an additional layer with different partitioning scheme, e.g., 5 fuzzy regions as in Figure 5.5(e), there are even less choices of different shapes that could still produce features that are similar. Hence, Texture features with spatial information may capture also shape information. However, dedicated shape features usually are better suited to deal with invariances that are needed to support searches where shapes are deformed or misplaced.

within a range to provide limited invariance to translation. The score for ranking of the results is determined by computing the overall correlation between all blocks.

The QBIC system [Niblack et al., 1993, Flickner et al., 1995] uses 20 shape features of manually or semi-automatically identified objects like area, circularity, eccentricity, major axis orientation and algebraic moment invariants. The user can draw a binary silhouette image of the shape using a polygon drawing routine.

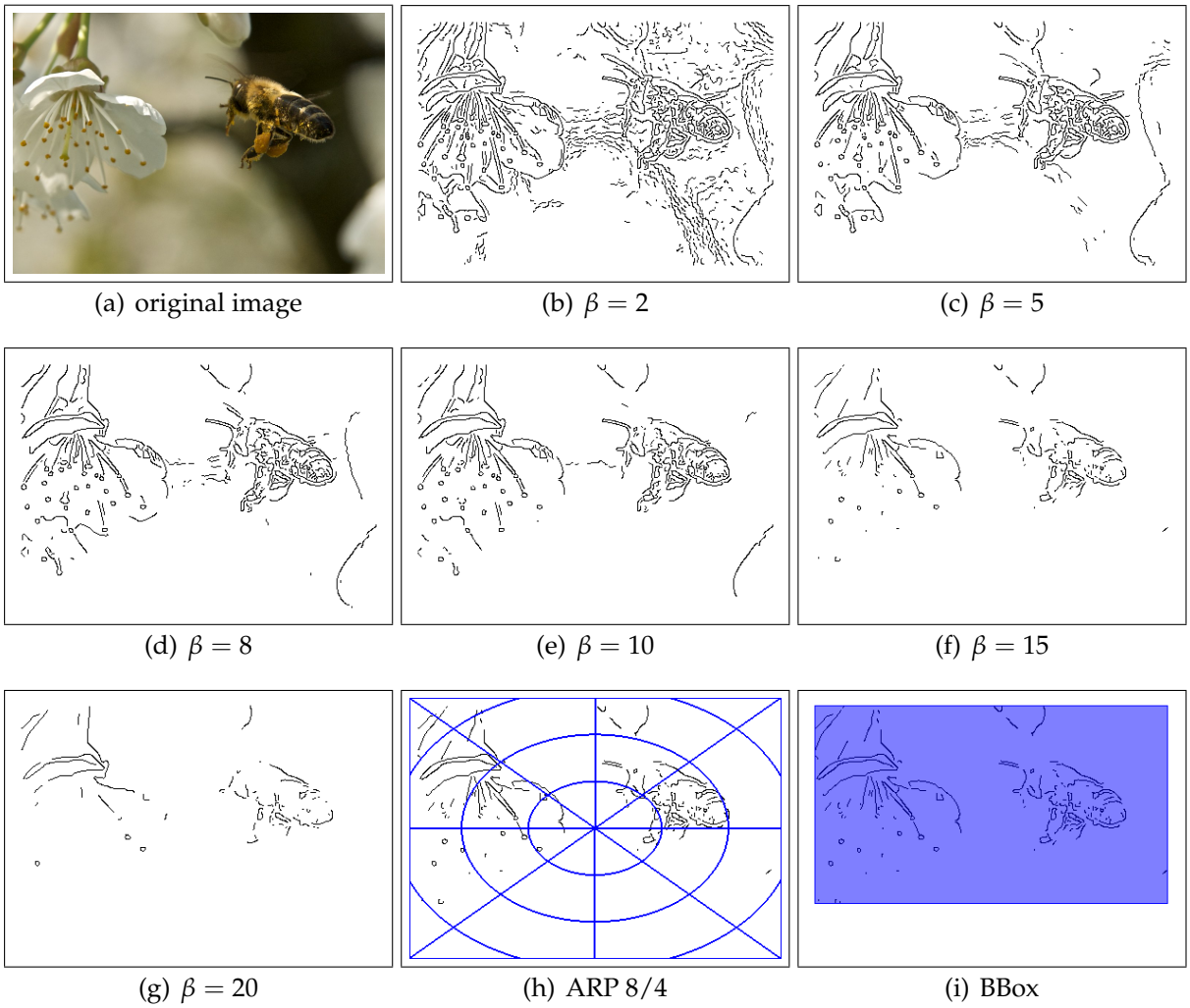
in [Del Bimbo and Pala, 1997], elastic matching of the sketch to the edges detected in images is performed by interpreting them as B-splines with 20 knots, not individual strokes or pixels. The approach is able to cope with differences in scale and claims to deal with small rotations (of the order of 12-15 degrees). It can also be used to find several objects in spatial relationship. To reduce computational complexity, filtering based on the aspect ratio of the object and the spatial relationship between objects is applied if more than one object is used. In [Berretti et al., 2000] this approach is extended to use tokens, such that also parts of an object can be retrieved.

Deformable contour models minimize the energy to deform some given contour shape to match an image. Splines in the Active Contour Models (a.k.a. Snakes) [Kass et al., 1988] and Level Sets [Malladi et al., 1995] are used in medical image segmentation [He et al., 2008]. Active Shape Models [Cootes et al., 1995] introduce point distribution models and learn the main modes of deformation with principal component analysis (PCA) from a training set. They have been used for object detection, classification, and image search. Active Appearance Models [Cootes et al., 1998, Cootes, 2010] considers also the gray intensities as image patterns around the shape, which has been applied in particular to the detection and identification of faces. [Jain and Vailaya, 1998] proposed the combination of a histogram of edge directionality with seven moment invariants and deformable shape templates for the use in Trademark retrieval.

Geometric moments have been proposed as region-based descriptors: Zernike moments in [Teague, 1980], General Fourier Descriptors (GFD) in [Zhang and Lu, 2002], and Invariant Zernike Moment Descriptors (IZMD) in [Li et al., 2009a]. Section coding [Berchtold et al., 1997, Berchtold and Keim, 1998] is another region-based descriptor that has been applied to CAD retrieval. Section coding uses a partitioning scheme similar to Figure 5.5(g) and constructs the descriptor from the relative area covered by the region in each partition. An extension into 3D section histograms has been applied to 3D protein databases [Ankerst et al., 1999].

In the context of MPEG-7, two different 2D shape descriptors have been proposed [Sikora, 2001, Bober, 2001]: For region-based description, Angular Radial Transform (ART) [Bober, 2001, p. 717] applies sinusoidal basis functions of the image and provides rotation invariance. For contour-based description, the image is smoothed gradually by repetitive application of a low-pass filter until the contour becomes convex. The Curvature Scale Space (CSS) [Bober, 2001, p. 718] represents the passes of this filter and peaks in it represent positions in the contour where concave and convex parts are separated. The prominent peaks are extracted as the descriptor for the shape.

Angular Radial Partitioning (ARP) proposed in [Chalechale et al., 2004] has shown better results in experiments for sketch-based image retrieval than Zernike moments, EHD, and ART while being also fast in extraction and search. The approach is based on



(a) original image (b) $\beta = 2$ (c) $\beta = 5$

(d) $\beta = 8$ (e) $\beta = 10$ (f) $\beta = 15$

(g) $\beta = 20$ (h) ARP 8/4 (i) BBox

Figure 5.9: An example image (a) and edge maps for ARP with various values of β (b)-(g). As the threshold β increases, more and more edges disappear. Partitioning is overlaid with 8 angular and 4 radial partitions (h) and a bounding box (i) on edge map with $\beta = 15$.

pure spatial distribution of edge pixels in the so-called edge maps (or pictoral index) in images normalized to a common size. Edge maps are generated through a variant of the Canny edge detector [Canny, 1986] with a single threshold β to control which information on luminance channel is important enough to be preserved. Figure 5.9 shows several edge maps processed with different values of β and the partitioning scheme like in Figure 5.5(g). Sketches are processed with skinning and noise reduction operators before comparing the sketches with the edge maps with regard to the number of edge pixels in each of the angular radial partitions. Rotation invariance can be achieved by applying a 1D FFT on the ARP feature vector. [Pinheiro, 2010] proposes an Angular Orientation Partition Edge Descriptor (AOP), in which not only the number of edge pixels are counted in each angular partition, but also their directionality. The latter is defined w.r.t. the radius (parallel and normal to radius), thus keeping it possible to create a

rotation invariant descriptor with a 1D FFT as in ARP which was not possible for edge-direction-based feature descriptors for texture like EHD and EPDH. A further extension in AOP aims at translation invariance by determining the gradient image mass center as the center for the angular division of the image instead of the image center as in ARP.

A different approach, much closer to the approach used in QVE, has originally been proposed for the recognition of handwritten characters: The image distortion model [Keysers et al., 2004, Keysers et al., 2007] uses a reduced resolution of the image, commonly scaled to 32 pixels in height while preserving the aspect ratio. Image deformations are allowed within the so-called warp range for individual pixels and the area around them (the so-called local context). Usually, the warp range is set to 2, resulting in a 5×5 area allowed for deformation and a local context of 1, resulting in a 3×3 area. It is either applied on the gray intensities of the pixels directly or on the edges identified with horizontal and vertical Sobel filter. Compared to QVE, it uses a sliding window instead of dividing the image into blocks and by default a much smaller resolution. This approach can also be used to compare also sketches with images by applying edge detection to the images. In an earlier work, [Pala and Santini, 1999], elastic deformations of images in gray scale has been performed as well as shapes compared with edges detected with a Canny edge detector. Another approach that uses images of a reduced size is proposed in [Torralba et al., 2008]: Images are scaled to 32×32 pixels while preserving the color. The similarity is computed by first minimizing globally the sum of squared differences for small translations, differences in scaling and image mirroring and then minimizing for shifts of individual pixels within a 5×5 area (the first is called D^2_{warp}, the second D^2_{shift} in [Torralba et al., 2008, p. 1963] – with D^2_{shift} being almost identical to IDM without a local context). The approach was evaluated for the tasks of person detection and localization, scene recognition, and automatic image annotation with the an emphasize on the role of the data set size as this increases the probability that among the images in the data set are at least some that are very similar to the query image. In [Torralba et al., 2008, p. 1968], the data set consists of 79 million 32×32 color images with (weak) text labels and an adapted Wordnet voting scheme is used to exploit semantic relationships between the text labels.

Dedicated for sketch-based retrieval [Eitz et al., 2009a, Eitz et al., 2010] proposes tensor descriptors derived for rectangular blocks in which the image (and sketch respectively) are subdivided. For comparing the distances, only blocks in which the user has drawn at least some edge pixels are considered.

Boundary detection tries to identify the contour of real objects instead of just edges that can also be located inside the object. [Martin et al., 2004] describes an approach to learn the optimal weights for combining several visual clues including brightness, color, and texture gradients. This approach was used in [Ferrari et al., 2010] to build shape models from training images that can be used to detect and localize instances of the same class in different images. The shape model are achieved through principal component analysis (PCA) to learn intra-class deformations – similar to [Cootes et al., 1995], but without the need for explicit point-to-point correspondence knowledge. Similarly [Schindler and Suter, 2008] solved this problem by performing (over-)segmentation of the image to derive edges that contain only closed contours. These contours are approximated by a closed polygon with a fixed num-

ber of equally spaced vertices to derive global shape models of the objects. Edgelet features are used for the detection of objects in [Wu and Nevatia, 2007] and humans in [Wu and Nevatia, 2005]. An edgelet is a short segment of line or curve that can be quantized in a few edge describing categories. It is well-suited for finding similar curves inside an image, therefore is invariant to translation. However, the edgelets are derived by methods of machine learning, therefore may require at least training to adapt to a particular problem domain. Similarly, the Boundary Fragment Model (BFM) [Opelt et al., 2006, Opelt et al., 2008] is trained to detect objects.

Maximally Stable Extremal Regions (MSER) identifies distinguished regions which have pixels that are either all brighter or darker than all pixels on the region's boundary for which an normalized ellipse is found and rotation invariant moments derived [Matas et al., 2004]. The Pyramid of Histograms of Oriented Graphs (PHOG) in [Bosch et al., 2007] captures the spatial layout of local image shape by employing multiple layers of increasingly finer grids to partition the image and the generated histograms. Interest point detectors have also been used in [Agarwal et al., 2004, Deselaers et al., 2005, Teynor et al., 2006] to identify keypoints from which patches are extracted to train classifiers for object detection, while intensity patches were used with randomized trees in [Lepetit and Fua, 2006] and randomized ferns in [Ozuysal et al., 2010].[17]

5.2.4 Relationship of Features with User's Task Input and Aim

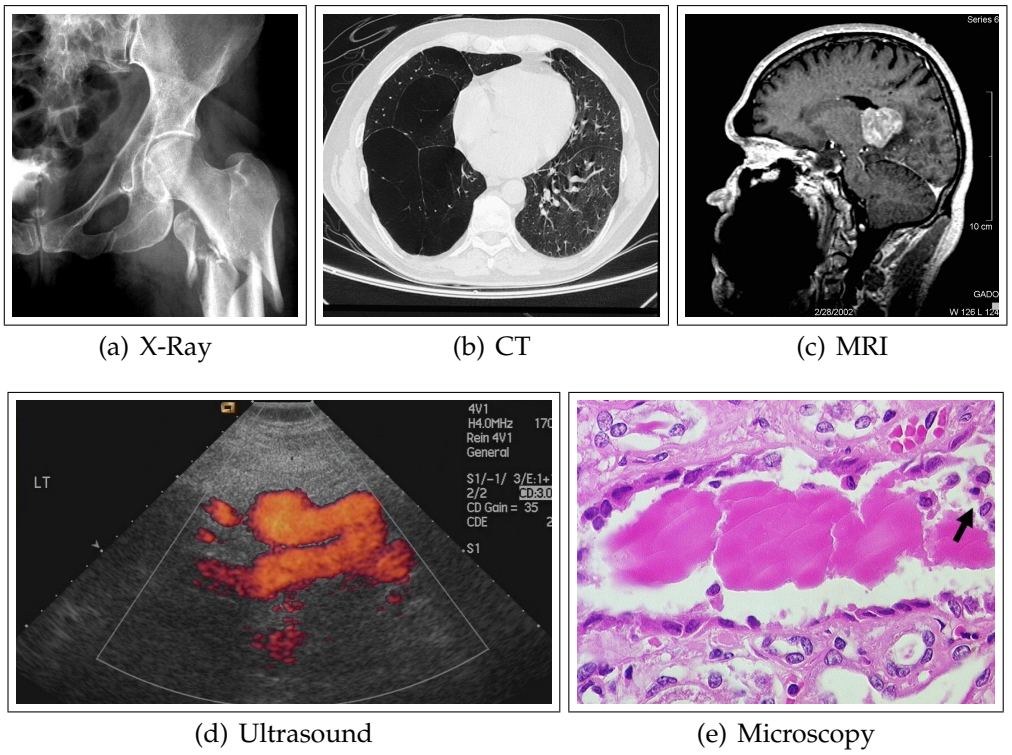
As we have seen, there is a very rich combination of features and regions that have been proposed in literature. Depending on task, at a different level of granularity features have to be used. The following list contains some very rough observations:

- To detect objects inside images no matter where, how big and at which angle they appear as needed in many *Object Detection* and *Object Identification* tasks, invariance to scale, translation and rotation are needed. These tasks will work best with "proper" *Segmentation* which separates the object completely from the environment. Such a segmentation is always hard to achieve. In particular when the object of interest in the input image provided by the user may be occluded, any approach relying on proper *Segmentation* is likely to fail.

  Sufficiently good results may be achievable with salient keypoints in combination appropriate training of the system on the *classes* that shall be detected: In a successful approach, the system "learns" only keypoints that are inside the object, therefore not affected by the environment, and convey enough information of both, the area of the objects and the variability inside the class.

- For *Classification* of an image as a whole, global features might be sufficient or even achieve better performance than local features as "overtraining" on the examples is a little less likely: Global features map more closely to the this particular

[17] As patches are small images by themselves, they carry all perceptual features of a full image. However, in the mentioned approaches, the patches are based on gray-level intensities, therefore do not contain color information anymore. The patches are not analyzed for visual homogeneity, therefore are not focussed on texture.



(a) X-Ray (b) CT (c) MRI

(d) Ultrasound (e) Microscopy

Figure 5.10: Medical images from different modalities: (a) shows a fracture of the femur bone using X-ray and therefore a projection of the 3D object into 2D. (b) shows the computerized tomography (CT) of the lung, (c) the magnetic resonance image (MRI) of the head – both always show only cross-sections (slices) of 3D objects. (d) shows a Doppler ultrasound image, which contains colors to indicate measurements instead of the visual appearance of the object itself. Such coloring is also common in functional MRI (fMRI). (e) shows the microscopic image of kidney tissue. (All images are taken from the query subjects used in the *ImageCLEF 2005 Medical Image Retrieval* benchmark task [Clough et al., 2005, Hersh et al., 2006].)

task aim and by using the the entire image, reduce the degree of freedom that would otherwise increase the danger of overtraining. For instance, for identifying the modality used in medical imaging like X-ray, CT, or MRI as shown in Figure 5.10(a)–(c), global features may work very well. However, for detecting individual deformations in the medical image, in particular those that correspond to symptoms of a disease, they are frequently not fine grained enough. When the medical use-case depends on individual parts of the image –not the image as a whole– similar observations as for *Object Detection/Identification* may apply; however, since many modalities generate images only with gray-level intensity information and no color, additional emphasize is put on texture and shape features. The assumptions on what regions and points may be interesting (static region in the center of an image as in 5 fuzzy regions; interest points / keypoints located at salient parts in the image) may not hold for this domain, e.g., because the image composition may not be the choice of the "photographer", but defined by the

modality and how the subject/patient can be placed for taking the image. In Figure 5.10(e) this is very visible as an arrow was included in the image to point out, where the region of interest is located.

- For finding *Known Images*, it might help to be able to identify individual image component – but this may depend on the used strategy. In many cases, users will have at least a rough memory on where the objects in the desired image were placed and full scale, translation and rotation invariance won't be needed and may even reduce expressiveness in queries. For that, some static image regions can work well directly, while for segments and keypoint, additional spatial constraints might be needed to restrict the effect of the invariances. When the user input can only be based on the memory of the image, it is frequently very hard for the user to remember colors precisely and will be impossible to reproduce in detail any color distribution or texture. Therefore letting the user sketch the image and using shape features may be of greater help than in other tasks.

- In *Themed Searches*, the "mood" of an image may correspond to the entire appearance of the image – like light or dark colors being predominant.

- Section 5.2 is restricted just to perceptual features – it does not discuss other information that can be used for retrieval (and certainly should used when appropriate), e.g., manual annotations that can be assigned to an image as a whole or just segments / manually defined regions.

Matching Tolerance is one aspect of great importance for Query Formulation and Execution. It does not only depend on the feature that is being used, but also on the Distance Measure to compare features with each other.

5.3 Similarity and Distance Measures

The basic assumption in similarity search is, that the more similar an item is to the query, the more helpful it will be for the user. So essentially, one is actually interested in a *utility measure* or *fitness function* to measure the how helpful an item is to the user w.r.t. the user's current task. With the different aims in mind, this means that ideally, any good measure will assign higher values to those items, that support the user better to achieve the aim in a way, such that an ordering based on this measure will result in a list with most helpful item on top.

As a consequence, if the user has some strict criterion like for instance "The image should be available without having to use it under a particular license / having to pay royalties for the intended use", this means: Any image, no matter how similar it is to the users' query, will have a *utility* > 0 as long as it doesn't satisfy this criterion.

For all items that do satisfy the criterion, similarity should mimic what the user considers useful for the task. [Santini and Jain, 1999, p. 882] describes this aspect as:

> Whenever a person interrogates a data repository asking for something close, or related, or similar to a certain description or to a sample, there is always the understatement that the similarity at stake is perceptual similarity. If our systems have to respond in an "intuitive" and "intelligent" manner, they must use a similarity model resembling the humans'.

For being able to formulate and evaluate similarity mathematically, one would like to have numerical bounds for it: 1 meaning indistinguishable under some perceptual property – which always has to be the case for identical items, but identity is not a requirement. 0 meaning that items have nothing in common w.r.t. the perceptual property.

The perceptual properties will be represented by a feature. What is missing, is a way to measure similarity such that it adheres the bounds and provides appropriate ordering.

In order to achieve the aim of remaining within the bounds, instead of computing the similarity directly, the dissimilitary between items can be computed first and then turned into a (normalized) similarity score through a correspondence function that maps the dissimilarity values from the range $[0, \infty[$ with 0 being the best possible value to $[0, 1]$ with 1 being the best possible value (cf. [Ciaccia et al., 1998, p. 14], [Weber, 2001, p. 25], and [Schmidt, 2006, pp. 216–221]). The dissimilarity is computed through a distance measure.

In general, a distance measure shall be a function that assigns a real value to any two features. Let $\mathcal{S}$ be the subspace of $\mathcal{F}$ for which a distance measure δ is defined:[18]

$$\delta : \mathcal{S} \times \mathcal{S} \to \mathbb{R} \tag{5.5}$$

[18] Although it is possible to define fairly generic distance measures, e.g., a function that can handle arbitrary vectors of real numbers, it will never be possible to come up with a single distance measure that can compute a meaningful value for any feature and any combination of features defined by $\mathcal{F} \times \mathcal{F}$. And this will also not be needed as there will always be only a certain combination of features and distance measures – they will never "live" in isolation. The only important aspect will be that for a combination of ϕ and δ that for every image $\mathfrak{I}$, $\phi(\mathfrak{I}) \in \mathcal{S}$ such that for every valid image there will be a feature that can be processed by the distance measure.

5.3.1 Properties of Distance Measures

Two properties that very many distance measures provide are:

$$\textbf{Self identity: } \forall x \in \mathcal{S} : \delta(x, x) = 0 \tag{5.6}$$

$$\textbf{Non-negativity: } \forall x, y \in \mathcal{S} : \delta(x, y) \geq 0 \tag{5.7}$$

A property that can also be achieved by a number of distance measures is, that it becomes only zero for the identity:

$$\forall x, y \in \mathcal{S} : \delta(x, y) = 0 \text{ if and only if } x = y \tag{5.8}$$

However, whether or not Equation (5.8) has any practical implications like being able to say that two images must be identical if the measured distance is zero depends not only on the used distance measure, but also on the used feature.[19] If a feature extracted with function ϕ is invariant to some transformation t with $t(\mathfrak{I}) \neq \mathfrak{I}$, this will result in $\phi(\mathfrak{I}) = \phi(t(\mathfrak{I}))$ as this is the definition of invariance in Equation (5.2). And whenever the extracted feature $\phi(\mathfrak{I})$ is not unique for all $\mathfrak{I} \in \mathcal{I}$ there may also be collisions in feature space that would result in the invalidation of the assumption that two images must be identical if the measured distance is zero.

On the other hand, any distance measure that cannot guarantee that it satisfies Equation (5.8) can provide additional invariance that the used feature did not offer by itself. For instance, one could define an ϵ-insensitive distance measure based on any δ, that intentionally ignores any very small differences:

$$\delta_\epsilon(x, y) = \begin{cases} 0 & \delta(x, y) < \epsilon \\ \delta(x, y) - \epsilon & \text{otherwise} \end{cases} \tag{5.9}$$

Such an ϵ-insensitive distance measure could be used to provide additional matching tolerance for one combination of feature ϕ and distance measure δ that would not have by itself.

Symmetry is another property which is frequently assumed and desired for distances as it matches certainly the experience and definition of distance in real world, where *distance* does not depend on a (travel) direction.

$$\textbf{Symmetry: } \forall x, y \in \mathcal{S} : \delta(x, y) = \delta(y, x) \tag{5.10}$$

The triangle inequality allows to give bounds: The direct distance between two items can never be further than the sum of distances using one or more intermediate hops.

$$\textbf{Triangle inequality: } \forall x, y, z \in \mathcal{S} : \delta(x, z) \leq \delta(x, y) + \delta(y, z) \tag{5.11}$$

[19] [Schmidt, 2006, pp. 166] defines "Distance function" as a function that satisfies Equation (5.6), Equation (5.7), and Equation (5.8) but also Equation (5.10) and Equation (5.11) – which are the properties that any *metric* has to satisfy. For discussing also functions that do not necessarily satisfy all these equations, we will use the term "distance measure" as a general concept independent of these mathematical properties.

5.3.2 Minkowski Norms

The Minkowski Norms L_m [Schmidt, 2006, pp. 170ff] define metric distance functions for n-dimensional vectors of real values:[20]

$$L_m : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_0^+$$

$$L_m(\vec{x}, \vec{y}) = \sqrt[m]{\sum_{i=0}^{n} |x_i - y_i|^m} \tag{5.12}$$

For particular values of m, this leads to commonly used, well-known metric distance functions.

$m = 1$: *Manhattan* or *city block distance* Equation (5.13)
$m = 2$: *Euclidean distance* Equation (5.14)
$m = \infty$ *Maximum* or *Tschebyscheff distance* Equation (5.15)

$$L_1(\vec{x}, \vec{y}) = \sum_{i=0}^{n} |x_i - y_i| \tag{5.13}$$

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2} \tag{5.14}$$

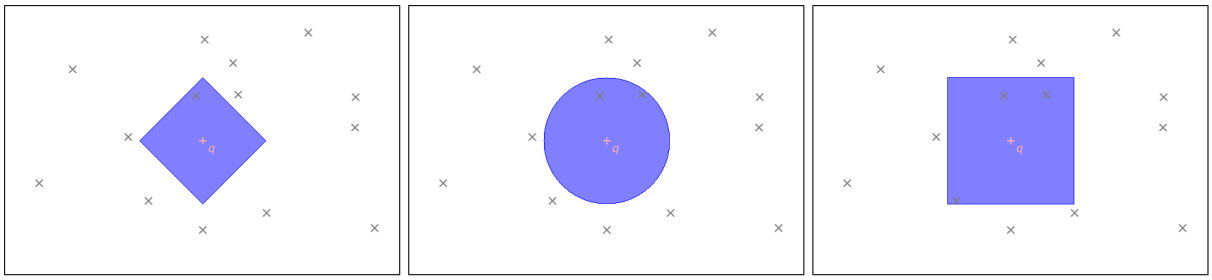
$$L_\infty(\vec{x}, \vec{y}) = \max_{i=0}^{n} |x_i - y_i| \tag{5.15}$$

For illustration purposes, Figure 5.11 shows the distances in $\mathbb{R}^2$ with a space defined by 600×400 pixels. The point q is located in the center, so having the coordinates $(300, 200)$ and the blue area in each illustration covers any $p \in \mathbb{R}^2$ for which $L_m(q, p) \leq 100$. For $m = 1$ and therefore the *Manhattan distance* in Figure 5.11(a) this means that the sum of the absolute difference in the coordinates may not exceed 100. For $m = 2$ and therefore the *Euclidean distance* in (b) this means that the area corresponds graphically to a circle with a radius of 100 around q. For $m = \infty$ in (c) the area corresponds to all points where in each dimension the coordinates do not differ by more than 100.

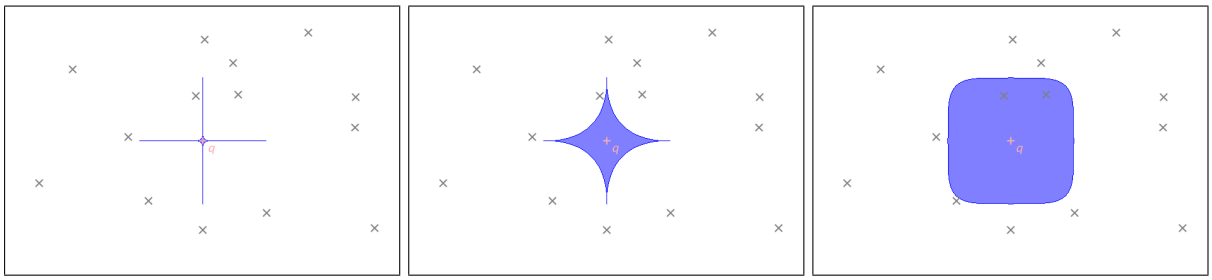
Additional values of m are shown in (d)–(f) – even though they have no great importance in real searches. But they can show illustrate the impact a little further that m has for the *matching tolerance*: The smaller the value of m, the more the tolerance will be given just to follow the axes; the greater the value of m, the more the tolerance is evaluated for each axis.

This impact becomes important when considering how such searches in high dimensional feature spaces relate (or do not follow) to our common intuition that origins from our experiences $\mathbb{R}^1$, $\mathbb{R}^2$, and $\mathbb{R}^3$: In $\mathbb{R}^1$, it is common to rather state just the bounds, e.g., the range $x \in [200, 400]$ instead of specifying the middle 300 and the range 100. In

[20] In an n-dimensional vector space, any vector can represent an individual point in space. Therefore the distance between two vectors in such a feature space corresponds to the distance between the two points. A *norm* is the function that computes the length of a vector $\vec{v}$in a normed vector space and usually denoted by $||\vec{v}||$. Therefore *norm* is frequently used as a synonym for those distance functions that can be used to defined normed vector spaces. See also [Weber, 2001, pp. 18ff].
Note also that due to the relationship to the Lebesgue integrable functions known as L^p spaces, some literature may use p instead of m to index the norms, hence refer to ℓ^p norms.



(a) L_1 (b) L_2 (c) L_∞



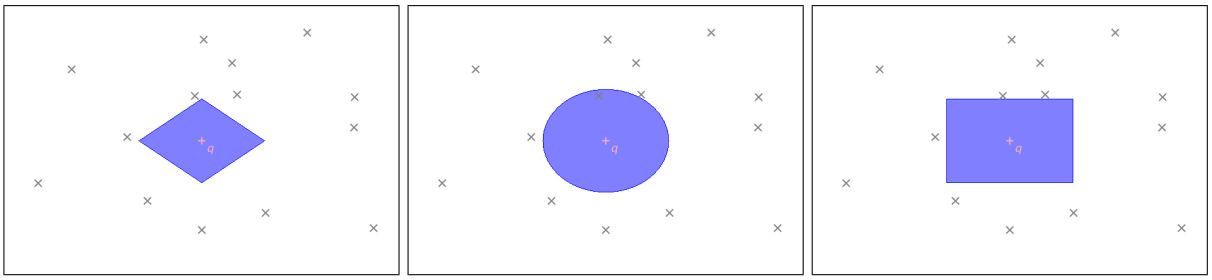
(d) $L_{0.2}$ (e) $L_{0.5}$ (f) L_5

Figure 5.11: Illustrations of Minkowski distance functions in two dimensional space: Each image shows the space of 600×400 pixels. The point q symbolizes the query (colored in pink), the blue area around contains all points that have a distance less than 100. The other points with diagonal crosses indicate some documents in the collection. (a) shows L_1 a.k.a. Manhattan, city block, or taxicab distance, (b) shows L_2 the Euclidean distance, (c) shows L_∞ a.k.a. Maximum distance. (d)–(f) show additional values which have less practical importance.

$\mathbb{R}^2$, specifying a range of 100 does not correspond to a query with $x \in [200, 400]$ and $y \in [100, 300]$ – unless $m = \infty$. For all other values of m, these bounds would only address some extreme points, but for the vast majority of points within the bounds, we would really need to compute (or, for instance, draw the circle) to determine whether points lie inside or outside the range as our intuition becomes insufficient as a criterion. In $\mathbb{R}^3$, it already gets harder to visualize and with each added dimension, the bounds that are easy to compute for $m = \infty$ become less helpful for estimating which area gets covered – and as a consequence, which items of the collection would be selected if $m \neq \infty$. [Aggarwal et al., 2001] reported that for a number of applications, the higher the number of dimensions, the better distance metrics with lower values of m performed – and therefore fractional distance metrics with $0 < m < 1$ may (and did in experiments) outperform established distance measures like the Euclidean or Manhattan distance. Fractional distance measures prefer points that differ only towards one dimension significantly over points that differ only a little but in all dimensions.

Variants of Minkowski Norms

W.r.t. to using Minkowski norms as distance measures for similarity search, this means that we must pay even higher attention that the distance measures captures the prefer-



(a) L_1 (b) L_2 (c) L_∞

Figure 5.12: Illustrations of weighted Minkowski distance functions in two dimensional space with weights $(1, 1.5)$ to compensate for the different ranges in the two dimensions with 600×400 pixels. Compared to Figure 5.11, the differences in the Y dimension are given greater importance, therefore the area within the distance 100 spreads *less* in that dimension.

ence of the user. One important aspect for this can be how different dimensions are perceived by the user: If each dimension has the same importance, the Minkowski norms may work directly. If not, weighting might become necessary.

$$L_{m,\vec{w}}(\vec{x},\vec{y}) = \sqrt[m]{\sum_{i=0}^{n} w_i |x_i - y_i|^m} \tag{5.16}$$

Equation (5.16) shows a weighted version of Equation (5.12) (cf. [Schmidt, 2006, p. 174] and [Weber, 2001, p. 19]). For the example space with 600×400, we may want to use the weights $(1, \frac{600}{400})$ to achieve that the distances in dimension X can have at most the same impact as in dimension Y. Figure 5.12 shows the illustrations for the weighted Minkowski distances. Thus, for instance, if a feature like the color moments [Stricker and Orengo, 1995] described on page 115 carries in one dimension a average luminance value and this should receive less importance, this could be achieved by weighting.[21] [22] And if a feature is the concatenation of values derived for individual regions, weighting can be used to assign different importance to different regions – and in the extreme case ignore some areas by assigning the weight 0, to switch the

[21] Another, complimentary option is, to create a variant of the distance metric that is ϵ-insensitive *per dimension*, not just on the overall distance as defined in Equation (5.9). Sometimes, it may not just be desired to ignore small distances per dimension, but also limit the effect of individual dimensions. A simple approach to achieve this can be to use a threshold t [Springmann et al., 2008]. Using the shorthand functions min and max to get a compact representation instead of cases, the variant of Equation (5.16) would then be:

$$L_{m,\vec{w},\vec{\epsilon},t}(\vec{x},\vec{y}) = \sqrt[m]{\sum_{i=0}^{n} w_i (\min(\max(|x_i - y_i| - \epsilon_i, 0), t))^m}$$

[22] For many Histogram-based features, the proposed distance function is a *Quadratic distance function* of the form $\delta_A(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \times A \times (\vec{x}-\vec{y})}$ [Schmidt, 2006, pp. 175–179] and [Weber, 2001, pp. 19–22], which is used for instance in [Carson et al., 2002]. For some values of A, this can represent a rotation of the space in order to better correspond to the perceptual differences of the colors in the histogram. A similarity measure the use with color histograms for computer vision tasks like object detection, identification, and localization rather than comparing images is the *Histogram Intersection* [Swain and Ballard, 1991].

semantic from *relevant/empty* to *irrelevant/unknown* and negative weights for *unwanted areas* (cf. Chapter 2.3.3 on page 53).

Therefore, depending on the semantics of feature, there might be several slight variations of the distance function that lead to results closer to the users' intention. For the image distortion model [Keysers et al., 2004, Keysers et al., 2007] as mentioned already as a feature on page 122, the main aspect is actually not captured in the feature itself (which is essentially just a scaled down version of the image with or without the application of an edge detection filter[23]), but in the distance function: If two images $\mathfrak{Q}, \mathfrak{R} \in \mathcal{I}$ are scaled to common height and width, δ_{IDM} allows for local displacements within a so called local *warp range* w. Equation (5.17) shows IDM a basic version where $\mathfrak{Q}(x, y)$ denote the value of the pixel at position (x, y) and the warp range be defined in the number of pixels that one pixel is allowed to be shifted in any dimension. Increasing the warp range w will provide higher tolerance in matching. Common values for IDM can be to scale the images to at most 32 pixels on the longer side and allowing to displace individual pixels by at most $w = 2$ or $w = 3$ pixels, thus minimizing the distance for each pixel over an area of 5×5 or 7×7 pixels.

$$\delta_{IDM}(\mathfrak{Q}, \mathfrak{R}) = \sqrt{\sum_{x=0}^{width} \sum_{y=0}^{height} \min_{\substack{x' \in [x-w, x+w], \\ y' \in [y-w, y+w]}} (\mathfrak{Q}(x, y) - \mathfrak{R}(x', y'))^2} \tag{5.17}$$

If pixels are treated independently of each other, this may lead to rather inconsistent deformations. In order to restrict these deformations, one can either optimize pixels not independently but use for instance a (pseudo 2D) Hidden Markov Model [Keysers et al., 2007] or just consider not individual pixels but areas in the optimization. The latter is commonly referred to *local context* has been done in most uses of the image distortion model, including [Keysers et al., 2007] and [Springmann et al., 2008]. The local context again can be defined by the number of pixels l that in each dimension shall be considered the context of pixel that is currently evaluated, thus being another parameter that can be defined for each query independently of the feature extraction and therefore allows to adapt to the matching tolerance that the user needs. The resulting formula to compute IDM with local context is shown in Equation (5.18):

$$\delta_{IDM}(\mathfrak{Q}, \mathfrak{R}) = \sqrt{\sum_{x=0}^{width} \sum_{y=0}^{height} \min_{\substack{x' \in [x-w, x+w], \\ y' \in [y-w, y+w]}} \left(\frac{\sum_{\hat{x}=-l}^{l} \sum_{\hat{y}=-l}^{l} \mathfrak{Q}(x + \hat{x}, y + \hat{y}) - \mathfrak{R}(x' + \hat{x}, y' + \hat{y}))^2}{(2l + 1)^2} \right)} \tag{5.18}$$

Figure 5.13 illustrates the use of the image distortion model as a distance measure for two medical images. For IDM, it is common to not scale each image to the same fixed width and height during feature extraction, but to preserve the aspect ratio. $\mathfrak{R}$(x',y') then does not necessarily has to be the pixel exactly at position (x', y') but rather the position corresponding to (x', y') in the query image $\mathfrak{Q}$ w.r.t. the (potentially) different

[23] Therefore $\mathcal{S} \subset \mathcal{F}$ for δ_{IDM} is also a subset of the space of all images: $\mathcal{S} \subset \mathcal{I}$

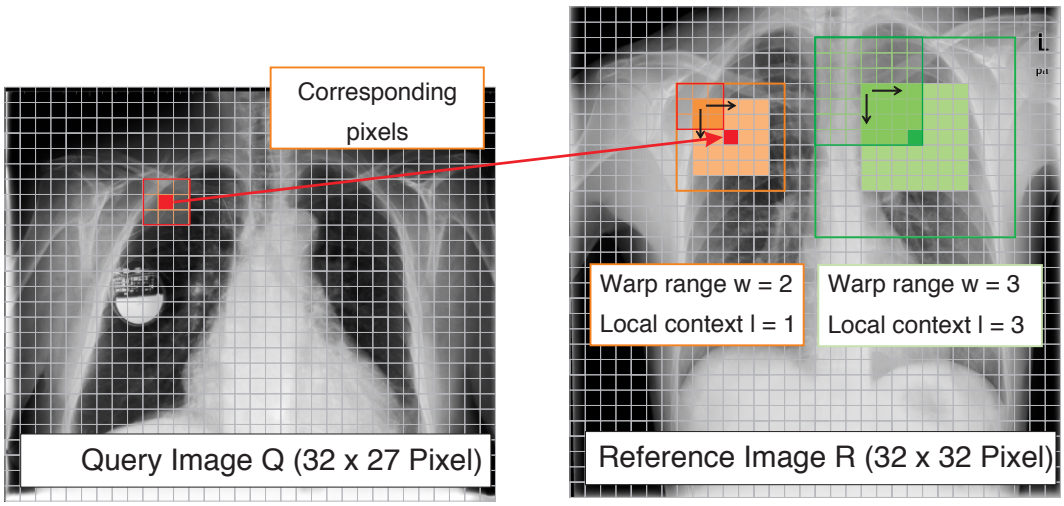
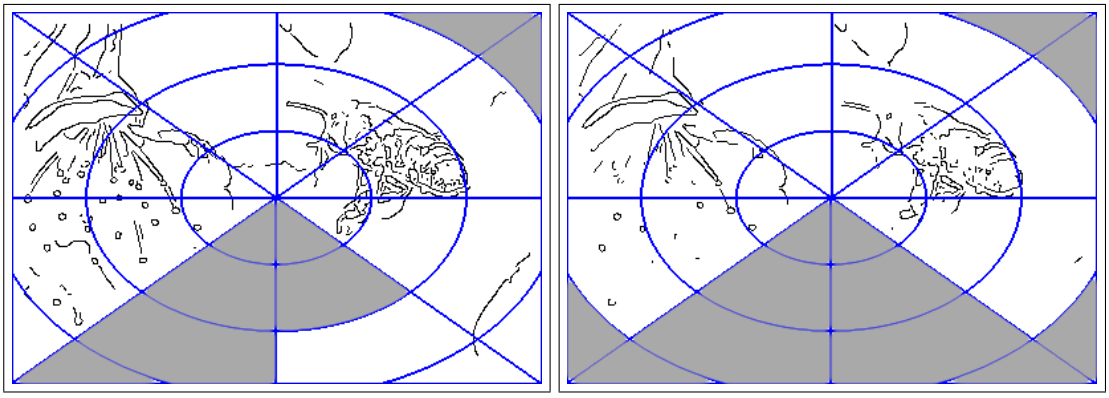


Figure 5.13: Image Distortion Model for two medical images, illustrated for one pixel and it's local context in image Q and the corresponding area to which it may be warped in orange color. The small arrows indicate that the local context area in the reference image R will move to minimize the cost of deformation. The green area in image R shows the effect of choosing higher values for the w and l.



(a) $\beta = 10$ (b) $\beta = 15$

Figure 5.14: Edge maps for ARP with $\beta = 10$ (a) and $\beta = 15$ (a) with areas that are completely empty grayed out. Those gray regions could be interpreted as "unknown" in distance evaluation using a weight or 0. As different values of β lead to different areas being empty, not all gray areas of (b) are also empty in (a).

aspect ratio. This becomes in particular important when using a local context, e.g., $l = 1$ to define a 3×3 patch as local context, as the patch can then maintain the aspect ratio of the original image – and therefore edges in the patch remain their exact directionality. A consequence of this is, that the number of pixels may differ for each image $\mathfrak{Q}$ and as this defines the number of values that have to be summed up in Equation (5.17), for images of different aspect ratio, $d_{IDM}(\mathfrak{Q}, \mathfrak{R}) = d_{IDM}(\mathfrak{R}, \mathfrak{Q})$ may not hold – the image distortion model violates the property of symmetry and is therefore not a metric distance function.

The same might be true for the weighted Minkowski distance functions if the weights are derived from an image itself: Consider Figure 5.14 in which all of the angular / radial partitions have been grayed out which are completely empty. When applying the semantic that empty areas shall correspond to *unknown* areas, a distance function (in case of ARP the Manhattan distance L_1 [Chalechale et al., 2004]) to compare the number of pixels in each partition should ignore those gray areas which can be implemented by assigning these regions a weight of 0 [Springmann et al., 2010b, Springmann et al., 2010a].[24] If those weights are derived only from the query image, symmetry cannot be guaranteed. If on the other hand, the weights depend on both, the query image and the image it is compared to, the distance scores generated for comparing the same query image to different images of the collection will no longer lead to a consistent order that the user would expect.

5.3.3 Metric Proporties Revisited

If we look back on the properties *self identity* in Equation (5.6), *non-negativity* in Equation (5.7), *symmetry* in Equation (5.10) and the *triangle inequality* in Equation (5.11), we can show that none the properties of a metric distance function are necessary requirements to achieve the intended goal of ordering items based on utility as introduced on page 126. For this, the only requirements would be that identical values have the same utility / similarity, the maximum utility corresponds to minimal distance and that the order matches the users' perception.

- *Self identity and non-negativity*: As long as the distance score satisfies Equation (5.19), one can construct a correspondence function that is able to normalize the similarity value within $[0, 1]$ and has appropriate semantics.

$$\forall x, y \in \mathcal{S} : \delta(x, x) \leq \delta(x, y) \tag{5.19}$$

  (Minimality, cf. [Santini and Jain, 1999, p. 872] – and mentions that this property is not necessarily satisfied by human perception as psychological experiments by [Tversky, 1977] has shown.)

- *Symmetry*: Is in many cases not of interest or impossible/inapplicable to real-world user tasks. For instance, consider a known image search performed through Query by Sketching. Symmetry would assume that the known item is as helpful to the user as the sketch. However, the user created the sketch for the sole purpose of finding the image.

- *Triangle inequality*: Also for this property it is known, that human perception does not follow this property. [Santini and Jain, 1995]

  > The triangular inequality is the most debated and troublesome of the metric axioms, since, according to the model, the satisfaction or violation of this property by the function d is not accessible to experimentation. It is common wisdom, however, that the triangular inequality does not hold for human similarity perception.

[24] Cf. also *Symmetrieproblem* [Schmidt, 2006, p. 217]

The properties are sufficient (given proper normalization in the correspondence function), very well understood and can be very helpful for optimization as well as constructing more complex functions from existing one that still provide all needed properties.[25]

5.3.4 Distance Measures for Complex Queries

As a user may want to specify more than a single example in the query, we defined already the function Φ_{MO} on page 112 to extract the features of a set of images to specify a query with a set of query images $\mathcal{Q} \subset \mathcal{P}(\mathcal{I})$. Such queries are called *Multi–Feature Single–Object Queries* [Böhm et al., 2001a, p. 215].[26] To operate on this set of images elegantly, we therefore need a distance measure that can operate on the set of features computed by $\Phi_{MO}(\mathcal{Q})$ to compare it with a single feature from single image from the collection $\phi(\mathfrak{R})$:

$$\Delta_{MO} : \mathcal{P}(\mathcal{F}) \times \mathcal{F} \to \mathbb{R}$$

Particular distance measures can be defined to incorporate the desired semantics for the user, e.g., [Böhm et al., 2001a, p. 214] proposes three different distance combining functions: *Weighted Average*, *Fuzzy-And*, and *Fuzzy-Or*.[27] Let q_i be the i-th feature of $Q = \Phi_{MO}(\mathcal{Q})$ and $r = \phi(\mathfrak{R})$.

Weighted Average has the semantics that items should be similar to all query images, with weights defining the importance of each query image. Equation (5.20) shows the calculation of the aggregated distance and Figure 5.15(a)–(c) illustrations of an example.

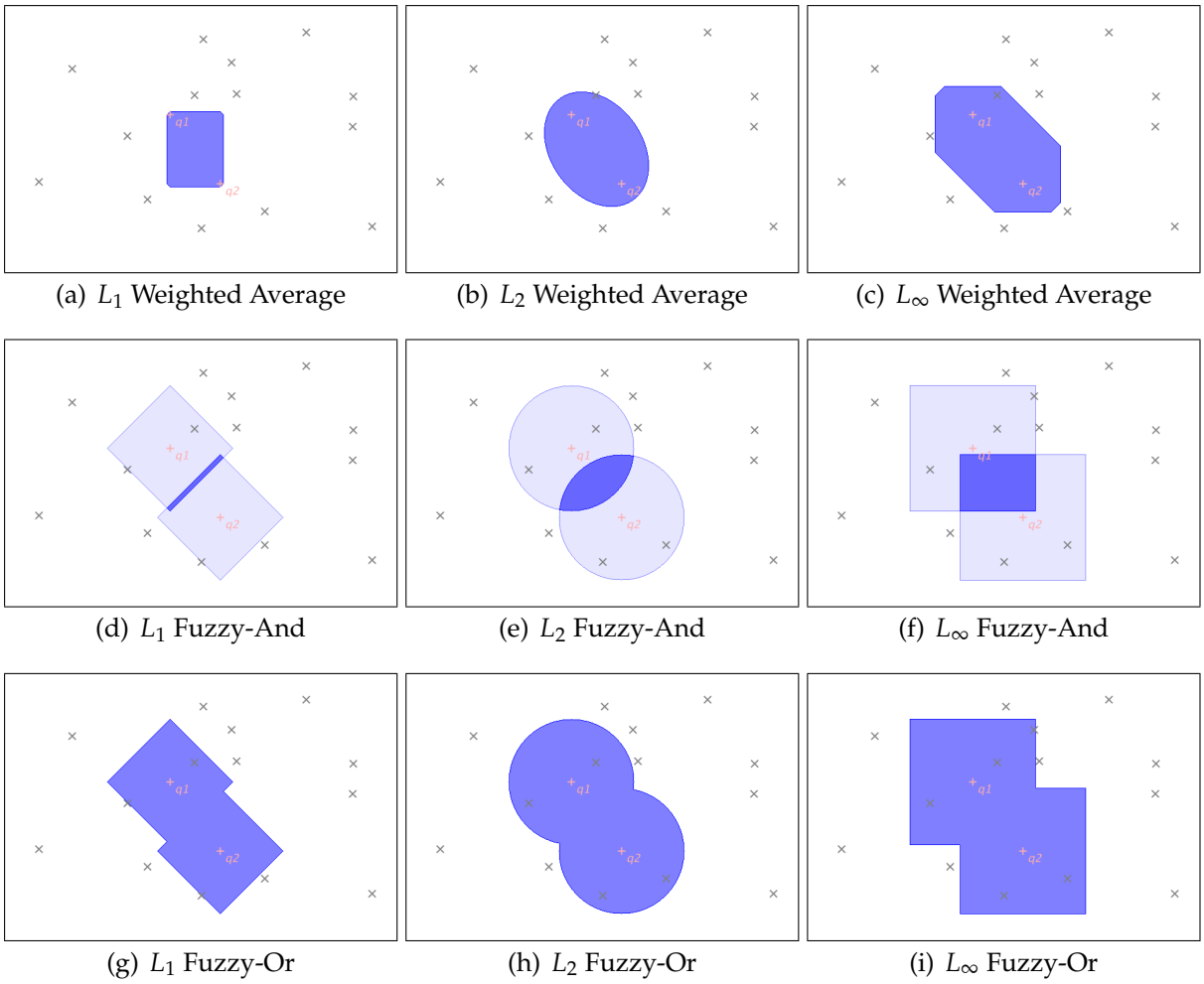
$$\Delta_{MO}^{avg}(Q, r) = \frac{\sum_{i=0}^{|Q|} w_i \delta(q_i, r)}{\sum_{i=0}^{|Q|} w_i} \tag{5.20}$$

Fuzzy-And has the semantics that items are not allowed to deviate from either of the two – it is not sufficient to be similar to one, the have to be similar to all of them. Equation (5.21) shows the calculation of the aggregated distance for which the computation of the maximum can be used and Figure 5.15(d)–(f) examples. The dark blue area contains all points in space that are similar to the features q_1 and q_2 – which is made more

[25] Alternative models for evaluating similarity have been proposed in [Tversky, 1977] in the form of the Feature Contrast Model and the Fuzzy Feature Contrast Model in [Santini and Jain, 1999]. [Rorissa, 2007] tested Tversky's Feature Contrast Model in a study with 150 participants, which confirmed the model for image similarity using user-assigned textual attributes as features. However, [Rorissa et al., 2008] has also shown that visual features including scaled down versions of the image to 32×32 pixels compared using the Euclidean distance, color and patch histograms using the Jensen-Shannon divergence correlate to human similarity judgements; thus showing that distance-based methods can also be successfully applied to estimate human similarity judgements – even though they are not based on the Feature Contrast Model.

[26] Cf. also [Weber, 2001, p. 16]

[27] Note on the naming: Fuzzy-And and Fuzzy-Or do not relate to fuzzification of values, but rather on the similarity in which they are implemented. Just as the operations in Fuzzy Logic / Fuzzy Sets for $\cup$ and $\vee$ are computed with the maximum and for $\cap$ and $\wedge$ computed with the minimum (cf. [Zadeh, 1965]), these functions compute the semantics of "and" and "or" on multiple objects with the maximum and minimum, respectively.



(a) L_1 Weighted Average (b) L_2 Weighted Average (c) L_∞ Weighted Average

(d) L_1 Fuzzy-And (e) L_2 Fuzzy-And (f) L_∞ Fuzzy-And

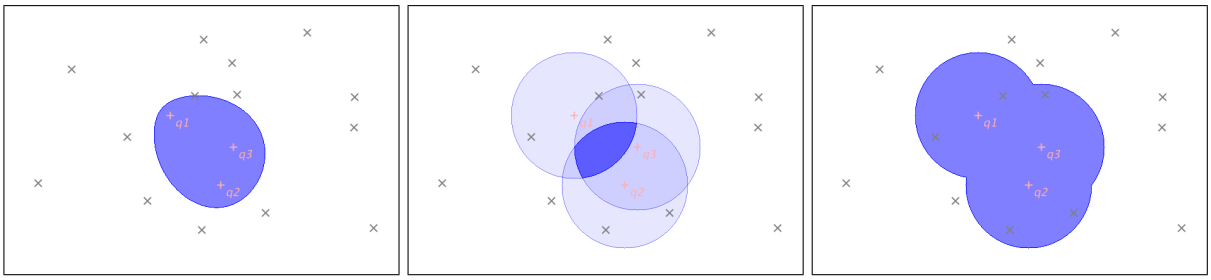
(g) L_1 Fuzzy-Or (h) L_2 Fuzzy-Or (i) L_∞ Fuzzy-Or

Figure 5.15: Illustrations of Aggregated Distances for Single–Feature Multi–Object Queries with two features q_1 and q_2 from two different objects: First row shows (here unweighted) average of unweighted versions of Minkowski norms L_m with $m = 2$ (a), $m = 2$ (b), and $m = \infty$ (c). The second and third row show illustrations for Fuzzy-And (d)–(f) and Fuzzy-Or (g)–(i).

visible by depicting in light blue the area covered by measuring the distance to q_1 and q_2 independently; the dark blue area is precisely the intersection of the two.

$$\Delta_{MO}^{and}(Q, r) = \max_{i=0}^{|Q|} \delta(q_i, r) \tag{5.21}$$

Fuzzy-Or has the semantics that items are allowed to deviate from either of the two – as long as they don't deviate from both. Equation (5.22) shows the calculation of the aggregated distance for which the computation of the minimum can be used and Figure 5.15(g)–(i) examples. The dark blue area contains all points in space that are similar to the features q_1 or q_2 – it is the union of the areas which would be defined by just q_1 and q_2 independently.



(a) L_2 Weighted Average (b) L_2 Fuzzy-And (c) L_2 Fuzzy-Or

Figure 5.16: Illustrations of Aggregated Distances for Single–Feature Multi–Object Queries with three features q_1, q_2, and q_3 from three different objects with L_2 norm.

$$\Delta_{MO}^{or}(Q,r) = \min_{i=0}^{|Q|} \delta(q_i,r) \tag{5.22}$$

Figure 5.16 shows all three aggregated distances with the Euclidean distance for a query with $|\mathcal{Q}| = 3$. W.r.t. to symmetry: Independent of the used distance measure within the evaluation, none of the Δ_{MO} can provide symmetry as they are already defined with different domains for the arguments: A set of features for the first argument, a single feature as the second argument. This definition was based on the semantics a user would use: The first argument is provided by the user – the query constructed of several query objects for which the user can state whether it is sufficient that results are similar to just one or all of them; the second argument is always a feature of images in the collection, not a set of images.

So far, complex queries were restricted to multiple query objects for each of which the same feature was extracted using Φ_{MO}. As already mentioned in Section 5.2 on page 112, it may happen that instead of the same feature for several objects, the user may rather prefer to extract several features from the same objects; in particular, if the search tasks demands that similarity is based on several perceptual features, e.g., color and texture – and a single feature would not be sufficient to capture both. For such Multi–Feature Single–Object Queries [Böhm et al., 2001a], the desired distance measure would directly compute $\Delta_{MF}(\Phi(\mathfrak{Q}), \Phi(\mathfrak{R}))$. It is easily possible to construct such distance measures with the same three basic concepts (and base implementation) of a weighted average, Fuzzy-And, and Fuzzy-Or – but this time computed comparing two corresponding features. For this, it might be necessary that for each feature type, a different distance measure δ is used.[28]

A particular interpretation of Multi–Feature Single–Object Queries can be that the single object (a single image, e.g., the query image $\mathfrak{Q}$) is cut into n regions / segments / keypoints $\mathfrak{Q}_1$, $\mathfrak{Q}_2$, …, $\mathfrak{Q}_n$ and for each of these, an independent feature f_{q_1}, f_{q_1}, …, f_{q_n} is generated. It is particular in a sense, that in this case it might be necessary to identify the corresponding region / segment / keypoint from the image $\mathfrak{Q}$ to which $\mathfrak{Q}$ is compared - as $\mathfrak{R}$ might be cut into m regions $\mathfrak{R}_1$, $\mathfrak{R}_2$, …, $\mathfrak{R}_m$ and 1)

[28] Visualization of the distances for Multi–Feature Single–Object Queries is not possible without adding new dimensions to represent the added features.

m and n may not be the same, and/or 2) for instance $\mathfrak{Q}_1$ might actually match better $\mathfrak{R}_4$ than $\mathfrak{R}_1$. Something like this was for instance the case in Figure 5.3 on page 103 for the extracted SIFT keypoints, but could also occur for segmented regions, e.g., as in [Carson et al., 2002, Ardizzoni et al., 1999, Weber and Mlivoncic, 2003]. In such a case, it is necessary to perform a *matching* step before the actual distance can be computed; in particular if spatial constraints between regions have to be met or no region should be allowed to be matched more than once, which can lead to a global optimization problem. As common matching schemes do share a number of aspects with the ways searches are performed in the context of Query Execution which is described in the Section 5.4, we will not further detail on matching here.

Of course, one could also derive several perceptual features from each of the regions, which would naturally lead to a Multi-Feature Multi-Object Query [Böhm et al., 2001a]. Similarly, the user may wish to provide multiple query objects and still use for each of them several perceptual features. Therefore as a generic counterpart to the feature extraction function Φ introduced in Equation (5.4) on 112, there shall also be a distance measure Δ that operates on the results of Φ:

$$\Delta : \mathcal{P}(\mathcal{S}) \times \mathcal{P}(\mathcal{S}) \rightarrow \mathbb{R} \tag{5.23}$$

Particular implementation of Δ may be constructed from combinations or variants of Δ_{MO} and Δ_{MF} to match the needed semantics. One issue that may occur in any of these situations is normalization: In order to compare the scores generated by different distance measures from different features, these have to be normalized. This does not mean that just the bounds have to be identical, but also that w.r.t. the user's perception of similarity, any deviation by the same amount must correspond to the same perceived (dis-)similarity. For this reason and if the correspondence function to convert the computed dissimilarity into similarity scores provides sufficiently good normalization, it might be beneficial to define the aggregation on the similarity scores rather than distance measurements. Such an approach is described in [Schmidt, 2006, pp. 238–247] providing the same semantics of the Weighted Average, Fuzzy-And, and Fuzzy-Or. As both approaches, aggregating distance scores and aggregating similarity scores, essentially provide the same expressivity for the user and aggregation of distance scores has been detailed already, we will stick to this approach.

5.4 Query Execution

Now that we have defined similarity and distance measures, we can extend the definition of the set of retrieved documents *Res* from Chapter 2.1 on page 20 to ease further definitions and descriptions of algorithms: Let RES be the set of tuples $(dist, img)$ with $img \in Docs$ and $dist$ being the distance to the query $\mathcal{Q}$ using the features defined by Φ and the distance measure Δ:

$$RES = \{(dist, img) | img \in Docs, dist = \Delta(\Phi(\mathcal{Q}), \Phi(\{img\}))\} \tag{5.24}$$

When RES is sorted in ascending order based on the distance, $RES[i]$ shall be the i-th of the ranked result tuple – just like $Res[i]$ was the i-th retrieved document.

RES would always contain all documents in from the collection; this would not be ideal and simply not needed in most cases: The user is mostly interested in documents which are very similar to the query. "Similar" can be defined in absolute or relative terms:

- *Range Search* is a search with absolute similarity criterion: Only those tuples will be displayed to the user where $dist$ remains within some range. For range search based on distance, the range is usually defined only by specifying an upper bound on the distance value $r \in \mathbb{R}_+$ with the implicit understanding that no matter how much below this threshold the distance falls, it will always be a result the user is interested in.[29] [30]

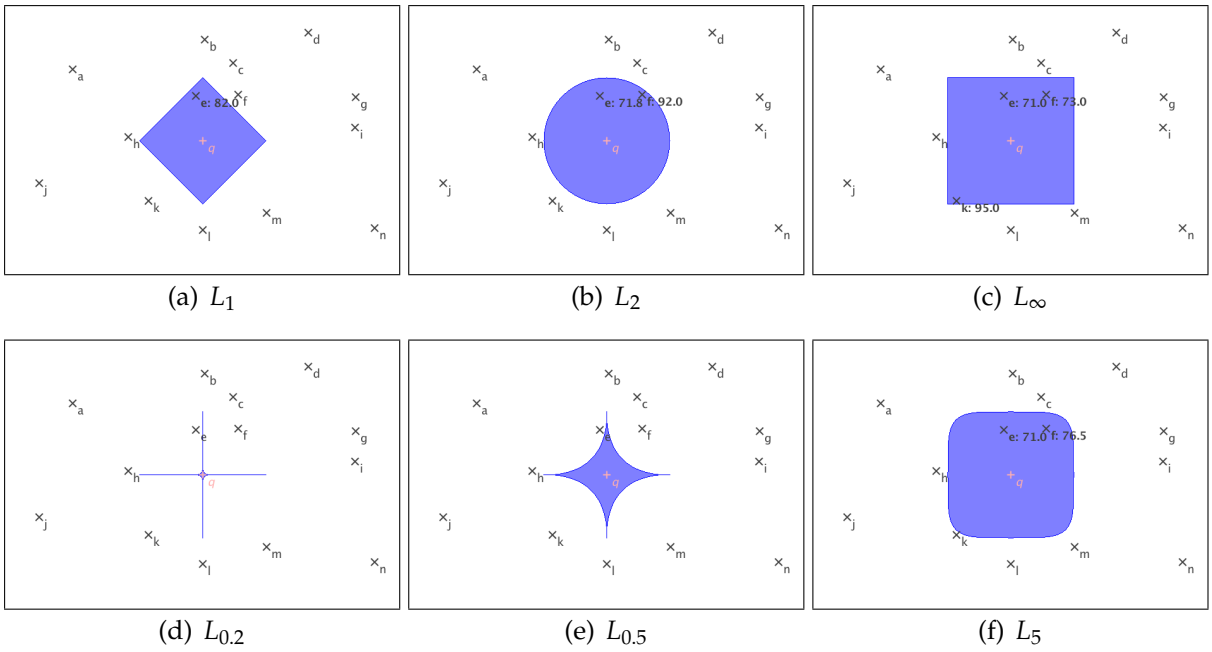
$$RES_{RangeSearch} = \{(dist, img) \in RES | dist \leq r\} \tag{5.25}$$

- *Nearest Neighbor Search* is a search with relative similarity criterion: Like already mentioned in Chapter 2.1 on page 21 one may define a cut-off-value k above which the results become uninteresting. Only the best k tuples will be displayed to the user.

$$\forall i \in \mathbb{N} \wedge i \leq k : RES_{NNSearch}[i] = RES[i] \tag{5.26}$$

[29] Of course, one could also define a range with upper and lower bound $[r^{lb}, r^{ub}]$ and $r^{lb} \leq dist \leq r^{ub}$ being the criterion for the range search. For distance measures satisfying *non-negativity* of Equation (5.7), the common lower bound would always be zero – and even if *non-negativity* would be violated, there's seldom a reason to exclude results with even lower distance. The only common exception to this rule are optimizations for paged result presentation: When there's a button to for getting more results, the search does not need to return again the results that it did return in the last iteration. So if the range of the i-th iteration was $[r_i^{lb}, r_i^{ub}[$, the $i+1$-th iteration will use $[r_{i+1}^{lb}, r_{i+1}^{ub}[$ with $r_{i+1}^{lb} = r_i^{ub}$ and $r_{i+1}^{lb} < r_{i+1}^{ub}$.

[30] The range search could also be defined on *similarity* rather than *distance scores*, which basically just require to keep $r \in [0, 1]$ and replace $\leq$ with $\geq$ in the critirion: $sim \geq r$. However, in addition to not depend on a particular correspondence function for the conversion, the definition based on distance has the very intuitive graphical interpretation used in Figure 5.17(b): The area within the range r around the point q with Euclidean distance can be determined by drawing a circle around q of radius r. For L_∞ as in Figure 5.17(c), the area can be determined by drawing lines parallel to the X and Y axis with distance $\frac{r}{2}$ in each direction – the area between the crossing of these lines that contains the point q is the sought area. For the Manhattan distance as in Figure 5.17(a), we can identify the points directly above and below q by adding / subtracting the vector $(0, \frac{r}{2})$ and perform the same for the points to the left and right by subtracting / adding the vector $(\frac{r}{2}, 0)$ – the square defined by these four corner points is the sought area. All of these graphical interpretations are only possible as r corresponds to a distance; for similarity measures we would have to invert the correspondence function to get back to distances.



(a) L_1 (b) L_2 (c) L_∞

(d) $L_{0.2}$ (e) $L_{0.5}$ (f) L_5

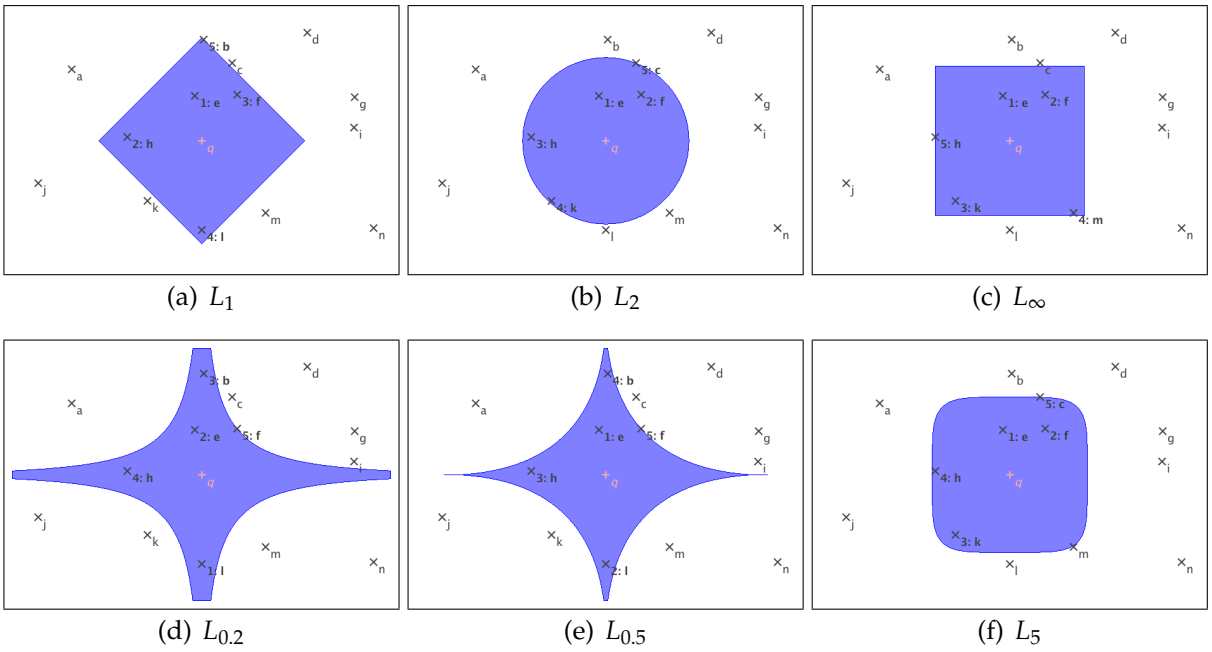
Figure 5.17: Illustrations of Range Search in two dimensional space: The range r is kept constant at 100 for different Minkowski norms. For all points within the range, the label is followed by the distance.

Figure 5.17 illustrates the results of a range search: The points within the area that is covered by the range may differ for each different query. In the examples in Figure 5.17, neither the query q nor the range $r = 100$ have been modified. Only the used distance measure has been changed to a different Minkowski norm for each example. For L_1 in Figure 5.17(a), the range only contains point e. For L_2 in (b), the range contains points e and f. For L_∞ in (c), the range contains points e, f and k. In the case of $L_{0.2}$ in (d) and $L_{0.5}$ in (e) the range is empty. For L_5 in (f), the range contains again points e and f.

Figure 5.18 illustrates the results of a nearest neighbor search: As k was set to 5, each example contains the same number of points in the blue independent of the used Minkowski norm. However, which points are the nearest neighbors and their order may change:

Distance Measure	Nearest Neighbors	Distance of k-th neighbor
L_1	e, h, f, l, b	163.0
L_2	e, f, h, k, c	132.0
L_∞	e, f, k, m, h	118.0
$L_{0.2}$	l, e, b, h, f	2049.6
$L_{0.5}$	e, l, h, b, f	256.9
L_5	e, f, k, h, c	123.2

For every nearest neighbor search, there is a corresponding range search that would return the same documents: The range search with r being the distance of the k-th near-



(a) L_1 (b) L_2 (c) L_∞

(d) $L_{0.2}$ (e) $L_{0.5}$ (f) L_5

Figure 5.18: Illustrations of k Nearest Neighbor Search in two dimensional space: k is kept constant at 5 for different Minkowski norms. For all k nearest neighbors, the rank is printed in front of the label for the point.

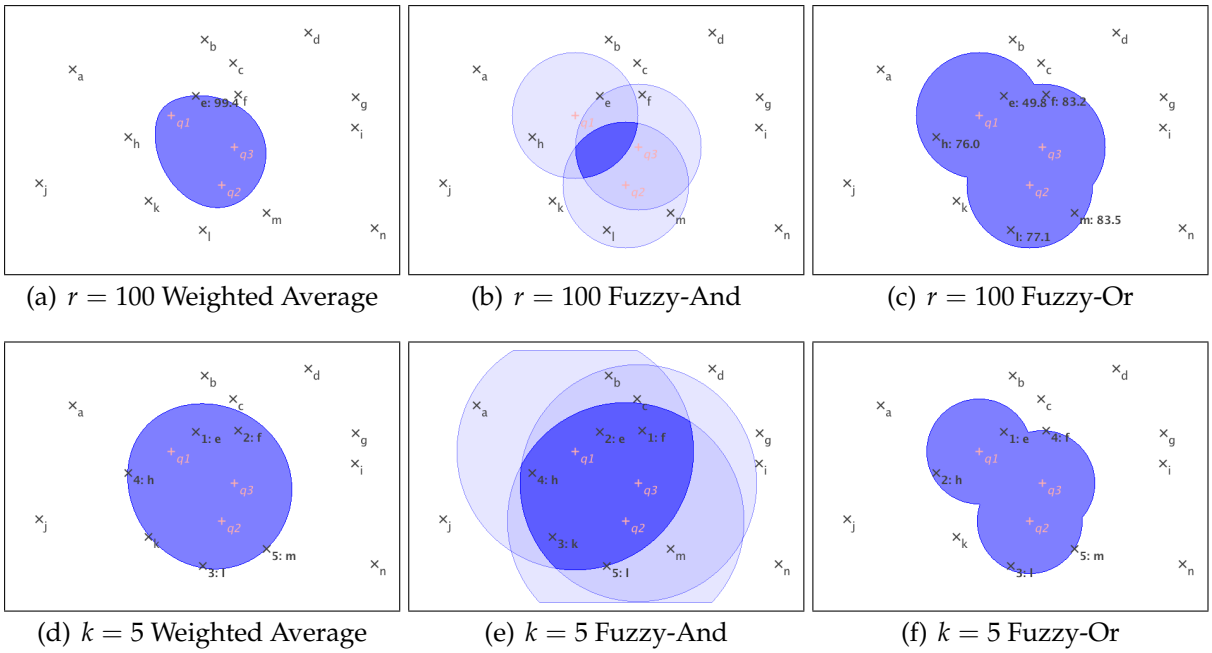
est neighbor. Figure 5.19 shows also some illustrations of with multiple query objects for the Euclidean distance with range and nearest neighbor searches.

5.4.1 Extension to All Aspects of the User's Preferences

Range Search in Equation (5.25) and Nearest Neighbor Search in Equation (5.26) have been defined based on distance evaluations for similarity search. As mentioned in Section 5.3 on page 126, the user is rather interested in *utility* than strict *similarity*, therefore focusing on similarity search in isolation may not be sufficient.

To the user, there are not just perceptual features like those mentioned in Section 5.2; as described in detail in Chapter 2.2.3 on page 40, there are three important aspects to user preferences: *Perceptual attributes*, *interpretational attributes*, and *reactive attributes* [Jörgensen, 1998]. For some of these, textual annotations like keywords or tags as well as class labels might be much more appropriate than, e.g., high-dimensional numerical feature vectors which are commonly used for the perceptual features extracted from the pixel information of the image.

There are also some similarity and distance measures outside of the domain of perceptual features, for instance the Levenshtein distance (a.k.a. (String) Edit Distance, cf. [Schmidt, 2006, pp. 203–205]) which could be used just like any other distance if the text and other attributes are also accessible through some function ϕ; still for many searches, in particular faceted searches, exact match on some of the attributes will be a common requirement. Chapter 4.7 on page 98, handling of exact matches has been described in the context of Content Management.



(a) $r = 100$ Weighted Average (b) $r = 100$ Fuzzy-And (c) $r = 100$ Fuzzy-Or

(d) $k = 5$ Weighted Average (e) $k = 5$ Fuzzy-And (f) $k = 5$ Fuzzy-Or

Figure 5.19: Illustrations of range and kNN search with aggregated distances for Single–Feature Multi–Object Queries with three features q_1, q_2, and q_3 from three different objects with Euclidean distance: First row shows range search with $r = 100$ for Weighted Average (a), Fuzzy-And (b), and Fuzzy-Or (c). The second row shows nearest neighbor search with $k = 5$ for Weighted Average (d) this would be the same area as with $r = 138$, for Fuzzy-And (e) $r = 187$, and for Fuzzy-Or (f) $r = 83$.

Such exact match requirements can be expressed as a filter predicate P:[31]

$$P : \mathcal{I} \rightarrow \{\textbf{true}, \textbf{false}\} \tag{5.27}$$

For instance, the user might be interested only in images that have at least a resolution of 640×480 pixels. Let *width* and *height* be functions that return the height and width in pixels of a given image: $width : \mathcal{I} \rightarrow \mathbb{N}$, $height : \mathcal{I} \rightarrow \mathbb{N}$. We can then easily define a predicate to match the user's requirement:

$$P_{minres}(img) = \begin{cases} \textbf{true} & \text{if } width(img) \geq 640 \wedge height(img) \geq 480 \\ \textbf{false} & \text{otherwise} \end{cases} \tag{5.28}$$

From a pure analysis of the functionality, there are basically three options how to extend similarity-based searches with exact match criteria for some of the attributes:

1. *Restrict the evaluated features to items that satisfy the filter predicate:* Instead of processing all images in *Docs* in Equation (5.24), process only the subset that evaluates $P(img)$ to **true**:

$$RES = \{(dist, img) | img \in Docs \wedge P(img), dist = \Delta(\Phi(\mathcal{Q}), \Phi(\{img\}))\} \tag{5.29}$$

[31] Cf. also Chapter 3 "A Flexible Model for Complex Similarity Queries" in [Mlivoncic, 2006, pp. 33–43]

2. *Turn the filter predicate into a distance measure:* Use an aggregated distance measure / wrap an existing distance measure Δ that evaluates to ∞ whenever $P(\mathfrak{R})$ evaluates to **false**.[32]

$$\Delta_{P,\Delta}(\Phi(\mathcal{Q}),\Phi(\{\mathfrak{R}\})) = \begin{cases} \Delta(\Phi(\mathcal{Q}),\Phi(\{\mathfrak{R}\})) & \text{if } P(\mathfrak{R}) = \textbf{true} \\ \infty & \text{otherwise} \end{cases} \tag{5.30}$$

3. *Filter the result list RES:* Drop any tuple from RES where $P(img) = \textbf{false}$.[33]

$$RES_P = \{(dist, img) \in RES | P(img)\} \tag{5.31}$$

Which of the three options should be used depends in practice mainly on which auxiliary structures exist and how elegantly this can be integrated in the existing implementation, which will be subject of Chapter 10.5.

5.4.2 Relationship between Search and Matching Regions

As mentioned in Section 5.3.4 on page 137, matching is related to the search: For instance, the segmented regions in Blobworld [Carson et al., 2002] from the query image $\mathfrak{Q}$ are matched to the most similar region in $\mathfrak{R}$, so essentially for each region a kNN search is performed in all regions of $\mathfrak{R}$, with $k = 1$. The final distance between $\mathfrak{Q}$ and $\mathfrak{R}$ (or in the case of Blobworld similarity) is aggregated from the individual scores for each region in $\mathfrak{Q}$.

[Ardizzoni et al., 1999, Weber and Mlivoncic, 2003] extend this basic approach by prohibiting multiple assignments of one region in $\mathfrak{R}$ to several regions in $\mathfrak{Q}$, for which it is necessary to solve the assignment problem on a bipartite graph. At the same time, particular semantics can be added what it means if some region cannot be matched as there is no matching region left. [Weber and Mlivoncic, 2003] introduces different penalty values p_1 and p_2 for unmatched regions in $\mathfrak{Q}$ and $\mathfrak{R}$. By concrete assignments of values to p_1 and p_2, the semantics can be $\mathfrak{R}$ *contains* $\mathfrak{Q}$ – therefore regions in $\mathfrak{R}$ that cannot be matched regions in $\mathfrak{Q}$ will not be penalized ($p_1 = 0$) as this is considered irrelevant. The opposite case, that for *contains* some region in $\mathfrak{Q}$ is not matched to any region in $\mathfrak{R}$ has to be penalized ($p_2 > 0$). Notice, that this is the semantics that Blobworld would assume as only regions from the query image $\mathfrak{Q}$ are used in matching. However, since Blobworld does not prohibit multiple assignments, it will always find a matching region for any region in $\mathfrak{Q}$. Therefore Blobworld cannot ensure, for instance, that if the query contains two tigers (each represented as a region of it's own) and it is compared to an image containing only one tiger region, both tigers might get matched

[32] For range searches with reasonable values of r (that is, $r < \infty$), this will perform as desired. For kNN searches, it would still return k matches independent of the evaluation of $P(\mathfrak{R})$ – the ranking would be as desired, but in case $P(\mathfrak{I}) = \textbf{false}$ for less than k images $\mathfrak{I} \in Docs$, it will return some images with distance ∞. To prevent this, additional post-processing of $RES_{NNSearch}$ can drop any tuple with distance ∞.

[33] Notice that for Range Searches, it is possible to perform the result list filtering on either RES before the actual range search or on $RES_{RangeSearch}$ and therefore after performing the range search. For kNN Searches, filtering *must* be performed before $RES_{NNSearch}$ is computed – as $|RES_{NNSearch}|$ will be only k and removing any item would therefore return to few results.

with this single region. As a consequence, when searching over the entire collection, this image might get returned as the best match in Blobworld even if other images in the collection contain also two tigers (cf. [Weber and Mlivoncic, 2003] – in which this problem is solved through the penalty value p_2).

The inverted semantics is $\mathfrak{R}$ *is-part-of* $\mathfrak{Q}$, such that regions not all regions from $\mathfrak{Q}$ have to be matched to $\mathfrak{R}$ to achieve best distance scores; but for any region in $\mathfrak{R}$ where no matching region in $\mathfrak{Q}$ was found, this has to be penalized. The third and last semantics is, that any region should be matched in both directions, so p_1 and $p_2 > 0$. Notice that if $p_1 = p_2$, this is the only setting in which any of the proposed matching schemes can become symmetric; therefore symmetry being a special case for region based retrieval rather than the common situation.

When keypoints are used for object recognition for instance described in [Lowe, 1999], the assumed semantics usually is *test if the image contains some object*. For this, it is not only necessary to identify the corresponding regions, but also reject if they are too dissimilar. This could be achieved by performing a simple 1-NN search for matching keypoints and rejecting any match that has a distance greater than some global threshold, hence combining range search and nearest neighbor search.[34] However, not all keypoints will be as reliable as others as some may discriminate better between objects. [Lowe, 2004] therefore proposes to perform a 2-NN and compute the ratio between the distance of the first and the second best match, such that distance of the best match has to be less than 0.8 times the distance of the second best match.[35]

So there are a number of usages of the search strategies to implement the matching of regions between two images with the particular use depending on the desired semantics of the matching.

5.4.3 Relationship between Search Strategy and Task Input and Aim

If we consider the use of range and nearest neighbor search for providing similarity search functionality to the user, the following tendency will be apparent:

- For known image search, kNN is the more natural way of searching: The quality of image search depends mainly on the rank of the sought image. The user may give up at a certain k, so limiting by the k is a natural match, whereas defining a range on the distance is not. Returning empty results as it may occur in range searches when the user has not yet developed a "feeling" for the range r, is of absolutely no help to the user.

- For classification, both search paradigms are of interest: The classifier can either be trained to determine class membership based on the absolute or relative distance - the first would be based on a range search, the second on a kNN classifier. In controlled environments, it might be the better choice to define a threshold on the range as this can be fine-tuned to avoid results which are "very far off the query". If we compare this to a kNN classifier, if the query happens to fall in a

[34] In [Weber, 2001, p. 27], such a combined search is called *Best Match Similarity Search*.

[35] Note, that in this case, the details of [Lowe, 2004] require that the second best match is restricted to images from a different object. Therefore it requires a variant of the 2-NN search.

very sparsely populated part of the space, this may result in the nearest neighbors being already so distant, that results should be handled with extreme care. On the other hand, if the environment changes significantly for each query, it may not be possible to empirically derive a good value for r and a kNN classifier may lead to better results.

- For themed search, a lot of the problem lies in presenting good and enough choices to the user. "Enough" can be handled by both, either range search with big ranges or kNN search with a large k. "Good" choices may actually prefer diversity over closest matches (which may result in "more of the same" rather than alternatives), for which post-processing steps like sampling from the found candidates can assist.

The last item already points to the next important building block: When we have computed the query results, we need to display them with some user interface and provide the user with the ability to further interact with the results and the system.

6

User Interaction

Essentially, the user has to pass through two stages when interacting with the system to achieve the aim of any retrieval task.

1. The first stage is the *Query Formulation*, which needs to support allow the user to provide input and adjust the matching tolerance to the needs for this task. Section 6.1 will describe the required functionality for this stage in more details.

2. The second stage is the *Result Presentation*, in which the user is given the possibility to explore the search results and start the usage of the retrieved images and associated metadata. Section 6.2 will provide details on aspects regarding this stage.

These two stages focus on fundamentally different aspects of user interaction; not surprisingly, in combination they have to cover every aspect of image-related search tasks use this is the only way the user can influence and perform the task. Figure 6.1 shows an illustration of these aspects.

Unfortunately for the user, not every search task will end successfully after the first query has been executed and the results displayed. Frequently, it is necessary that the the query has to be modified several times until the user found enough satisfactory results. Therefore there will be a loop that is executed to perform sequences of stage 1 and 2 inside a session until the task has been finished successfully – or the user gives up. The details on how the system can support the user in performing this activity will be described in Section 6.3.

6.1 Initial Query Formulation

During the first stage, the user will express the information need by selecting a information seeking strategy and query input that matches this strategy. As highlighted in Figure 6.1, this relates in ITM to whatever input is available on the axis Task Input and Aim and to the amount of Matching Tolerance that is desirable. The following section will give an overview over the available input methods.

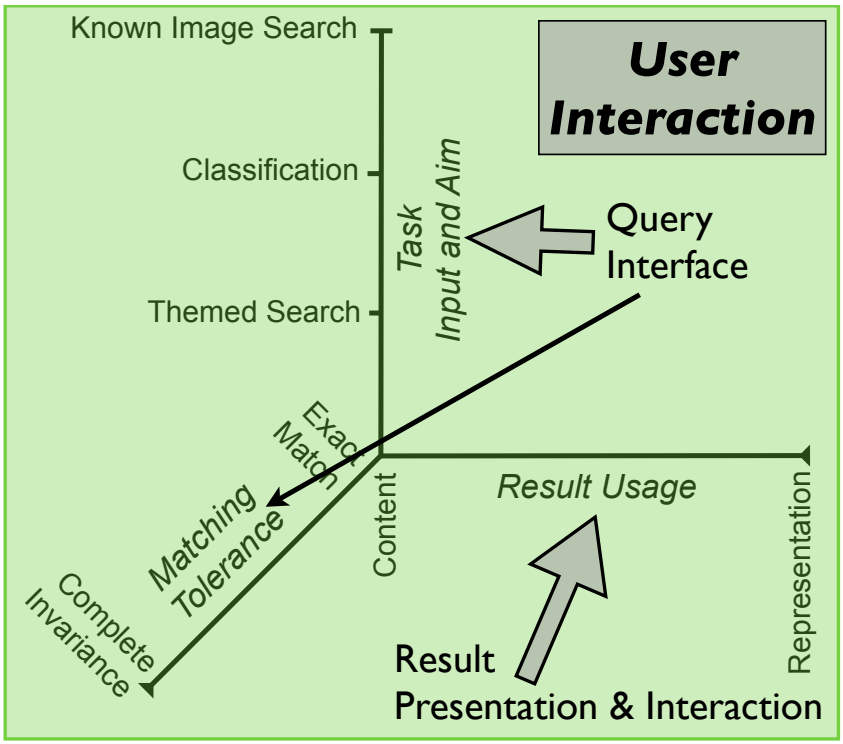


Figure 6.1: Aspects in ITM covered by User Interfaces: The user interface has to support the user in every aspect. In particular by providing an appropriate query interface for the particular task including the ability to influence the matching tolerance according to the user's needs. The presentation of the results must allow the user to perform the intended usage.

6.1.1 Overview on Input Methods for Seeking Strategies

The supported information seeking strategies may lead to significantly different user interface designs and may even affect the choice of targeted input devices. The main distinction is based on the kind of input:

- *Textual* or *numeric query input* as well as selecting from a *distinct set of values* like a particular class will usually lead to *exact match searches* which are mainly metadata-driven. They provide similar techniques as used for other media types, in particular text documents – but may also be based on metadata that has been derived by analyzing the image. If we consider the evaluation of these queries, in most cases, this input will get evaluated in a *predicate* as described in Chapter 5.4.1 on page 140f. Matching tolerance can be added by using ranges for values rather than a single value (e.g., date of picture between January 1 and March 31, resolution >2 megapixels), providing alternatives (e.g., contains cat *or* dog), exploiting ontologies (e.g., include all pets instead of just cats and dogs), and alike.[1]

[1]Of course, also similarity-based matching is possible, e.g., compute the edit distance for longer (free-) texts or rank based on absolute difference of numerical values directly. Although possibly, in practice for image retrieval, it is much more common to evaluate these inputs as predicates, which can even integrate

  - *Keyword-based search* in a simple case will require just a simple text entry field[2] to search any textual description of the image, collected information about the creator/owner or title, annotation, tags. Additional text may have been extract by applying optical character recognition (OCR) on the image content to identify text inside the image.
  - *Fielded search* may allow the user to restrict the search on individual fields as it is common for searches in library catalogues, e.g., searching only for the name of the creator or only in the title. Therefore this would usually provide several field to enter the search terms and a drop down list to select the field plus the semantics to connect several such selected query elements, for instance, using the boolean operators *and* or *or*. Fielded search may therefore be a more structured search compared to plain keyword-based search.
  - Whenever there is a limited number of possible choices, e.g., if there are predetermined categories or a generated list of all tags assigned to images, this can be turned into items the user may interact with. Complex classification backed by an hierarchical structure or ontology may be presented as a tree, which the user can traverse. For flat hierarchies like they are common for user-assigned tags but also classifications without intermediate subclasses, a list of instances can be presented to the user to choose from. From the user interface perspective, it may be helpful to display representative images for each item rather or in addition to the textual name, e.g., if people were detected and identified/tagged in the images of collection, a photo of a person instead just the name of the person can be displayed. In [Del Bimbo, 1999, p. 20], such a strategy is called *Iconic querying*.

    Faceted search may be used to provide several independent categories to quickly filter the set of results, just like boolean operators are used to combine several individual query elements for fielded search.

    Any manual or automatic classification can contribute classes, e.g., even analyzing the image for a limited set of dominant colors. Other instances of derived metadata may include image size, aspect ratio, kinds of images (in particular whether they are black and white photos, color photos, or clipart-like illustrations), automatically detected objects or people, etc.

- *Image and sketch query input* can be used for *content-based similarity searches* and allow the user to provide visual examples to search for similar content based on perceptual features from Chapter 5.2 on page 110ff. For these kinds of inputs, searching for an exact match would provide no value to the user; for instance, if

to a certain extent the mentioned techniques: Edit distance can be applied on the individual terms in the query text to add alternative / corrected spellings and any single numerical value can be converted into a range by providing the bounds. The difference is, that the results of a predicate search follow closely the semantics of a result set whereas similarity search would return ranked lists. In particular when several predicates are combined in a faceted search, this set semantics is desired, whereas ordering of the results might be on different, potentially unrelated criteria like alphabetic or chronologic order, popularity/ratings/votes, or similarity to an example.

[2] Which may be as simple as the traditional Google web search form.

we consider a user-drawn sketch, we have to assume that no other sketch will ever be identical. Hence, it is an inherent property of this input that it has to be compared on similarity.

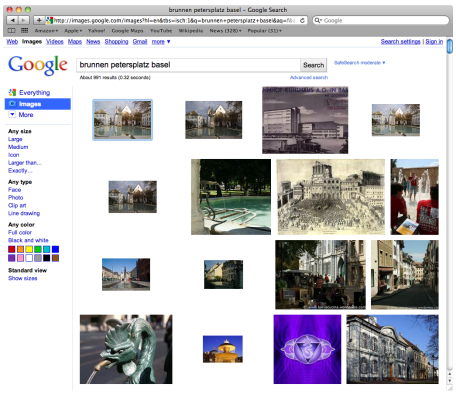
- *Query by Example* takes an image that is provided by the user. For this, the user may load or insert an image to the query that did exist or take an image at the time of the query formulation.[3] For the first option, loading the image file from filesystem, adding through drag 'n' drop of an image from the screen or providing context menu entries for images (in particular in web browser windows) or common UI operations to implement the selection. For generating new images, built-in cameras like webcams in PCs or if mobile phones with cameras are used can be means to avoid having to first transfer image files before being able to initiate searches.[4]
- *Query by Sketch* lets the user draw the example. For this, the common input methods used with (desktop or mobile) computers are not very user-friendly and not precise enough. Therefore other input devices like graphic tablets, tablet PCs, digital pen and paper can provide better alternatives that should be considered when designing the user interface as described in Chapter 2.3.2 on page 49ff and displayed in Figure 6.5.[5]

To continue the discussion in a less abstract manner and as it is an integral part of the user interaction, the next section will provide example interfaces for providing

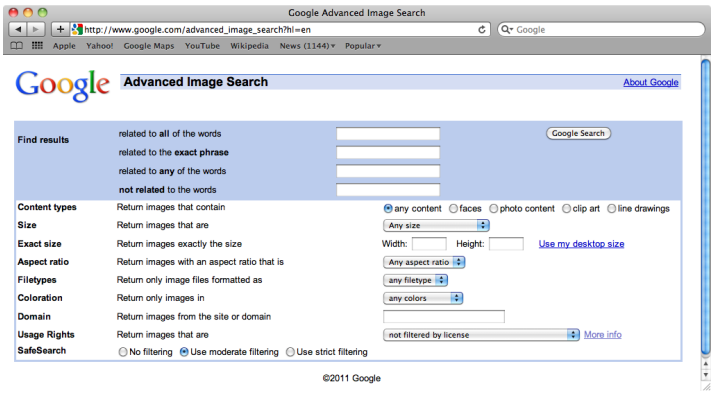
[3] In [Del Bimbo, 1999, p. 20] this strategy is called *Query by Image*. We prefer to stick to *Query by Example* in a sense that the desired results are expected to be from the same domain as the inputs it is used also in other domains, e.g., for database queries – and in this particular domain, as the input happens to an image, also the results are expected to be images. We prefer this notion, as it makes clear that the example is provided by the user – while in *Iconic querying* the image/icon is not provided by the user.

[4] The availability of smartphones with affordable internet connections has boosted interest on methods to acquire images on the phone and use within searches. Through content analysis on the phone or on a server, this does not restrict the use only to *similarity searches*, but may also be a more convenient input method for metadata-driven aspects; in particular scanning a bar code from a picture can result in faster and more accurate input than entering (alpha-)numerical values. But also "selecting" from a number of known classes or instances can be performed more convenient than letting the user scroll through long lists or type with the (touch screen) keyboard the name. In these cases, the image ultimately is converted into (distinct) values as query input as desired such that they can get evaluated in a predicate-like manner, even if –in particular for selecting classes– intermediately perceptual features may get extracted.

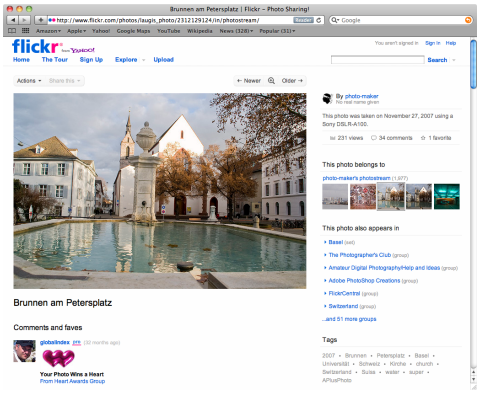
[5] In [Del Bimbo, 1999, p. 20], an explicit distinction is used between *Query by painting* which is described as bing "usually employed in colour-based retrieval" and *Query by sketch* which is described as "commonly used to retrieve images that contain objects with shapes similar to the example". In this thesis, we do not make this distinction as it frequently hard to maintain: The distinction is entirely based on the kind of perceptual features used to compare the user input to the images in the database. For instance, even if colored regions were used, the boundary of the regions can be used to define a shape and on the other hand, lines in sketch may not only indicate the shape, but also the texture of objects. We therefore consider "sketch" in a less strict sense that it refers to some visual input created by the user for the sole purpose of search, therefore being not a finished work in contrast to images stored in the database, which are finished works. Therefore we use the term "sketch" independent of whether it contains colors or not and rather as a distinction that the input is from a different domain than the desired results, in contrast to *Query by Example*, where they are from the same domain. Notice that there may exist borderline cases, for instance, if the database actually contains only sketches, e.g., documentation of the early stages of new designs in fashion or for cars.



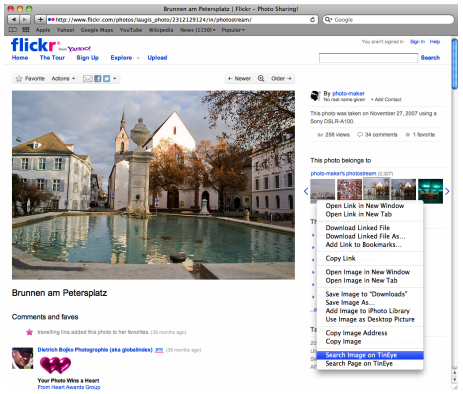
(a) Google Images



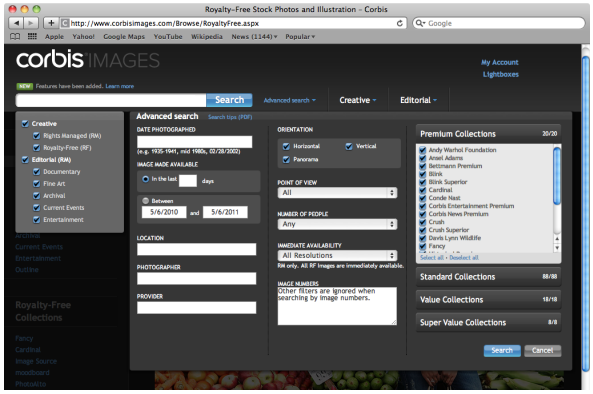
(b) Google Advanced



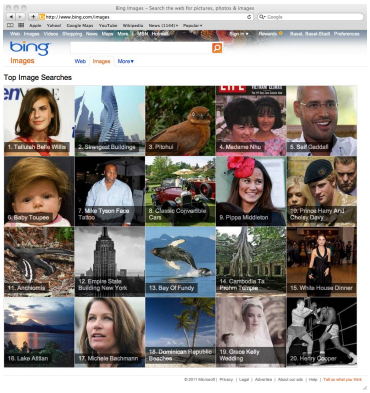
(c) Flickr



(d) TinEye Context Menu Option



(e) Corbis Advanced



(f) Bing Images

Figure 6.2: Examples of Search Input Methods I: All of the depicted webpages provide a simple text input field to search for images. In addition, various other methods are present to provide alternative and additional means to find images.

the query input which also shows how some of the input methods can be combined in practice. It will also highlight how the intended application area which may be closely related to the Task Input and Aim can influence user interface design decisions.

6.1.2 Example Interfaces for Textual, Numerical, and Category/Class Input

Figure 6.2 shows various examples of different input methods for the query formulation that fall into the first group of textual or numerical query inputs.[6]

Google Images[7] provides additional facets as shown in Figure 6.2(a) as soon as the initial text search was performed on the left side of the page; similar and additional options are available in the advanced search form shown in Figure 6.2(b). Similar options with an even stronger focus on rights management can be found in the advanced search for stock photos from Corbis[8] shown in Figure 6.2(e). Another metadata-driven search form the (virtual) museum context of the Web Gallery of Art [Web Gallery of Art, 1996][9] is displayed in Figure 6.13(a) on page 163.

When an image is displayed on Flickr[10] like the one in (c) from the fountain search scenario in Chapter 1.4.1 in Figure 1.2, navigational links are provided to switch to *groups* or *tags*.

TinEye[11] provides plugins for popular browsers to ease reverse image search as shown in Figure 6.2(d): From the context menu after right-clicking on any image inside the browser, an entry allows to search for almost identical images used on different websites.[12]

The start page for Microsoft's Bing Image search[13] shown in Figure 6.2(f) adds search entries that have been popular recently, including some sample images. This can be seen as an attempt to assist in exploring the available images.

Due to the exploratory character of *Themed Searches*, it can be very helpful to provide some selected examples of popular content to select from – right from the beginning to assist in browsing the collection rather than targeted searches. The selection of these images can be done either through curation, user ratings, but also by automatic selection based on some popularity metric which may include number of times an image has been clicked, commented, or linked on the web.

---

[6] The illustrations in this chapter are used to give an impression on what possibilities exist to provide the functional aspects of the building block. As this chapter is dedicated on functional aspects, the discussion is limited to the question of "What is provided?" rather than "How (well) is the functionality provided?" and the providing one or two examples is considered sufficient as there will be no comparison between various implementations.

[7] `http://images.google.com/`

[8] `http://www.corbisimages.com`

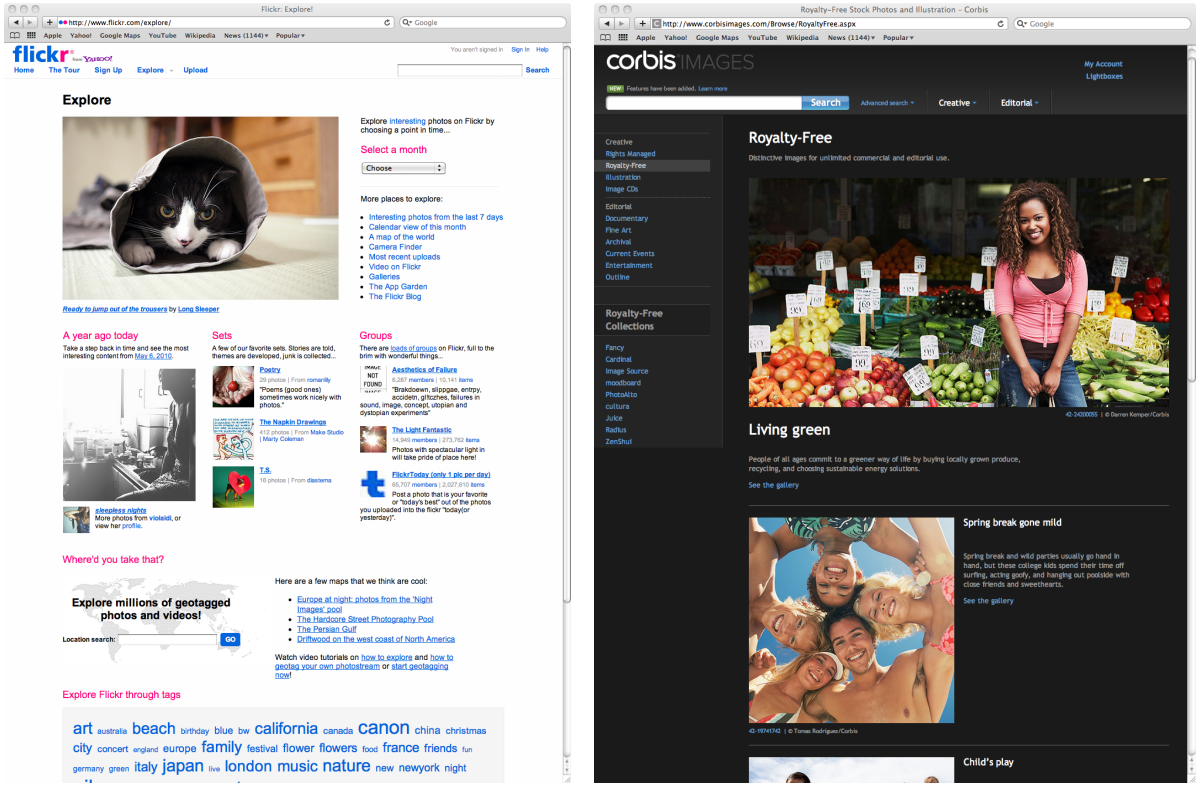
[9] `http://www.wga.hu/frames-e.html?/search.html`

[10] `http://www.flickr.com`

[11] `http://www.tineye.com/`

[12] A very similar browser plug-in was recently made available for Firefox and Google Chrome for Google's "Search by Image" feature [Google Inc., 2011c, Wright, 2011, Singhal, 2011]. The search results make it easy to identify websites that use the same or very similar images and find different sizes of an image. Google's "Search by Image" also proposes descriptions of the images if the system feels confident about them based on similar images found in the search index or presents some visually similar images which –even in worst case, when nothing closely related was found– are at least similar in color.

[13] `http://www.bing.com/images`



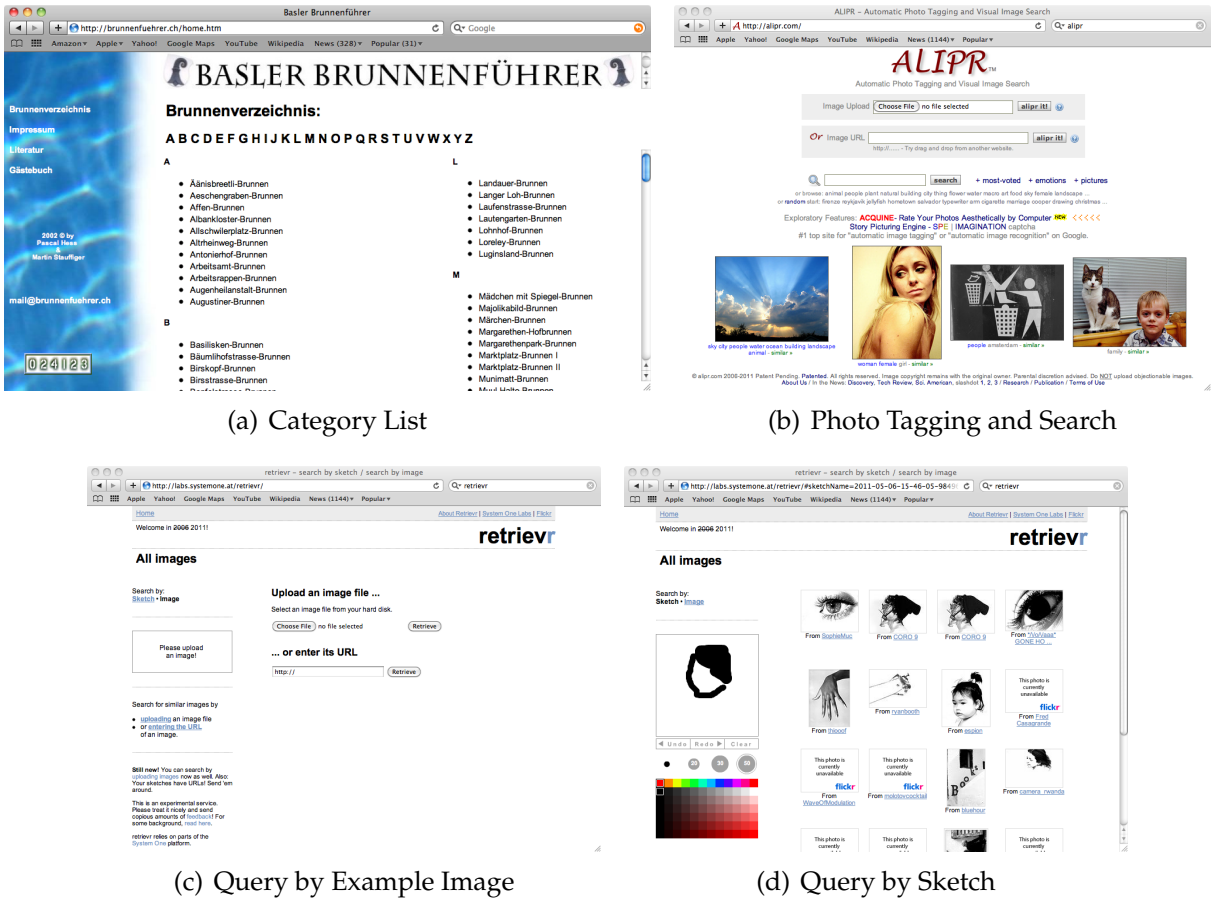
(a) Flickr Explore (b) Corbis Stock Photos

Figure 6.3: Examples of Search Input Methods II: Two interfaces dedicated to explore the available content by selected topics and representative images.

Such approaches con also be found on Flickr when switching to the Explore view[14] as shown in Figure 6.3(a). On the bottom of the page, tags are organized into a *tag cloud* to highlight more popular tags. Corbis provides curated galleries which are displayed with selected images to attract visitors as shown in Figure 6.3(b).

By clustering the images, a broader overview over the entire content can be delivered even if no or too little category information is present to determine a solid classification of the images. The latter would be essential to provide proper assistance for *Retrieval by Class*. For this, a simple listing of all available classes can be sufficient, from which the user selects the class of interest. Such a simple interface is used on the website of the Basler Brunnenführer[15] as shown in Figure 6.4(a). Similarly, most systems that allow tagging of people in images provide also a list of all tagged people to select the images that contain a person, e.g., inside Apple iPhoto and Aperture, Google Picasa [Google Picasa Help, 2011a] and Picasa Web [Google Picasa Help, 2011b], Flickr [Camp, 2009] and Facebook [Odio, 2010]. More complex classification schemes may also be backed by an ontology, in which the user can navigate and select particular classes. In cases where selecting such classes from a screen

[14] http://www.flickr.com/explore/
[15] http://www.brunnenfuehrer.ch



(a) Category List (b) Photo Tagging and Search

(c) Query by Example Image (d) Query by Sketch

Figure 6.4: Examples of Search Input Methods III: A simple list of available classes (in this case: links to pictures of different fountains and background information) is available in (a). (b) provides the ability to upload an image to let it tag by the system / perform *Image Classification*, but also enables to continue afterwards to retrieve images with the same or related tags – or search for similar images based on perceptual features. Similarity search is also supported for Query by Example by uploading an image in (c) or Query by Sketch by enabling the user to draw in (d).

is inconvenient, also speech commands can be recorded and natural language processing tailored to the ontology be used (cf. [Karanastasi et al., 2006, Binding et al., 2007]).

Figure 6.4(b) shows the ALIPR entry page[16] that allows to upload images / specify the URL of some image on the web to be tagged (with automated recommendations) and search for either related images based on the tags or visually similar images based on on perceptual features.[17]

[16] `http://alipr.com`, ALIPR stands for *Automatic Linguistic Indexing of Pictures – Real Time* [Li and Wang, 2008]

[17] Meaningful automatic tagging of images is considered a hard problem and may remain unsolvable without human assistance [Pavlidis, 2009b]. Many dedicated user interfaces for annotating images –not directly for the purpose of querying– have been proposed in literature, e.g., for annotating images with text labels on the web with segmentation in [Russell et al., 2008] and as a game in [von Ahn and Dabbish, 2004] or on `http://images.google.com/imagelabeler/` and proposed

As already mentioned on page 148, also cameras on modern phones can be used to acquire example images – which can be very convenient as (in particular) smartphones usually can also transmit the image files easily or process them directly on the phone. One popular example smartphone application is Google Goggles[18] that performs *Image Classification* through the use of a smartphone and its camera for a variety of objects including landmarks, famous paintings, books. The underlying "Search by Image" functionality is or will also be soon available on the desktop search of Google [Google Inc., 2011c, Wright, 2011, Singhal, 2011] with the ability to drag'n'drop images, paste the URL, or upload an image for search. Catepix from Xerox[19] is available as an online service for social media as well as a smartphone app to classify the user's pictures into fairly broad categories like people, mountain, nature, animals, cars, urban, monument, food, concert, night scene, winter sports, etc. TagSense goes further by creating a short-lived network with surrounding smartphones to use more available sensors to capture information on when, where, who, and what was photographed, e.g., to automatically tag a shot in which a particular person was dancing [Duke Today, 2011, Qin et al., 2011]. Domain-specific applications are kooaba Visual Search[20] that links from an image to additional information for media like newspapers, magazines, CDs, DVDs, books and Leafsnap[21] that allows to identify species of plants by taking images of their leaves. An earlier version of the latter was called LeafView and was running on Tablet PCs [White et al., 2007]. The product named "ArtDNA" provided by Collectrium[22] is another example that is used at several art exhibitions and galleries to look up information about exhibits by taking pictures with a smartphone; similarly, the J. Paul Getty Museum in Los Angeles has partnered with Google to provide additional information within Goggles [J. Paul Getty Museum, 2011]. Notice, that in almost all the examples, look-ups are performed based on the class information derived from the picture, not based on visual similarity, e.g., Collectrium will rather respond with a message letting the user known that the ArDNA of an transmitted image is not known rather than proposing another image that may look similar. The exception from this list is Google Goggles / Images, which -if it cannot classify the image or in addition to it- shows as a fallback similar images.

6.1.3 Example Interfaces for Images and Sketches as Query Input

Also retrievr[23] allows the user to upload an example image / specify the URL as shown in Figure 6.4(c) to search for similar content based on perceptual features. In contrast to ALIPR, retrievr does not offer any classification of the uploaded images. Instead, it also allows the user to draw a sketch as shown in Figure 6.4(d) to search for similar

as a replacement of text CAPTCHAs to fight against spam and other automated exploits on the Internet in [Morrison et al., 2009, Morrison et al., 2010].

[18] `http://www.google.com/mobile/goggles/`

[19] `https://catepix.services.open.xerox.com`

[20] `http://www.kooaba.com/`

[21] `http://leafsnap.com/`

[22] `http://www.collectrium.com/`

[23] `http://labs.systemone.at/retrievr/`

(a) Tablet PC (b) Interactive Paper (c) Interactive Paper Buttons

Figure 6.5: Novel Input Devices for Sketching Visual Examples: A big drawing area on a Tablet PC can be used for drawing sketches with a stylus in (a). With a digital pen and interactive paper, the user can draw with a (almost) common pen on paper from which the sketch is wirelessly transferred to the application running on the laptop in the background of (b). The ability to turn any printed element on paper into a control, e.g., a button to issue a search command is displayed in (c).

images.[24] As a perceptual feature, multiresolution wavelet decompositions of the color channels [Jacobs et al., 1995] are used.[25] Another user interface to support *Query by Example* will be shown Section 6.2 in Figure 6.7 on page 157 with the added ability to specify several example images.

Query by Example may be helpful, if a very similar example image is available or can easily be generated, e.g., using the camera in a modern cellphone, and the actual task is targeted toward exploiting the information associated with a particular image and/or finding a particular (potentially known) image that differs in some of its aspects from the example; therefore the usage being mainly *representation-oriented*.[26]

Query by Sketch can be of particular interest for *Known Image Search* where the original image might not be available and the metadata that would ease the lookup insufficient or forgotten. For this, providing an appropriate input device can be crucial as it can have great impact on the ability of the user to provide sketches that are good enough to retrieve meaningful results. Figure 6.5 shows some input devices as mentioned already in Chapter 2.3.2 on page 49ff.

Still, depending on the collection content, e.g., very domain-specific collections, sketching without additional assistance may simply ask too much of the user. In such

[24] Before the announcement of Google's "Search by Image" feature [Wright, 2011], Google provided also a feature to search for similar images [Rosenberg et al., 2009, Murphy-Chutorian and Rosenberg, 2009] which differs w.r.t. the input methods that it allows the user only to select images from previous search results based on keywords; the user could not upload new images. A search result will be presented in Figure 6.6(c) on page 156.

[25] The same features with similar interfaces have also been used in the applications imgSeek (`http://www.imgseek.net/`) and digiKam (`http://www.digikam.org/`).

[26] For completely *content-oriented usage*, if there is an example or can be generated easily, there would be no need to perform a search – the example itself is sufficient to end the task successfully.

situations, the user can get additional support from the user interface, e.g., if it provides template objects for which the user will only adjust the spatial layout to formulate the query. [Chang and Fu, 1979, Egenhofer, 1997, Di Sciascio et al., 2002, Liang et al., 2005] describe approaches that focus mainly on spatial relationships expressed in the sketch. In particular in the context of *Themed Search*, this can be helpful as the user might have the concepts that should be present and how they are arranged in mind, but may not be able to draw precisely the appearance of the particular (previously unseen) object in a particular position – which would make retrieval very unreliable due to the enormous deviation when relying only on the sketch to identify the "right" objects to include in the results. Therefore approaches like Sketch2Photo [Chen et al., 2009][27] allow the user to also add textual labels to the sketch. This particular approach aims at creating image montages, therefore not relying on existing images to match the (spatial) query directly. Instead, new images are generated out of many existing images by first selecting candidates that contain the desired object and can be blended nicely with each other and then arranging the objects according to the spatial constraints.

6.2 Result Presentation

After the query has been executed, the results have to be displayed to the user. In digital libraries with text-documents, it is common to present mostly the bibliographic metadata of the item, e.g., the title, author, kind of publication, and (if available) enrich it with an abstract. The simplest possibility is certainly to list the results either as entries in a table or as a more space-efficient list.

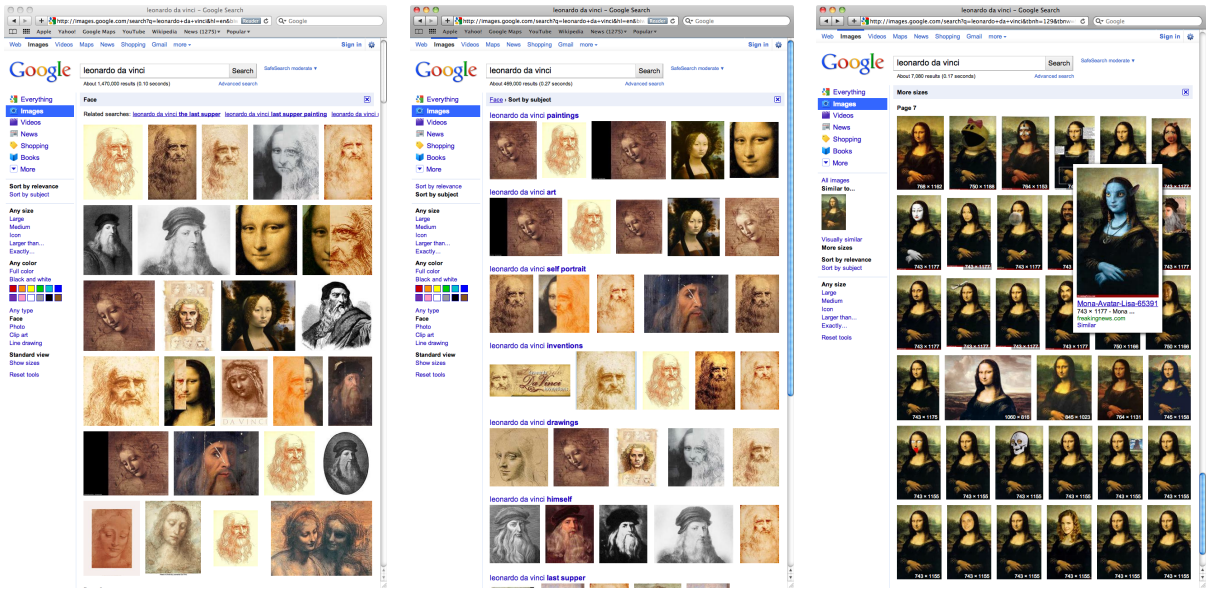
6.2.1 Result Lists

Such lists of results are frequently used for web searches (cf. [Zhang, 2008, pp. 181–183]). For such less curated content like web-searches, short contextual text snippets assist the user – in particular when the search is of informational rather than navigational nature (cf. [Cutrell and Guan, 2007]).[28] For navigational searches, in particular searches for known sites, showing small images of the websites (thumbnails) and "visual snippets" improved the search process [Teevan et al., 2009]. Also other studies showed that even small images can improve web searches [Woodruff et al., 2002, Xue et al., 2008] and Google provides now a feature named Instant Previews [Krishnan, 2010, Google Inc., 2011b].

This tendency to use a visual aid to present even websites to user is naturally much stronger for content that is visual in its nature. Therefore the presentation of individual search results for image content in a list uses thumbnails; scaled-down versions of the

[27] `http://cg.cs.tsinghua.edu.cn/montage/main.htm`

[28] Navigational web searches are searches with the aim to reach a particular website, e.g, finding again a previously visited page or the homepage of a particular company. In contrast, informational web searches aim at acquiring a particular information/fact, e.g., the opening ours of the nearest post office or the place in which Leonarda da Vinci died. Compared to the result usage of image searches, navigational searches are representation-oriented, informational searches are content-oriented.



(a) Images Sort by Relevance (b) Images Sort by Subject (c) Similar Images

Figure 6.6: Examples of Search Result on Google Images: Search for images with keywords "leonardo da vinci" and showing faces; results are sorted by relevance in (a) and by subject in (b). In (c), the Similar Images feature of Google is used to retrieve alternate versions of the Mona Lisa. Notice that this is using "more sizes" to retrieve almost identical images to the query image and the results are page 7, so many closer matches have been presented already before these results. As the mouse was placed over a version inspired by the movie Avatar, this thumbnail is enlarged and some metadata presented.

images. In contrast to text snippets, they can be generated at various sizes and easily organized in rows and columns, which allows to use the available space on screen much more efficiently than one dimensional lists. Depending on the differences between the individual results, already from a glance over the thumbnails users will be able to select which images might be relevant to their search task and which can be discarded / ignored immediately. This is a significant benefit over traditional text-based list results where the user has to read title and snippets to perform the same kind of distinction; it is also the reason why the strategy to browse collections instead of performing searches scales much better with images than with text documents.

Figure 6.6 shows search results of Google Images that use a very compact view with only the thumbnails being presented.[29] By default, the results are ordered based on relevance w.r.t. to the query as shown in Figure 6.6(a). Google also allows to sort the images by subject [Google Inc., 2011a, Tanguay, 2011] as shown in Figure 6.6(b). The ability to browse quickly over even long lists of results leads to a situation in which it is beneficial for the user to return much more results as a "result page" than fits on a

[29] Google Images (`http://images.google.com`) may appear differently on different browsers depending on the supported features, in particular the supported subset of HTML5, as well as the minor differences introduced by the rendering engine of the browser. The screenshots here are presented as examples, not to market or discuss one particular search engine in detail.

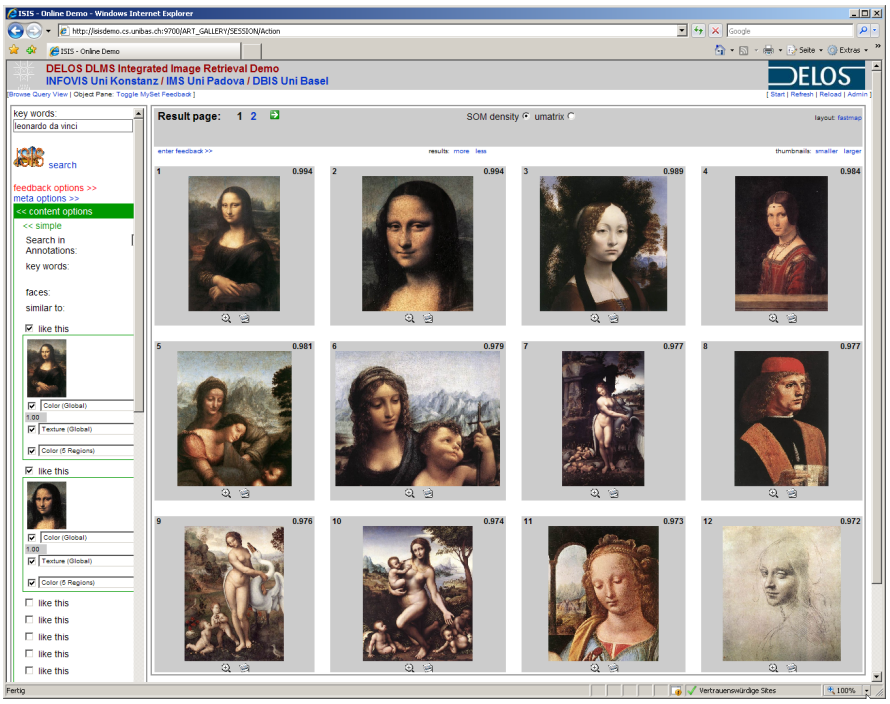


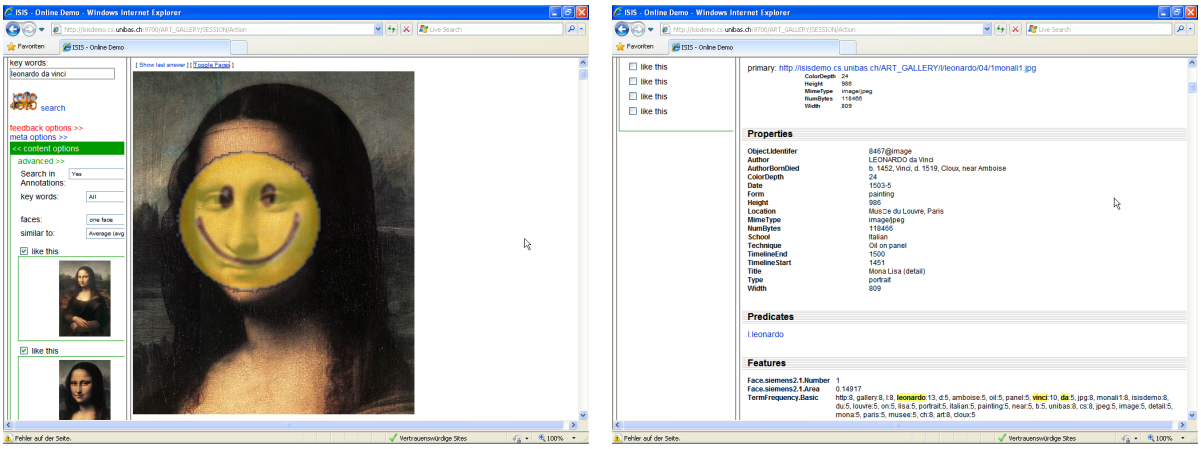
Figure 6.7: Examples of Search Result Visualizations as a List in DelosDLMS [Agosti et al., 2007]: Just displaying the results as thumbnails organized in rows and columns.

the screen and let the user scroll over the many results rather than navigating between several result pages. This can be observed in Figure 6.6(c), where the vertical scrollbar indicates that these are not the top results, but with approximately 30 images per screen and this being labeled "page 7", ranks between 180 and 210. Compared to text searches, such ranks would not be reached frequently by the users as the are commonly very focused only on the top results (cf. [Guan and Cutrell, 2007]).

Whenever needed, the image can be enlarged by showing a bigger version and the associated metadata when the user places the mouse cursor over a thumbnail image or clicks on it. The information offered to the user determines which kinds of result usage are supported: For content-oriented searches, the provided level of detail of the image in full resolution must be sufficient to derive the needed information from the content. For representation-oriented searches, it may be necessary to link to associated information, e.g., for crawled images, link back out to the URL in which the content was stored originally; such information may have been persisted in content management as described for the rich content model in Chapter 4.1.1 on page 86.

Figure 6.7 shows search results as a list in DelosDLMS [Agosti et al., 2007]. The searched collection is the content of the Web Gallery of Art [Web Gallery of Art, 1996][30] which contains over 20'000 reproductions of European paintings and sculptures of the Romanesque, Gothic, Renaissance, Baroque, Neoclassicism, Romanticism periods (1000-1850). The query was for paintings of Leonardo Da Vinci (expressed as a keyword search) that contain a face and are visually similar to two exam-

[30] http://www.wga.hu



(a) Enlarged View (b) Metadata

Figure 6.8: Examples of Search Result Visualizations for Details: When any of thumbnails in the results of Figure 6.7 is clicked, DelosDLMS presents the details associated with it. The image in bigger size is shown in (a) with an overlay to highlight where in the image the face was detected. When the user scrolls down, the metadata is revealed as in (b).

ple images of the Mona Lisa (one of the overall painting, one with a detail of the face)[31]. Similarity –and therefore also the ranking– was computed using the perceptual features color moments [Stricker and Orengo, 1995] and Gabor texture moments [Stricker and Orengo, 1995] on the global image, as well as color moments with five fuzzy regions [Stricker and Dimai, 1996]. As the query images are taken from the collection and match the predicates (keywords, contain faces), the result list also contain the query images. Figure 6.8 shows the detailed view when a result item is selected by clicking on the thumbnail.

6.2.2 Spatial Arrangement of Results based on Pairwise (Dis-)Similarity

In a list presentation, the (dis-)similarity between the query and result is only expressed as a score, but not visualized by itself. Furthermore, the (dis-)similarity between result images amongst each other is not displayed at all. Multidimensional Scaling (MDS) [Zhang, 2008, pp. 143–162] originated in psychology and projects the distances in the high-dimensional feature space into a lower dimensional display space (a process also known as *embedding*), e.g., arrange the search results in 2D to preserve the distances between items as good as possible. Many algorithms have been proposed to achieve such a projection which may have different properties, in particular with regard to computational complexity with increasing number of items to be displayed.

[31] The query images are the rank 1 and 5 in Figure 6.13(b); the images provided with the detail information available at `http://www.wga.hu/frames-e.html?/html/l/leonardo/04/1monali1.html` and `http://www.wga.hu/frames-e.html?/html/l/leonardo/04/1monali.html`.

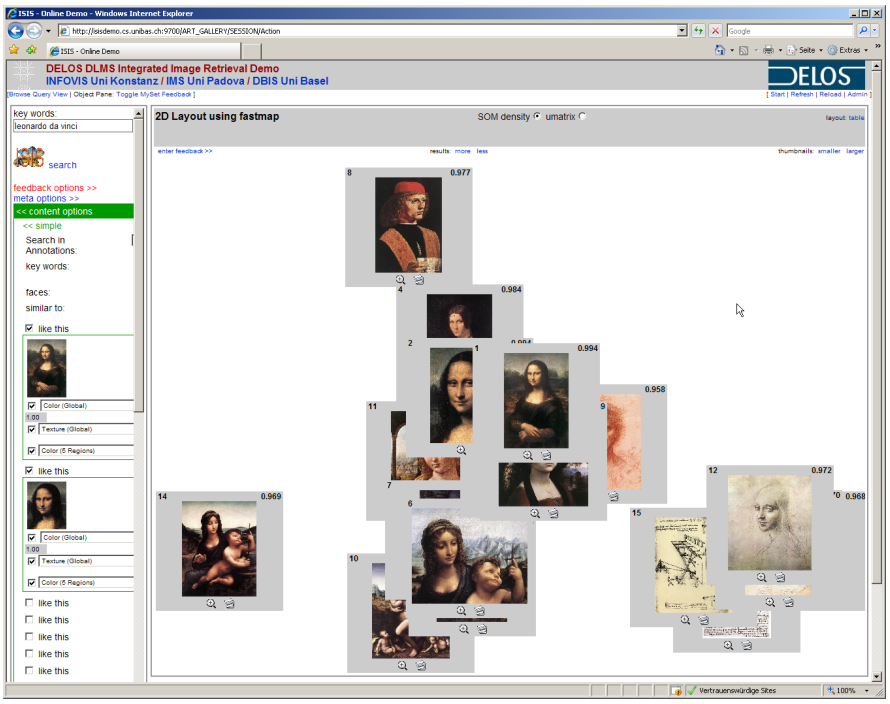


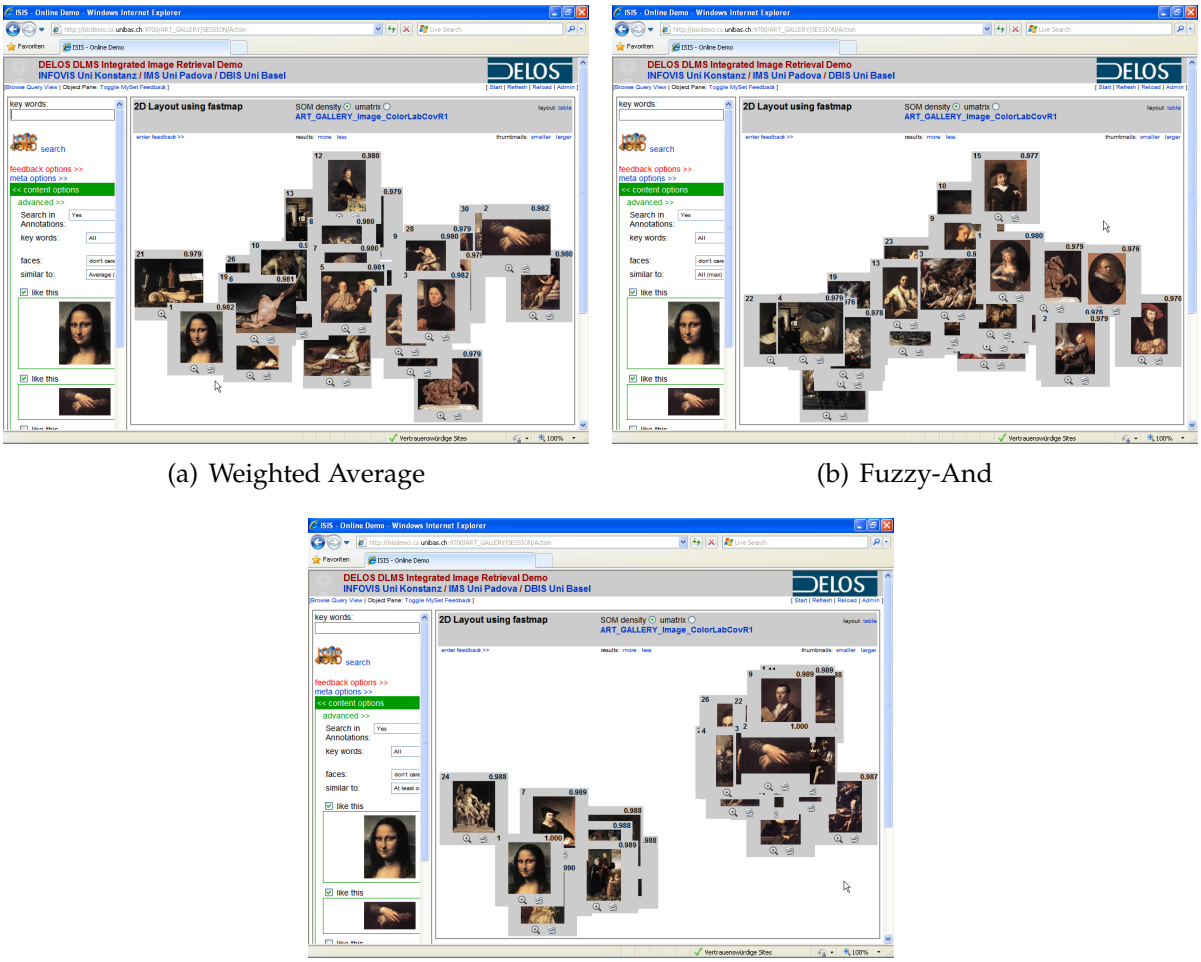
Figure 6.9: Examples of Search Result Visualizations using FastMap in DelosDLMS: Displaying the results spatially organized such that the distances between individual items are projected into 2D using FastMap.

The most popular MDS techniques that have been used for displaying results in content-based image retrieval are FastMap [Faloutsos and Lin, 1995], MetricMap [Wang et al., 1999], and Landmark MDS (LMDS) [de Silva and Tenenbaum, 2002], which have been shown to be all based on same mathematical technique known as the Nyström approximation [Platt, 2005]. Figure 6.9 shows a FastMap visualization of the same search used in Figure 6.7.

To illustrate better the ability to project the high dimensional distances into 2D, Figure 6.10 shows some results for a simplified query with the three different distance aggregation methods described in Chapter 5.3.4 on page 134ff. The query is simplified to be only a similarity search over the entire collection for the 30 nearest neighbors; in other words: no keywords or predicates that images must contain faces are used. Furthermore only a single feature is extracted from the example images and used to determine (dis-)similarity: global color moments. As examples, two detail images of the Mona Lisa are used – one showing the face, the other showing the hands – such that the examples are similar but not the most similar images in the collection.

Figure 6.10(a) shows the weighted average and therefore the results contain mostly images that are between the image of the head of Mona Lisa (ranked as 1, placed to the lower left) and the hands (ranked as 2, placed to the upper right). Notice that even though the examples were ranked in top positions, they do not have a perfect similarity score of one, but slightly below at 0.982.

Figure 6.10(b) shows the results when a Fuzzy-And is applied, therefore determining the 30 nearest neighbors based on the maximum distances. In this case, the pairwise



(a) Weighted Average

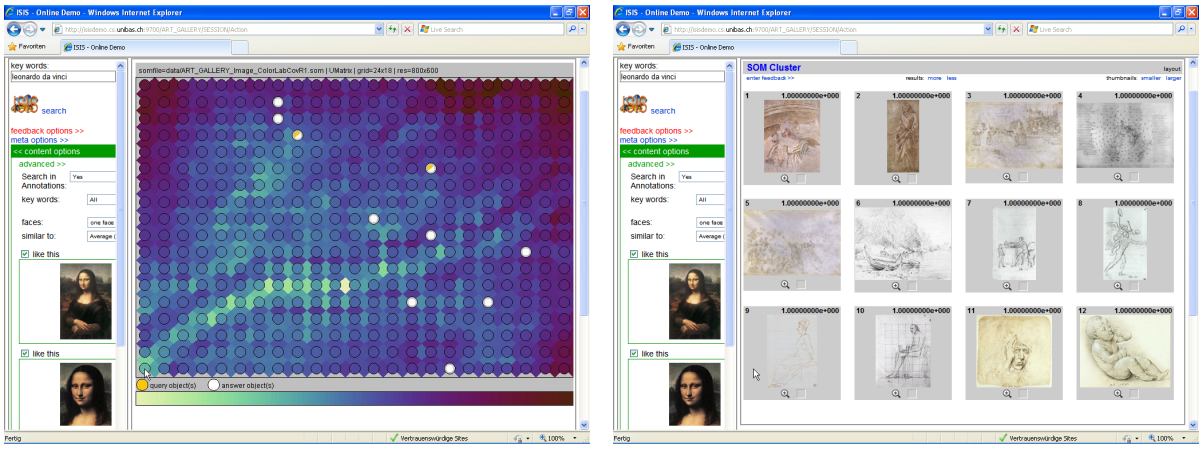
(b) Fuzzy-And

(c) Fuzzy-Or

Figure 6.10: Examples of Search Result Visualizations using FastMap with different functions to compute the aggregated distances: Plain similarity search over the entire collection for global color moments with two detail pictures for the weighted average in (a), similarity to both with Fuzzy-And (maximum distance) in (b), and similarity to either of them with Fuzzy-Or (minimum distance) in (c).

distance between the two query images is greater than the distances to more than 30 other images. This leads to the situation that the query images are not among the search results. Looking at the results from left to right, one can easily see how the color of the results shifts more towards a red or orange tone.

Finally, Figure 6.10(c) shows the results when a Fuzzy-Or is applied, therefore determining the 30 nearest neighbors based on the minimal distances to either of the two query images. This leads to two clusters of results: One is centered around the face of Mona Lisa to the lower left, one is centered around the hands to the upper right. As the distance of any image to itself is zero and the corresponding similarity score 1.0, both query images achieve this perfect score. The found results are so close, that most of the thumbnails overlap and therefore cover each other. By moving the mouse cursor



(a) Clusters in Self-Organizing Map (b) Content of lowest left cluster

Figure 6.11: Examples of Search Result Visualizations overlayed on the Collection in DelosDLMS: The entire collection has been processed to generate a Self-Organizing Map (SOM) presented in (a). Onto this map, the clusters containing search results are overlayed with white dots; yellow dots indicate that a query image is present in a cluster. The SOM is generated for a single feature, in this case the global color moments. By clicking on one of the 24x18 grid nodes, the images inside that cluster get displayed. In (b) shows the content of the lowest left node, which is far away from any of the query and result images and should therefore be highly dissimilar with them; however, images within a cluster should be similar among each other.

over the thumbnails, the user moves the thumbnail under the cursor to foreground and therefore has the ability to see it uncluttered.

Instead of computing an aggregated distance from several features, the authors of [Heesch and Rüger, 2004, Heesch et al., 2006] proposes to combine MDS with Markov clustering to generate NN^k networks. Given a focal image, this approach will determine the a set of nearest neighbors where each element of the set is the 1-nearest neighbor for the focal image for a different metric weighting the distance for the k different perceptual features.

The approach to arrange results in 2D while trying to preserve their distances in high-dimensional feature space can be extended to arrange the entire collection. Instead of displaying individual items (images) of the collection, clusters can be computed that contain images that are similar to each other. Such clustering can be performed using a neural network to two-dimensional grid nodes with Self-Organizing Maps (SOM) [Kohonen, 1990], [Zhang, 2008, pp. 107–125].

Figure 6.11(a) shows the SOM visualization for the entire collection which has been precomputed for the global color moments features. Each grid node contains several images that are similar based on this feature and the color indicates the distance to the neighboring grid node [Schreck et al., 2009]. The search results have been overlayed to the SOM: white dots indicate that at least one of the found results is placed in the grid node; yellow dots indicate that a query image was placed in the node. When clicked, the node will get expanded and all images inside the node are shown to the user. A compact

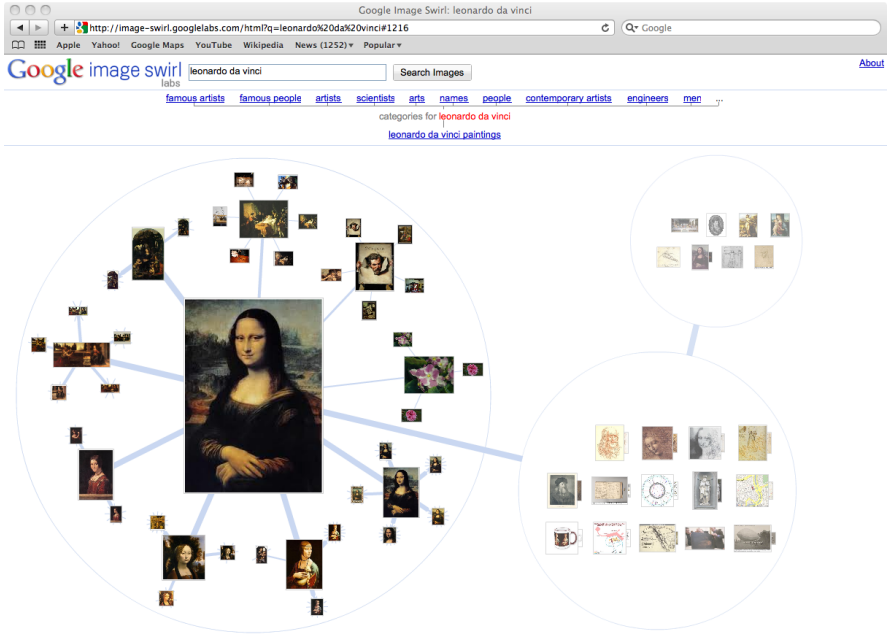


Figure 6.12: Examples of Search Result Visualizations as a Graph: Google Image Swirl allows the user to explore the indexed images based on "visual links" that have been identified using perceptual features at salient keypoints and the VisualRank algorithm.

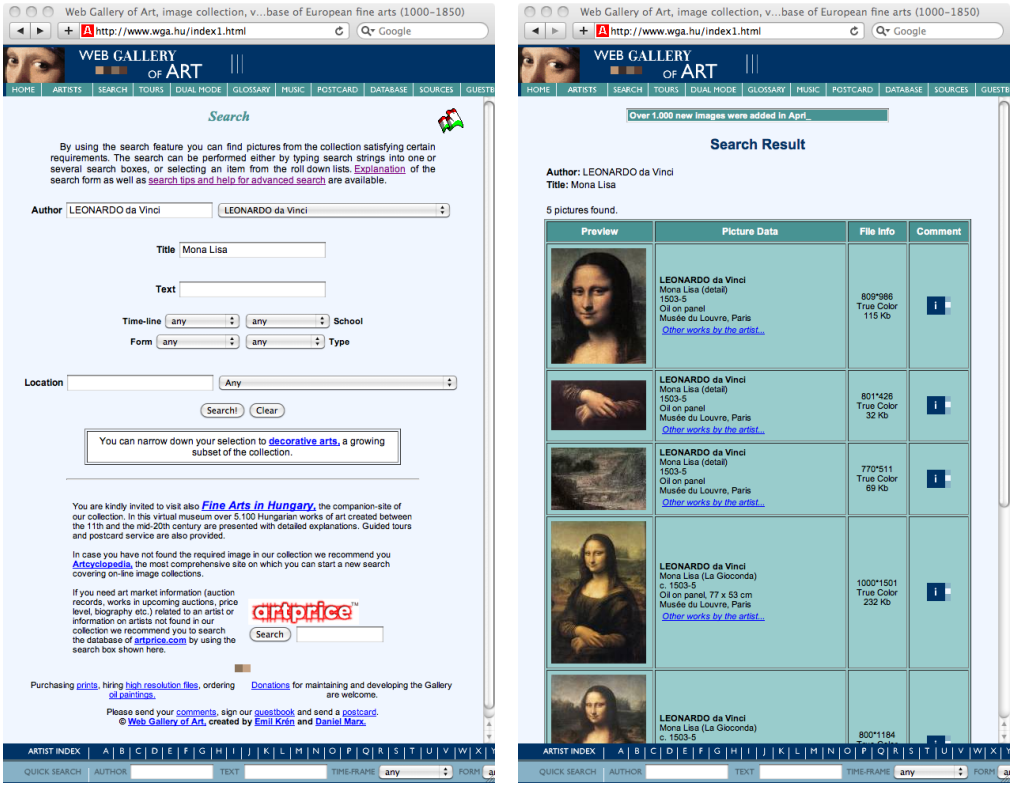
representation like the nodes in the SOM allows to navigate quickly through the entire collection based on the pairwise distances between the images in the collection. To illustrate this, Figure 6.11(b) shows the content of the lowest left grid node; therefore containing only images that are highly dissimilar to any of the query and result images, but similar with each other.

Such pairwise distances can also be used to construct a graph, in particular when salient keypoints are used as features and matching between these is performed. Google Image Swirl [Krishnan, 2009][32] provides an interface to explore such a graph, that can be constructed using SIFT features together with textual information associated with the images as described in [Jing and Baluja, 2008]. Figure 6.12 shows an screenshot of Google Image Swirl.

6.2.3 Metadata-driven Visualizations

So far, the result visualizations focused on presenting the thumbnail images and the similarity of images. In many cases, the metadata associated with an image may also be of great importance to the user and should be displayed. On the other hand, displaying all available metadata can distract the user and force to much scrolling in the results. Figure 6.13 shows the search interface and results as they are provided by the Web Gallery of Art on the website [Web Gallery of Art, 1996].[33]

[32] http://image-swirl.googlelabs.com
[33] http://www.wga.hu/frames-e.html?/search.html



(a) WGA Search (b) WGA Result

Figure 6.13: Examples of Search and Result Visualizations: The Web Gallery of Art provides a search form which focuses on the metadata about the images and the artists that created the paintings and sculptures as shown in (a). The list of search results is displayed in (b).

Zoomable User Interfaces (ZUI, [Reiterer and Büring, 2009]) may be used to prevent information overload while still provide quick access to the information needed by the user. Figure 6.14 shows the MedioVis interface [Grün et al., 2005] to Delos-DLMS [Binding et al., 2007]. In the upper half of the user interface, it provides a HyperGrid [Jetter et al., 2005] with a tabular representation of the metadata in which the user can zoom in and out to adjust the level of detail.

In Figure 6.14, the first row shows even the metadata about the height and width of the image, while the subsequent results have not been expanded to same extent. And if needed, the user can even collapse the results / zoom out such that each item is given just a single line of text To the lower left, the images for Query by Example are displayed. To the lower right, it provides a zoomable HyperScatter view [Gerken et al., 2008] to display each result as a point in a two-dimensional space, where metadata categories define the two axes. To highlight and navigate even in three-dimensional space, a visualization like Figure 6.15 which was generated by the DARE (Drawing Adequate REpresentations) system [Catarci and Santucci, 2001, Binding et al., 2007].

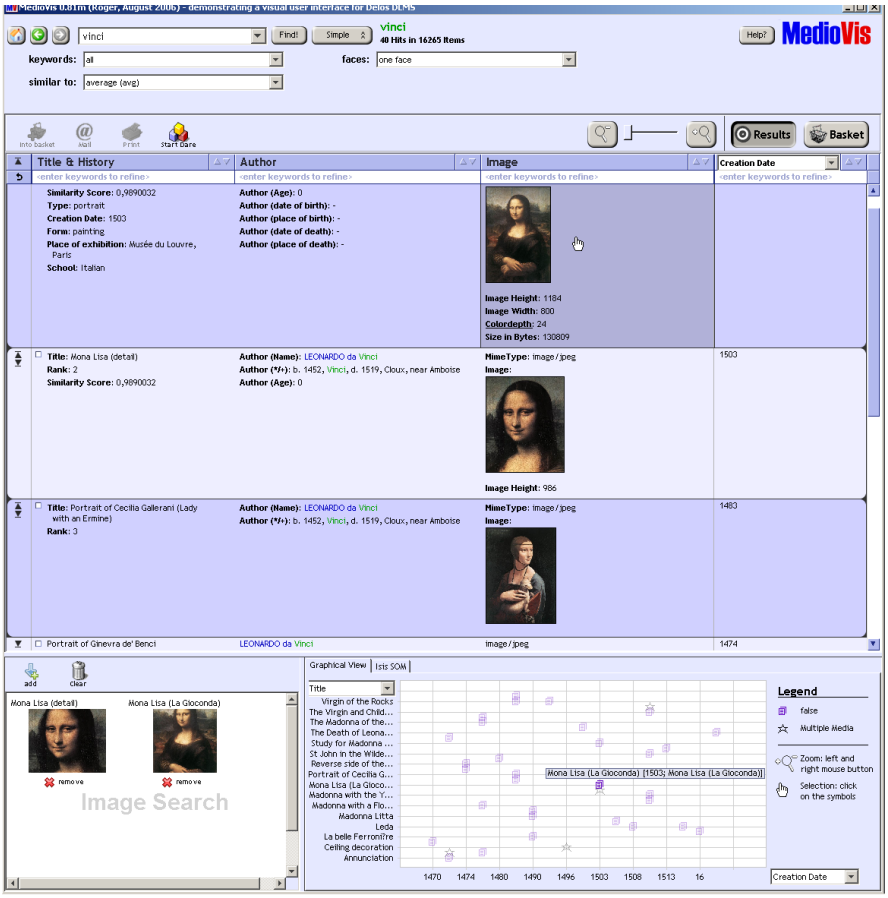


Figure 6.14: Examples of Search Result Visualizations in a zoomable interface: MedioVis

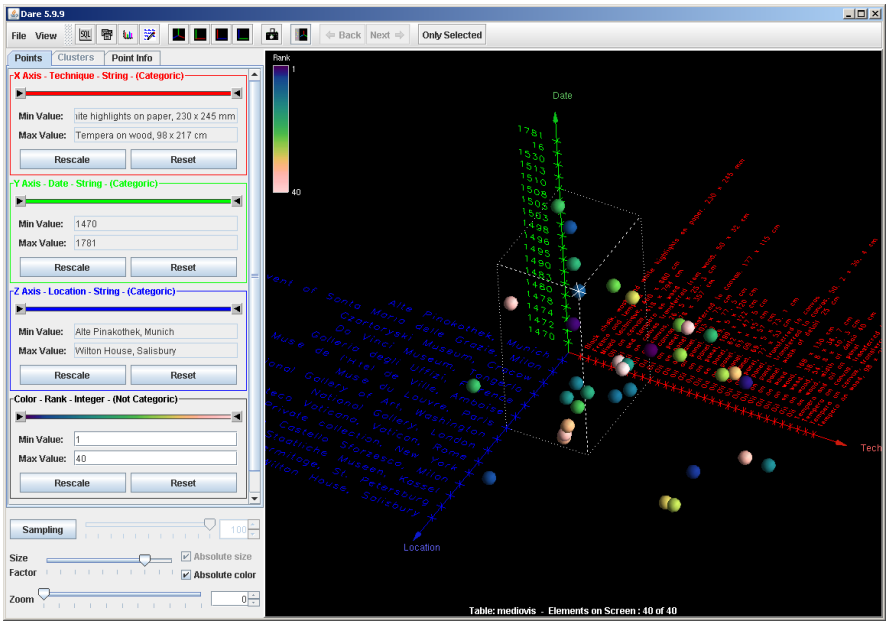


Figure 6.15: Examples of Search Result Visualizations based on Metadata: DARE

6.3 (Re-)Adjusting the Query

From the presentation of the results, relevance feedback provides mechanisms to adjust the initial query formulation.[34] This idea was first used in text retrieval, e.g., in [Salton and Buckley, 1990, Harman, 1992], but has also been applied to CBIR: Common approaches allow the user to give positive or negative feedback on individual images in the result list, e.g., in [Rui et al., 1997]. [Picard, 1995] highlights the potential of such an approach to learn the subjectivity that is inherent in particular in themed image searches. Relevance feedback Figure 6.16 on page 166 illustrates how is conceptually embedded in the search process.[35] There are several different strategies how relevance feedback can be applied, in particular Query Point Movement, Query Reweighting, and Query Expansion (cf. [Harman, 1992], [Ortega-Binderberger and Mehrotra, 2004], [Weber, 2001, pp. 11–14]).

6.3.1 Query Point Movement

Inspired by Rocchio's approach for relevance feedback in the Vector Space Model for text retrieval (cf. [Rocchio, 1971], [Manning et al., 2009, pp. 177–183]), many systems applied this technique also for content-based image retrieval, e.g., in [Rui et al., 1997, Müller et al., 2000].[36] With the background from text retrieval where each entry in the query vector corresponds to a single term, it assumes that there is a single query vector: New relevant terms are added by assigning non-zero, positive numbers to individual terms in the vector. This adjusting of the query vector can be performed automatically as soon as the user has reviewed some documents and marked them either as relevant or irrelevant. Figure 6.17(a) shows an example in a two-dimensional feature space: The query consists of a single point q_0 for which a 5-nearest neighbor search is performed with the Euclidean distance as distance function as described and illustrated in Chapter 5.4 on page 138. Out of these five results, three have been marked as relevant (c, e, f) and two as not relevant (h, k) by the user.

Let Pos be the subset of documents from the initial result Res_0 for the initial query q_0 that the user marked as good results and Neg the results that the user marked as

[34] As the concept of "relevance" is essential for relevance feedback, this section will reuse whenever it is appropriate the notation introduced in Chapter 2.1 on page 20, in particular the sets of retrieved documents Res and relevant document Rel.

[35] Notice, that for some search applications in text retrieval, even blindly assuming the top k documents would be relevant and performing relevance feedback just as if the user had given positive feedback on them (*pseudo relevance feedback*) improved the search results, but can be quite problematic in others [Manning et al., 2009, p. 187]. As pseudo relevance feedback only simulates the user feedback and does not provide any additional possibility to interact with the system, it will not be further elaborated here.

[36] Rocchio's approach is certainly not the only approach for query point movement, but it's the most common and therefore is discussed in detail here. A slightly different approach is used for instance in [Chatzichristofis and Boutalis, 2010] in which Automatic Relevance Feedback (ARF) is performed not directly on the entire feature space –as this is distorted by quantization artifacts for the features used in the approach–, but moves the query point exploiting a Self-Organizing Map (SOM) as has been introduced in Section 6.2.2 on page 161 for visualizing the results.

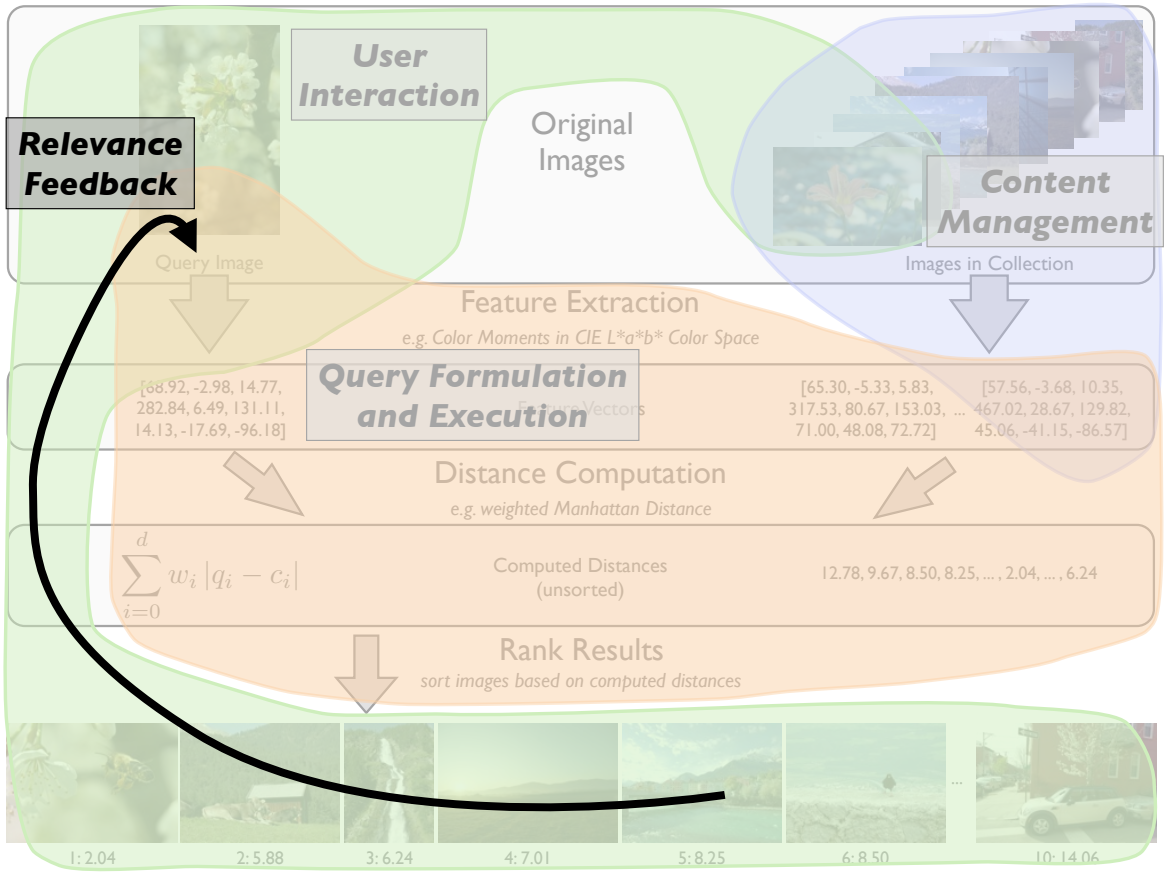


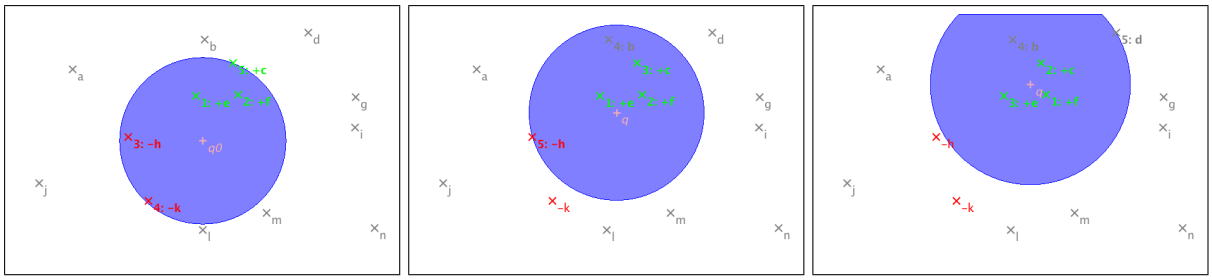
Figure 6.16: Illustration of Relevance Feedback in Search Process: By user the user interacting with the search results, the original query gets reformulated.

inappropriate.[37] The new query q_1 is then computed using Equation (6.1), which adds to the original query vector the average vector computed from all positive examples and subtracts the average vector of all negative examples:

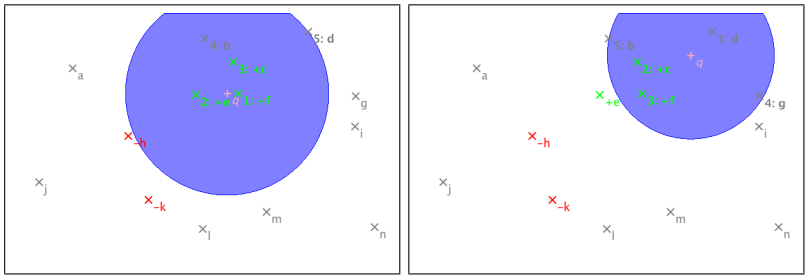
$$q_1 = \alpha q_0 + \beta \frac{\sum_{p \in Pos} \phi(p)}{|Pos|} - \gamma \frac{\sum_{n \in Neg} \phi(n)}{|Neg|} \tag{6.1}$$

The parameters α, β, and γ define how strong the impact of the original query of the user, the positive examples, and the negative examples should be. This formula has initially been used for text retrieval in the Vector Space Model. In the original context, the vectors for each document as well as the query contain the term frequencies and similarity is computed with the cosine measure, which calculates the angle between vectors. For this reason, the length of any vector is not important, only the direction is. The cosine measure therefore leads to an invariance towards the documents length. For text retrieval, this is commonly considered a highly desirable property as the query vector usually contains only very few words and retrieved documents are expected to be much longer – otherwise short documents would appear more similar to queries

[37] Notice, that the user does not have to rate *all* results in Res_0 in order to continue the search; in other words, it is not needed to wait until $Pos \cup Neg = Res_0$.



(a) Res_0 (b) $\beta = 0.5, \gamma = 0.0$ (c) $\beta = 1.0, \gamma = 0.0$



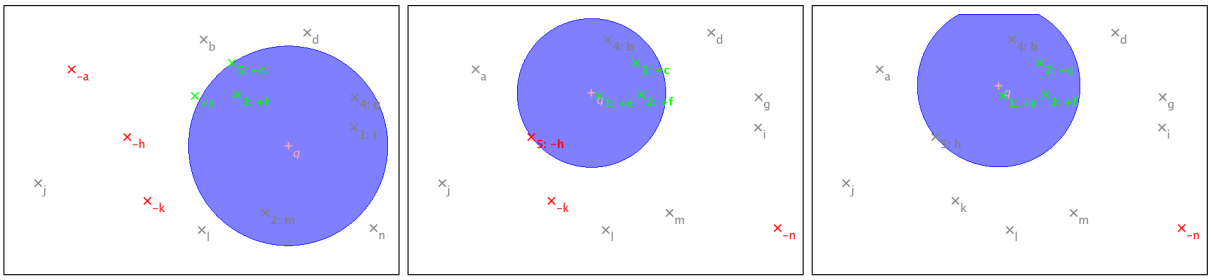
(d) $\beta = 0.75, \gamma = 0.15$ (e) $\beta = 0.0, \gamma = 1.0$

Figure 6.17: Illustrations of Query Point Movement using Rocchio's approach [Rocchio, 1971] for 5-nearest neighbor retrieval with Euclidean distance and $\alpha = 1$. (a) shows the initial result Res_0, which is equivalent to setting $\beta = \gamma = 0$. Positive results are printed in green and marked with a "+" symbol before their label, negative examples in red and marked with "-". (b)–(e) show particular combination of β and γ.

than longer documents. Thus, the values in Equation (6.1) can be chosen rather freely although they affect also the length of the query vector; preference should be given to greater values for α when the user has not given plenty of feedback yet.

In contrast, for content-based image retrieval, the query itself is considered an image (Query by Example) or a sketch (Query by Sketch) and therefore doesn't have a general bias comparable to the low number of terms in text retrieval. Using a distance measures as described in Chapter 5.3 on page 126ff, usually the retrieved documents are defined by some area in the feature space, therefore the direction and length of the feature vector is important. As a consequence, the parameters cannot be chosen freely: [Ortega-Binderberger and Mehrotra, 2004, p. 538] states the requirement, that the sum of the parameters should be 1. An alternative could be to always set $\alpha = 1$ and instead of applying the average vectors directly, compute the difference from q_0 and therefore the direction in which the examples lie w.r.t. to q_1. [Manning et al., 2009, p. 183] mentions $\alpha = 1.0$, $\beta = 0.75$, and $\gamma = 0.15$ as reasonable values.

Figure 6.17 shows various results for different values of β and γ with a fixed value of $\alpha = 1.0$. As illustrated in Figure 6.17(c) and Figure 6.17(d) both, setting $\beta = 1.00$ and $\gamma = 0.0$ (and therefore shifting the query completely to the center of positive feedback) or $\beta = 0.75$ and $\gamma = 0.15$ (using the values mentioned in [Manning et al., 2009, p. 183]) will result in this example in returning all the documents with positive feedback ($Pos = \{c, e, f\}$) and the points b and d as five nearest neighbors. Just relying on the negative



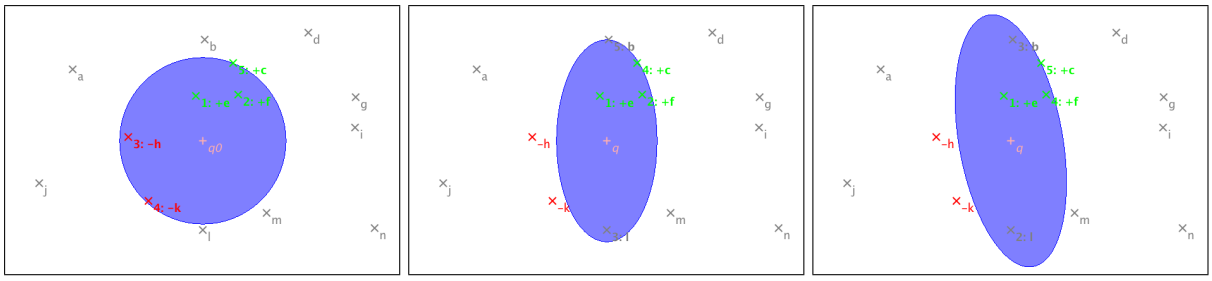
(a) $\beta = 0.0, \gamma = 1.0, -a$ (b) $\beta = 0.0, \gamma = 1.0 - n$ (c) $\beta = 0.75, \gamma = 0.15$

Figure 6.18: Illustrations of Query Point Movement using Rocchio's approach [Rocchio, 1971] with negative Feedback. (a) and (b) show the same search as in Figure 6.17(e) and therefore use *only* the negative feedback, this time with additional feedback on the documents a and n, respectively. (c) shows the same search as in Figure 6.17(d), therefore giving significantly stronger emphasize on the positive feedback than on the negative. However, when only the negative feedback is altered, in this case the negative feedback on n and k is removed and the only negative feedback is therefore on n, the result changes significantly.

feedback might be dangerous: as shown in Figure 6.17(e) for this particular example, when only the negative feedback is considered, the known good result e is now longer part of the result.

To further elaborate on this aspect, Figure 6.18(a) and (b) show the impact of adding just one non-relevant document to Neg. As one would expect for real searches with an appropriate feature for the search, irrelevant document can be found anywhere in feature space except for the desired area of positive results, even very far away. By adding such negative feedback on a as shown in Figure 6.18(a) and n in Figure 6.18(b), the results change significantly. In the latter case, the result even includes document h for which the user has given negative feedback ($h \in Neg$) as the direction of negative results is not very homogenous. This strong impact of negative feedback on images which are far away can even be observed when the positive feedback is considered and given greater importance than the negative feedback as shown in Figure 6.18(c): It uses the same parameters and same positive feedback as in Figure 6.17(d), only the negative feedback was set to $Neg = \{n\}$.

One may argue that in realistic searches, the results that will get presented to the user will not be so "far off" and therefore the user will not be given the chance to rate such very bad examples. However, in early iterations of the search, the user may simply not be "close enough" to find good examples and in particular when a k nearest neighbor search incorporates also a predicate for filtering as described in Chapter 5.4.1 on page 140, the results may be spread heavily over the entire feature space. Also realistic experiments showed that "too much negative feedback can destroy the query" [Müller et al., 2000]. Therefore many retrieval systems use only positive feedback ($\gamma = 0.0$) [Manning et al., 2009, p. 183]. In order to let the user not spend time in vain on giving such negative feedback and also have more screen available to show results that are relevant, the system may turn the negative feedback into a predicate to



(a) Unweigthed (b) 4:1 (c) 6:1, rotation 10°

Figure 6.19: Illustrations of Query Reweighting for the 5-nearest neighbor retrieval with Euclidean distance. (a) shows again the original result for better comparison. (b) has weights where the horizontal difference has four times greater impact than the vertical difference. (c) has weights where the horizontal difference has six times greater impact than the vertical difference and additionally, a rotation by 10°is performed.

filter out those results, that the user already marked as not helpful. This can be done by using *Neg* as a blacklist by using the following predicate:

$$P_{blacklist}(img) = \begin{cases} \textbf{false} & \text{if } img \in Neg \\ \textbf{true} & \text{otherwise} \end{cases} \tag{6.2}$$

6.3.2 Query Reweighting

Moving the query point itself is not the only option to modify a query to better adjust to what the user is actually looking for – this can also be achieved by adjusting the area that is retrieved around a fixed query point. For this, the distance measure that is used to identify this area needs to be parameterizable. The parameters are usually weights for the distance functions and by adjusting the weights, the area gets modified. Very prominent examples in literature are [Rui et al., 1998] and [Ishikawa et al., 1998] and Figure 6.19 shows two examples of what these reweighted retrieval schemes might look like.

Before going into the details of the approaches, it is first worthwhile to identify when such approaches can be used in general: This is on one hand the case when multiple features are used with weighted average as described in Chapter 5.3.4 on page 134 as a distance function[38] or when adjusting the weights of individual dimensions inside a feature as described for weighted Minkowski norms in Equation (5.16) on page 130.

In the first case, the weights define the importance of one feature compared to another feature, e.g.,, how important it will be to match the color compared to the texture of the images. As this affects the weights between different features, this can be called *interweights* (cf. [Rui et al., 1998, p. 648]).

[38]In theory, both – multiple features from a single query image or several features from more than one query image – could be applied. However, when multiple query images are used, query expansion as it will be described ion page 171 will frequently be used in practice as an alternative or in combination with query reweighting. For this reason, the discussion will focus mainly on different features from a single query image.

In contrast, when used on the individual dimensions of a single feature, these would be called *intraweights* (cf. [Rui et al., 1998, p. 648]). This could be the case for instance for Color moments as described in Chapter 5.2.1 on page 114 in CIE L*a*b* color space to give less importance on the luminance captured in the moments w.r.t. the L-channel if the user is not that much interested in the images overall brightness (first moment = average luminance) and contrast (second moment = variance). This might occur when the images were taken in uncontrolled settings and the values might have been subject of the image acquisition and image processing parameters rather than the objects in the image. On the other hand, the human perception of images is usually stronger influenced by the luminance than exact color values[39] and therefore the user might be tempted in other situations to put more emphasize on the luminance and therefore increase the weights for these moments.

In both cases, it might be helpful to start with an unbiased query and then learn the weights from the user feedback. Therefore it is important to perform appropriate normalization such that the absolute magnitude of the computed individual distances does not reduce the possibility for the user to evaluate the impact of individual weights as some non-normalized distance score might be so large that it outweighs by far any result achieved by other features / dimensions.[Rui et al., 1998, p. 647] proposes corresponding inter- and intranormalization of weights.

The general approach in query reweighting is to learn weights for the distance function to separate optimally the set of images in *Pos* from *Neg* while maintaining the query center. For this, many different machine learning techniques can be used – within the limitations that are imposed by the used distance function. Figure 6.19(b) shows an example with 2 dimensions of a weighted Euclidean distance function as described in Chapter 5.3.2 on page 129. For this, the weights can only alter the weight of the two dimensions, thus turning the shape of the area from a circle –when weights are equal– to an ellipse with its major and minor axis being parallel to the axes of the (Euclidean) space spanned by the features. [Rui et al., 1998] proposes an approach which corresponds to the this kind of weighted Euclidean distance. In order to allow also rotations of the area as depicted in Figure 6.19(c), applying the weighted Euclidean distance is not sufficient: Either the space has to be transformed, e.g., applying a affine transformation on the points as performed for Figure 6.19(c), or use a distance function that transforms the space internally, e.g., the *Mahalanobis* or *quadratic distance* (cf. [Weber, 2001, pp. 19–22] and [Bartolini et al., 2001]). The use of such *generalized Euclidean distance* for query reweighting, in combination with query point movement has been proposed in [Ishikawa et al., 1998]. A drawback of this approach is, that it requires at least as many relevant objects as feedback as the used features have dimensions to ensure that the used matrix is not underspecified (cf. [Ortega-Binderberger and Mehrotra, 2004, p. 539]. This is not problematic with the example in 2D and very simple / compact perceptual features, but can quickly become a severe limitation when features with more dimensions are used like 64-bin color histograms, color moments with weak spatial constraints (9 dim $\times$ 5 regions = 45 dimensions), SIFT feature descriptors (128 dimensions), or even the image distortion model with size of 32 $\times$ 32 pixels (1024 dimensions).

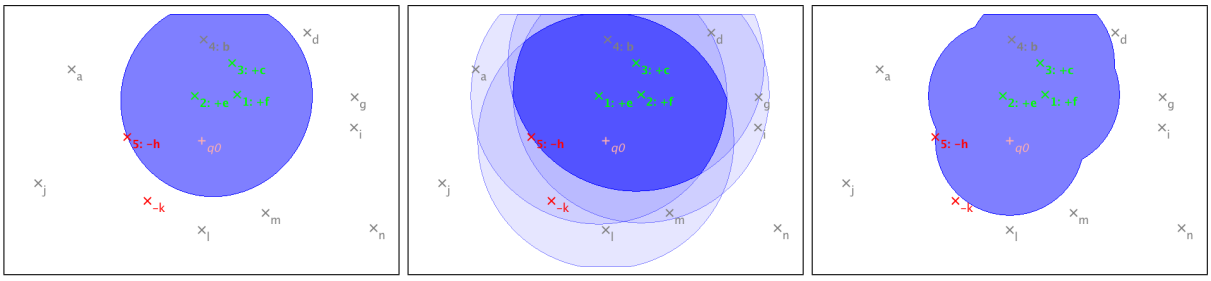
[39] For this reason, lossy image compression formats like JPEG use more bits for the quantization of the luminance than the chromatic information.

An interesting side-aspect of adjusting the intraweights might occur (mainly) when static regions are used to construct a concatenated single feature vector or intraweights are used with a set of features where each feature corresponds to a (segmented) region: In this case, when reweighting appropriately learns the user's preference, it might internally reveal which regions of the image are of importance to the query and which are not – without the user explicitly stating this. [Jing et al., 2004] proposes an approach for exploiting relevance feedback for region-based image retrieval that can learn weights for such a set of features either using a combination of query point movement and query reweighting inspired by [Rui et al., 1998] and [Ishikawa et al., 1998], or using as an alternative machine learning technique a support vector machine (SVM) with a Gaussian kernel.

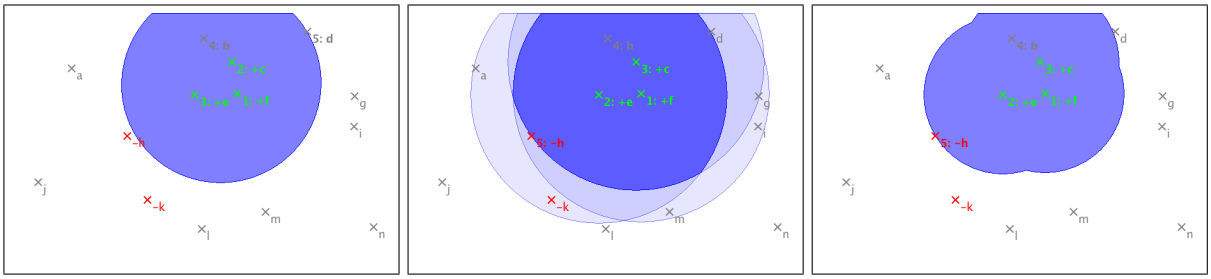
6.3.3 Query Expansion

When dealing with multiple distinguishable objects and images, it might be interesting not only to turn off some irrelevant parts of the original query by giving them a weight of zero, but also to include new clues. This happens implicitly for Rocchio's approach in the Vector Space Model for text retrieval when each dimension corresponds to an individual term, but is no longer present if the feature space is constructed differently. In this case, query expansion has to be implemented differently, e.g., by adding new terms to probabilistic text retrieval as done in [Harman, 1992] or adding new query images to a complex query in content-based image retrieval. The latter has been proposed in [Porkaew et al., 1999b, Porkaew et al., 1999a, Ortega-Binderberger and Mehrotra, 2004].

In detail, several strategies for the expansion have been proposed in these papers: [Porkaew et al., 1999b, pp. 749f] investigated amongst other aspects, which features should be added to the search – and which should be dropped. The idea behind this is to not rely strictly and only on what the user provided as (positive) feedback, but also add a bit of diversity as the search will tend otherwise to be fairly limited to whatever was present in the result set for the initial query. Therefore an approach could have been to add not only features that are very similar to objects in *Pos* to the query (*similar expansion, SE*), but also add some less similar to cover a bigger area of the space by the search (*distant expansion, DE*). However, the evaluation revealed that SE outperformed DE not only in the first iterations of relevance feedback (where this behavior is fairly expectable and DE was even outperformed by query point movement (QPM)), but constantly and therefore also after more iterations (up to the maximum of 5 for the experiment; DE outperformed QPM constantly after 3 iterations). As the approach of query expansion can be combined with query reweighting, the experiments showed that the combination performs best when an appropriate weight strategy is chosen: the *counting weight strategy (CW)* performed best, but was very sensitive to the the size of the considered ranked list, while the second best strategy, *Summation weight strategy (SW)* also performed very well while being more robust. [Ortega-Binderberger and Mehrotra, 2004, p. 542] elaborated in more detail which of the features with positive feedback are added based on the "separation", that is, how well they separate the relevant from the irrelevant objects. This is done computing the average distance and standard deviation from a point



(a) Weighted Average (b) Fuzzy-And (c) Fuzzy-Or



(d) Weighted Average, excl. q_0 (e) Fuzzy-And, excl. q_0 (f) Fuzzy-Or, excl. q_0

Figure 6.20: Illustrations of Query Expansion for the 5-nearest neighbor retrieval with Euclidean distance of Figure 6.17(a) with different distance combining functions as described in Chapter 5.3.4. (a)–(c) show the results when the documents with positive feedback are added to the query. (a)–(c) show the results when the original query q_o is excluded, therefore the new query contains only the documents with positive feedback.

p to each relevant other point (avg^p_{rel} and σ^p_{rel}) as well as to any query result with negative feedback or unrated (both considered *non-relevant*, $avg^p_{non-rel}$ and $\sigma^p_{non-rel}$). The separation sep^p for a single point is p defined as:

$$sep^p = (avg^p_{non-rel} - \sigma^p_{non-rel}) - (avg^p_{rel} - \sigma^p_{rel}) \tag{6.3}$$

The point with the highest non-negative separation sep^p is added to the query, therefore ensuring that there is at least one relevant and one non-relevant standard deviation separation between relevant and non-relevant values and the point. The approach is closely related to a technique used in text retrieval [Carmel et al., 2002].

Some examples for searches of the five nearest neighbors in 2D for the Euclidean distance are presented in Figure 6.20. Figure 6.20(a)–(c) show results when simply all images in *Pos* are added to the original query q_0 – "separation" as defined in [Ortega-Binderberger and Mehrotra, 2004] was not considered. (a)–(c) show the results when the original q_0 is replaced with the images in *Pos*, as the results found and rated positive with the initial query might be better than what the user could provide for the initial query. Notice, that the semantics of the distance combining function is very important:

- If a maximum distance of Equation (5.21) from page 135 is used (which corresponds to Fuzzy-And and is shown in Figure 6.20(b) and (e)), this implies that results are only relevant when they are similar to all the items in the query – and

therefore a single exceptional image in the query can obscure all results. Therefore the initial query should only be preserved if the user could provide a very good example. The user should also give positive feedback *carefully*.[40] As this is fairly counterintuitive for the common understanding of how relevance feedback should learn from the user's feedback instead of the user having to carefully prepare the query, Fuzzy-And is inappropriate for most query expansion approaches.

- The weighted average of Equation (5.20) is less problematic with regard to this aspect, but on the other hand becomes more similar to query point movement the more feedback is provided. In a graphical interpretation, for the Euclidean distance, having a single point results in an area that is a circle. A weighted average with equal weights for two points results in an ellipse. If we add significantly more points, the shape will get increasingly smoothed towards a circle around the (weighted) center of all points. This property is already visible in Figure 6.20(a) and (d). As query point movement can be achieved also with Rocchio's approach in a (usually) less complex manner, but also performed worse than query expansion, weighted average with many query points also appears less than ideal. In the case of [Porkaew et al., 1999b, Ortega-Binderberger and Mehrotra, 2004], the weighted sum of the distances was computed.[41] As points are not only added but also deleted based on the separation criterion and weights being zero, the overall number of points will remain within bounds that still result in a shape that differs significantly from a circle and therefore the approach does not yet get too similar to query point movement.

- The minimum distance from Equation (5.22) (which corresponds to Fuzzy-And and is shown in Figure 6.20(b) and (e)) in contrast to the maximum distance and the weighted average will assure that every point as well as some area around it will remain part of the results. In case of a range search, this means unless the range gets altered (even indirectly by reducing the weight), the initial search results for q_0 will always remain in the refined result sets. Such an adjustment of the range is implicitly performed in a kNN search: For a constant value of k, the area shrinks around any existing point in the query to gain area for any newly added point. This is well in line with the semantics of letting the user label relevant results as each of the relevant results is assured to remain part of new results, but on the other hand will quickly converge into a local minima which may not contain *all* the relevant results that may lay further away from the ones that the user has found so far.

As all these approaches have some unwanted properties, the next paragraphs will focus on the critical aspects.

[40] Keep in mind, that it is very easy to construct queries with Fuzzy-And that do not contain any of the query points as can be seen in Figure 5.15 on page 135, hence none of the images that the user explicitly declared relevant would be considered as results for the new query! Fuzzy-And works best, when each query point adds another new aspect that was not met by any of the other points in the query, e.g., one image has the right color and the other the right shape. But w.r.t. relevance feedback, the user shouldn't have given positive feedback if in fact the individual images did not match all the requirements.

[41] The sum of distances delivers the same results as the average if one ignores the absolute value of the distance scores.

6.3.4 Critical Issues with Negative Feedback due to Suboptimal Initial Query

One important problem that exists with all of the methods proposed for relevance feedback is, that there is a need for sufficiently good initial query. Otherwise, the user cannot provide positive feedback and can add only results to Neg. If only negative feedback is present, this will have the following effects:

- As stated already on page 168, too much negative feedback has shown in experiments in [Müller et al., 2000] "to destroy the query". As seen in Figure 6.18, providing only negative feedback leads to query point movements, which can provide little to no guarantee at all to converge to better results: It would basically be accidental to hit some area in feature space that contains good results.

- For query reweighting, the situation is similar, probably even worse as only the shape of the area is adjusted by altering the distance function, but not the placement of it. If range searches are performed and there is no restriction, e.g., a rule on the weights to sum up to one or some other regularization term that prevents infinitely growing weights on the distances, an highly undesirable but possible outcome of weight learning would be to increase the weights to shrink the area until it does not contain any negative examples anymore. kNN searches are less in danger of this, as the fixed k already imposes the restriction that always $k/|Docs|$ of all objects in space is returned. Yet, if the initial query was not very helpful, negative feedback will only return similar and therefore similarly bad results as the area cannot be moved, only reshaped.

- For query expansion, at least one positive example is needed to expand. When "separation" criterion [Ortega-Binderberger and Mehrotra, 2004, p. 542] as presented in Equation (6.3) is used, avg^p_{rel} and σ^p_{rel} would be undefined. If this fact is ignored and only the *non-rel* part is evaluated, the approach would pick the most extreme points with negative feedback to be added to the query – which might be even less helpful than just adding random results just to move the area somewhere with probably better results.

Hence, none of the proposed relevance feedback techniques can improve results significantly if the initial query results did not contain at least one relevant result.[42]

[42] A novel approach for large-scale negative mining to train *Examplar-SVM* was proposed in [Malisiewicz et al., 2011]. It requires a single positive example – for which the query image itself can be used, and many negative examples. So far, the approach has not been used for relevance feedback and the usage in [Shrivastava et al., 2011] states the speed as the central limitation of the approach as training the SVM at query time requires about 3 minutes on a 200-node cluster, therefore clearly making it not suitable for interactive use in common relevance feedback approaches – at least on current hardware. Moreover, in [Shrivastava et al., 2011] the negative feedback was even skipped by simply sampling random instances from the dataset to analyze what has been named the *data-driven uniqueness* of the query image, therefore bypassing any user interaction to collect feedback.

6.3.5 Critical Issues in Measuring the Benefit of Relevance Feedback

If the initial query is good enough, relevance feedback improves precision and recall as has been shown in many evaluations, including the ones cited in the previous paragraphs. However, some care has to be taken when comparing the numbers for precision and recall with and without relevance feedback.

First, we will analyze the aspect of precision as defined in Equation (2) on page 20: Both, kNN as well as range searches return a finite subset Res of all documents in the collection $Docs$. For kNN searches, $|Res|$ is fixed ($= k$). By removing some irrelevant results from the results, these will get replaced by other results.

To illustrate this effect in its simplest form, imagine implementing relevance feedback purely through blacklisting as the predicate in Equation (6.2): Assume $Neg \neq \emptyset$, any image that would be part of $Res \cap Neg$ gets replaced with some result from $Docs \setminus Neg$. For this replacement, until all relevant images are inside the results ($Rel \subset Res$), the probability is greater than zero that the replacement is a relevant result. Hence, through simple blacklisting, the result cannot get worse and precision can only increase with every iteration until the peak precision is reached. The latter is the case either when all k documents are relevant ($Res \subset Rel$) or the number of relevant document is smaller than k ($Rel \subset Res$).

In range searches the effect is even more immediate: As the range is fixed, blacklisting will shrink the number of results displayed to the user until $Res \subset Rel$ and the precision is 1. Even if this shrinks the number of results returned, it is guaranteed to have no negative impact on recall: Equation (1) on page 20 defines recall only based on the intersection of retrieved and relevant results, hence removing any results that are not inside Rel cannot reduce recall.

Notice that Query Expansion with Query Reweighting does indirectly perform blacklisting, depending on the distance combination function: If the –rather intuitive for the context of relevance feedback– combination function of a Fuzzy-Or is used, any document in Pos has a perfect distance of zero. Therefore any distance weights learned from the feedback will have the effect to shrink or shift the retrieved area in space such that it does not contain the documents with negative feedback. For other approaches, the behavior might be different – but as soon as any of the relevance feedback technique achieves its goal of containing less irrelevant results, the effect is present. E.g., the "seperation" criterion in [Ortega-Binderberger and Mehrotra, 2004] tries to achieve precisely this aim when adding or removing points to/from a query by picking points which are farthest away from the non-relevant points.

For recall as defined in Equation (1) on page 20 from user interaction perspective, one has to consider that with every iteration of relevance feedback that returned a new result, the overall number of images that the user has seen for performing the task has increased. If in the results also the images that the user has rated positive are included (as usually done, otherwise the precision usually would drop rapidly as good results are intentionally removed/blocked), the user has to skip over these result in every new result list. Recall as a set-based measure counts only unique appearances, therefore is not negatively affected by showing the same item several times to the user. Therefore,

we need to count the number of images that the user reviewed as results to achieve a certain recall and include those duplicates in the count, if we want to compare the approach using relevance feedback to other approaches, e.g., by simply increasing k or the range of the search instead of asking the user for relevance feedback. Furthermore, there is always a tradeoff for the user between the time invested in rating images over just reviewing more images. For *extremely* recall-oriented tasks, this means that the user has to invest actually more time until a recall close to 1 is reached, no matter how well the relevance feedback mechanism works, as there is the added overhead to rate images. The only assistance the system can offer is, to make it very simple to mark or save images as good results – which is a general requirement for good implementations of relevance feedback, but could be also be provided by any user interface that tries to assist the user well.

6.3.6 Critical Issues for Relevance Feedback for Image-Related Search Tasks

Due to the impact of the tasks on the benefit of relevance feedback, we will further elaborate this aspect for Image-Related Search Tasks as they have been subject of Chapter 2. For each of the different task input and aims, the possible gain through relevance feedback are significantly different:

- For *Known Image Search* as described in Chapter 2.2.1 on page 25, following the definition, only the known image $\mathfrak{I}$ is relevant in the sense that it satisfies the task aim: $Rel = \{\mathfrak{I}\}$. This, in its strict sense, has the two potential consequences depending on the found result Res_0 of the initial query q_0:

  1. If $\mathfrak{I} \in Res_0$, the image was found and the search task ends successfully. Thus, there will be no need for further iterations for which this feedback could be used.
  2. If $\mathfrak{I} \notin Res_0$, in a strong understanding of relevance, none of the returned results are relevant and only negative feedback can be given. As pointed out in Section 6.3.4 on page 174, none of the relevance feedback techniques performs well when only negative feedback is provided. Instead, it would be a much better strategy to just start over with a new initial query.

  If the criterion on "relevance" gets relaxed, positive feedback could be expressed by the user already for the feeling of "getting near where the known image must be". Also in this situation, it would be advisable to completely drop the initial query and instead continue the search with the newly found, better query images. This is in principal the situation if Rocchio's formula in Equation (6.1) is used with $\alpha = \gamma = 0, \beta = 1$ or query "expansion" is used with dropping the initial query and adding a single new query point. Those approaches work for early iterations, but become problematic if old relevance judgements do not decay over time or much feedback is given in a single round: the user would quickly end up in a

local minimum, which is hard to leave. Therefore it might be better to ease the redefinition of a new query based on the results then hiding this aspect in a relevance feedback mechanism.

From the interface perspective, any previously rated result (positive and negative) should no longer appear in the results of next iterations as the user already has seen them and they have not been the sought image – they would only add distraction instead of helping the user to find the sought item.[43]

- For *Classification* as described in Chapter 2.2.2 on page 32, any relevance feedback basically attempts to train a classifier "on the fly". The general assumption for this according to [Zhou and Huang, 2003, p. 536] based on [Kurita and Kato, 1993] is that every user's need is different and time varying, and therefore a database cannot adopt a fixed clustering structure; the total number of classes and the class membership are not available beforehand, since these are assumed to be user-dependent, and time varying as well.

  For building such the problem-specific classifier from the user's feedback, there exists an initial problem: There will be only little training to build a classifier, as each example will have to be rated by the user. According to [Zhou and Huang, 2003, p. 537] typically the user will provide feedback on less than 20 items per round of interaction, depending upon the user's patience and willingness to cooperate. This is usually a small fraction of all images of the collection; otherwise the technique wouldn't provide significant benefit over alternative strategies. Additionally, due to the initial query used, the training will have a bias towards instances that are close to the initial query; they will therefore not sample well the distribution of all instances in feature space.[44]

[43] There have been a number of publications for the PicHunter system that used known image search (*Target Search* in their terminology) as a test case (cf. [Cox et al., 1998, Cox et al., 1997, Cox et al., 1996, Cox et al., 2000]). The conducted experiments with image collections between 1'500 and 5'000 stock photo images, many searches required between 20 and 80 interactions (= rounds of Baysian relevance feedback), which reduced substantially the number of images that have been seen compared to what one would expect when just browsing the collection. However, with literate computer users with little training on the system, only about half of the searches were successful [Cox et al., 1996, p. 366] and the mean search length of successful searches was 75 iterations. The situation was improved between 28% and 32% with exploiting "hidden annotations" [Cox et al., 1997]. Still, as the images were taken from thematic CDs of which each contained 100 images, an approach which first tries to identify the thematic topic and then focusing just on images from this topic could be an alternative that is likely to perform better, as it is easier for the user grasp why certain images are presented after selecting a particular item. The latter was perceived as a problem with [Cox et al., 1997] when features did not match the users understanding of similarity and by explaining more how the systems measures similarity, results in [Cox et al., 1996] improved significantly.

[44] In statistical machine learning, usually a general assumption is that the data is *independently and identically distributed (IID)* and that the distribution of the training samples matches the distribution that the classifier will be used on. A bias like the one introduced by an initial query to select the first examples on which the user can provide feedback does invalidate the IID assumption. This does not mean that learning is impossible –the IID assumption is not strictly adhered in many other application areas in which statistical machine learning is used successfully–, but it does mean that any assumption and estimation on the behavior of convergence is invalid and also that accuracy of the classifier in general might degrade. Some approaches therefore do not attempt to present in each new iteration the improved results directly,

As a consequence, very accurate results can only be expected for rather simple classification problems, e.g., if the documents in feature space form clusters – out of which one cluster contains all relevant documents. If this feature space has a very high dimensionality, many approaches require at least the same number of training examples as there are dimensions to deliver a stable solution (cf. the *Singularity issue in sample covariance matrix* in [Zhou and Huang, 2003, pp. 540f]). This imposes a limitation on which set of features and feature combinations the user can effectively provide –or will be willing to provide– enough feedback for training the classifier. Notice also, that highly clustered datasets and the attempt to reduce dimensionality do somewhat oppose the idea behind relevance feedback, that the user has a particular information need that cannot easily be met by statically analyzing the dataset. Many approaches that use techniques like Expectation-maximization (EM), e.g., in constructing the features to learn relevant patches in [Carson et al., 2002, Deselaers et al., 2005] are therefore examples that oppose the idea of relevance feedback. As stated in [Zhou and Huang, 2003, p. 541] "so in principle, the rationale of relevance feedback contradicts that of pre-clustering" and therefore the features and approaches that can successfully combined with relevance feedback for creating good classifiers with high accuracy is even further limited.

This finding leads to another problem related to the specific properties of these tasks: When Retrieval by Class is performed, the task ends successfully when *all* instances have been retrieved. Therefore, it is good if the precision can be improved through relevance feedback, but what really matters is the precision at a recall of 1 – when there are no false negatives (= missing relevant documents), how many false positives (= irrelevant documents) are presented to the user and how big was the user's effort. Keep in mind that the attempt of relevance feedback is driven by the idea that the correct classification depends on the feedback of the user as the answer might not be predetermined. In such a setting, it is impossible to be 100% certain that there are no false negatives without actually looking at each and every document that was not included in the retrieved results yet. Hence any attempt to apply Retrieval by Class using relevance feedback can only be as good as the user feels confident in having identified all relevant documents using the system. If the confidence is not high, in particular because the user still feels surprised by some of the results, the remaining fallback solution to succeed in the task is to browse the entire collection – and this would destroy the benefit expected of relevance feedback.

In fairly simple problems where simple clustering can find the classes of interest even without feedback from the user, alternative strategies could even start with presenting the clusters, e.g., using interfaces like [Szekely et al., 2009] or visualizations with spatial arrangements based on pairwise dissimilarities and/or metadata-driven visualizations as presented in Section 6.2 and provide naviga-

but present examples that the classifier is most uncertain, thus using the feedback from the user maximize the information gain rather than maximizing the utility for the user *immediately*. This aspect is described as part of the user model in [Zhou and Huang, 2003, p. 538] and the immediate presentation of better results is named *greedy* while the more long-term oriented assistance in learning is named *cooperative*.

tional help to get all members of the class/cluster. This strategy can be even more effective, when instead of hiding semantic annotations that have been collected as feedback in [Cox et al., 1997] these are kept explicit, stored in content management as user annotations or (curated) class information. Thus, they will be available for future searches just like the information saved from previous relevance feedback sessions in [Bartolini et al., 2001].

Such examples can be found in social networks and photo management software, when people are tagged in pictures. As already described in Chapter 5.1.2 on page 104, some existing software already provides Face Detection algorithms, that also can propose which particular person might be present (Face Identification), based on a model trained from previous labels that the user assigned. These models can get incrementally refined through the feedback that the user gives on subsequent proposed labels. However, for retrieval by person name, these systems usually return in first place the labels that have been assigned and/or verified by the user – as the probability of such labels of being accurate is much higher (essentially certain if the user input wasn't wrong) compared to even very well-trained classifiers, that usually cannot get better than the quality of training data.

- For *Themed Searches* as defined in Chapter 2.2.3 on page 37, the assumptions behind relevance feedback and the actual task of finding images that match the user's preferences fit together very well. Therefore, out of all tasks, relevance feedback can be of greatest help for Themed Searches. The only limitation might exist that the user may want to explore some more the collection to assure that the chosen results are actually the ones that suit best the user's needs. From a systems perspective, this might appear as if the user's mind has changed during the search –which might occur if presented with better alternatives, but is not necessarily the case– and therefore needs to provide some support to "gradually forget" old feedback to provide enough flexibility for continued exploration.[45]

From this analysis, we can see that relevance feedback for Known Image Search and Retrieval by Class is problematic. Relevance feedback is a mechanism that provides means for the user to interact with the result presentation. In order to better support also the problematic tasks, a solution could be to focus more on how the user can interact with and in the query (re-)formulation and therefore close the interaction loop with Section 6.1 on page 145.

Relevance of Individual Regions in Images

In particular in tasks that either try to perform object detection and identification as its main goal or also as a strategy in the context of faceted search, the retrieval is usually not based on images as a whole, but on any kind of regions or keypoints[46]. The feedback can be traced back to the corresponding areas in the query image and adjust weights for the matching tolerance for unknown, irrelevant, or unwanted areas[47]. So far, this has

[45] This particular aspect was already mentioned in [Cox et al., 1996, p. 368] and [Cox et al., 2000, p. 33].
[46] cf. 101 on page 101
[47] cf. Chapter 2.3.3 on page 53

not been integrated that well with relevance feedback approaches which mostly operate on global features (cf. [Zhou and Huang, 2003, p. 542]); much fewer approaches attempt to learn the relevance of individual regions, e.g., as in [Jing et al., 2004].

For learning relevance of individual regions, the system should provide appropriate tools to provide feedback not just on individual images in the result list, but on areas in the result images [48] – for which novel input devices like Tablet PCs as depicted in Figure 6.5(a), devices with touch screens and (to a much lesser extend due to latency in printing) Digital Pen and Paper can be of great help [Springmann and Schuldt, 2008].

Those input devices can also be used to let the user modify the query itself by highlighting regions of interest instead of taking the detour to feed back this information from interactions the user performed on the query results; e.g., just selecting the wanted areas in the query images – which can jointly improve effectiveness and efficiency of searches (cf. [Springmann and Schuldt, 2008]).

6.3.7 Involving the User in Altering the Query Input

With text searches, it is fairly easy for the user to change the query: The user could simply type new words, replace existing words, or remove words from the query. The system can also assist the user by providing search term suggestions, spelling corrections, or even speech input.[49]

In faceted searches and other metadata-centric settings, this is frequently even less of an issue: The user may be able to modify of the query just by selecting / clicking on a different entry in the list of available options. In contrast, for images as query input, usually the user cannot alter the input easily. To modify an existing image to better suit the search task, the user will need powerful tools like photo editing software and the skills and time to use them.

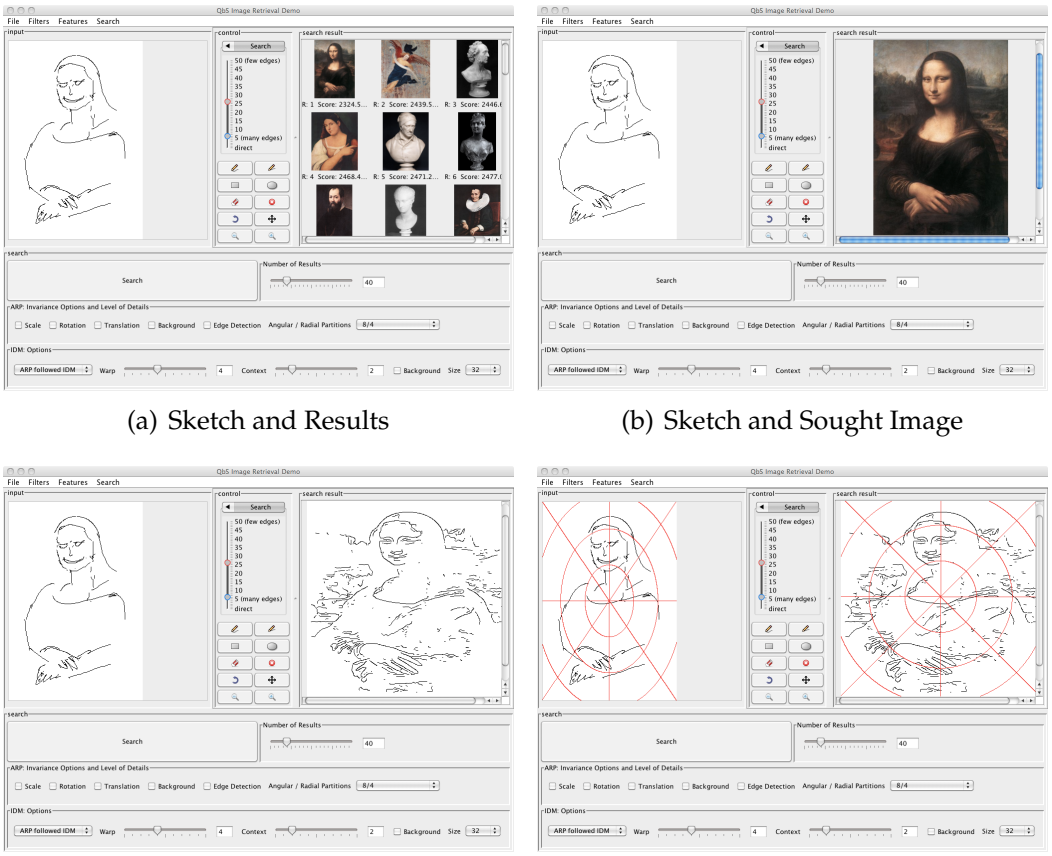
In cases in which the image was generated just for the search, in particular when taken with a webcam or smartphone, it might be easiest to just take another image – in situations where this can improve the results, this commonly requires much less time and effort. As an example: Imagine the user has a printout of an image exists and the user wants to search the digital version. The task therefore is a known image search and can be performed by taking a picture with a smartphone. If the first attempt fails, e.g., because the hand was shaking and the image was to blurred, in order to get good results, starting over by taking the picture a second time takes at most seconds while editing the blurred image would certainly take more time.

In contrast, when sketches are used as query input, recreating a sketch from scratch is much more time consuming. Avoiding the errors that led to the poor results in the

[48] This is similar to what has been proposed as an future extension in [Cox et al., 2000, p. 33]:

> Portions of Selected Image: Yet another independent form of more complex user feedback is to indicate the portion(s) of the image that is (are) similar to the target. The interface can still maintain simplicity by allowing the user to circumscribe relevant portions using the mouse.

[49] So far, speech input has been mostly the domain for mobile devices. In the context of DelosDLMS, also spoken queries for image retrieval has been made available in a desktop environment [Binding et al., 2007]. Google recently announced to also increase their efforts in making the "Voice Search" popular for desktop web and image searches [Wright, 2011, Singhal, 2011].



(a) Sketch and Results

(b) Sketch and Sought Image

(c) Sought Image as Edgemap

(d) Sought Image with ARP Regions

Figure 6.21: Interface for Query by Sketching that allows to alter the query sketch and also present results while sketch can still be changed (a), see one result in same size as sketch (b), see result as edgemap (c), and highlight feature-related information, e.g., angular and radial partitions (d).

first attempt may not be easy, in particular as the drawing skills of the user might be limited. Therefore it not only important to provide the user with appropriate input devices as shown in Figure 6.5, but also to provide the user with functionality to alter the sketch easily. One essential functionality is to remove parts of the sketch that might have been misleading in the search and give the user the ability to focus only on redrawing these parts. This can help with problems like differences in the way edges are treated, mismatch in the level of detail, or differences in color as described in Chapter 2.3.2 on pages 51–52.

In order to help the user to overcome these problems, it might be very helpful to present not only search results as thumbnails and full images, but also in the way the system perceives them in comparison to the sketch. For instance, if the search is performed based on edgemaps, the user should have an easy way to see the edgemaps of search results in similar size as the sketch to be able to adjust the sketch in a similar way. Figure 6.21 shows an interface for query by sketching with the sought image as an egdemap in (c) and region boundaries relevant for using ARP as a feature in (d).

Not all modifications require redrawing the sketch: In particular when the sketch is moved in its position, drawn at a wrong scale, or rotated like described in Chapter 2.3.3 on page 55, these issues can be solved by providing the user same elementary image editing functionality. In Figure 6.21, these tools are available with the toolbar between the sketch and the result view which contain in the lower rows buttons to rotate, move, shrink, and enlarge the sketch. Ideally, this would not only allow the user to alter the sketch as a whole, but also to apply these modifications only to parts of the sketch.

7

Summary of Functional Building Blocks

We have presented are the three main building blocks *Content Management*, *Query Formulation and Execution*, and *User Interaction* and have analyzed in depth the functionality that is necessary to build a digital library with the ability to perform similarity search for images. This analysis consisted of an extensive review of state-of-the-art approaches, based on the Image Task Model (ITM) that was presented in Chapter 2 and revealed several aspects that determine design choices when implementing a digital library depending on the particular needs for the intended kind of content and desired user group and their expected searches. At the same time, it also showed the concepts that are (mostly) ubiquitous and therefore are not only of interest to some particular cases, but can be found in any digital library for visual content with similarity search.

To highlight some of these findings:

- The *Generic Storage Model* as presented in Chapter 4.1.1 can be used to generate rich content models that ideally adapt to the content and usage of any digital library. Such potential adaptations and extensions depending on the expected *Task Input and Aim* as well as the *Result Usage* have been presented in Chapter 4.1.2. The *Generic Storage Model* can also be used to translate a content model to available technology to identify appropriate implementation strategies.

- The concept of *Perceptual Features* is always present in content-based image retrieval, while the choice of particular features and regions does not only depend on the particular content domain, but also on the user's *Task Input and Aim* as pointed out in Chapter 5.2.4. The features, as well as the complementary concept of *Similarity and Distance Measures* depend also on the *Matching Tolerance* that is needed for particular tasks. The execution of search either as kNN search or range search is affected again by the *Task Input and Aim*, see Chapter 5.4.3.

- For *User Interaction*, many of the aspects depend on the kind of content hosted in the collection and in particular the content (or other information) that the user can provide. But depending on the intended *Task Input and Aim*, the query user inter-

face as well as the presentation of results can get adapted. The latter is also influenced significantly by the intended *Result Usage*, as representation-oriented usage commonly require more and other information than just the image content itself as presented in Chapter 6.2.1. Furthermore the task has a tremendous impact on the potential use of relevance feedback techniques as pointed out in Chapter 6.3.6 – to a degree where completely different approaches need to be considered for supporting the user in reformulating (unsuccessful) queries.

Another important outcome of the analysis is the identification of interactions between the building blocks, that need to be considered when implementing a system; e.g., the extraction of features and generation of thumbnails whenever content is added to content management as described in Chapter 4.3.

The next chapters will now focus on particular aspects of implementing (selected) building blocks and show how these extend the state-of-the-art.

Part III

Implementation, Usage, and Evaluation of Individual Building Blocks

8

Introduction to Implementation, Usage, and Evaluation

In the following chapter, we will describe our implementation and usage of individual building blocks. This detailed discussion will therefore focus on concrete building blocks that implement selected functionality. The selection is based on usage scenarios and we include evaluations to proof that the particular implementation of the selected building block is well-suitable for the desired usage scenario and how our approach enhances the state-of-the art.

The structure is identical to Part II, therefore follows the main conceptual building blocks that are involved in the execution of a search process.

Table 8.1: Correspondence of Chapters

Building Block	Conceptual Analysis (Part II)	Implementation and Evaluation (Part III)
Content Management	Chapter 4, page 79	Chapter 9, page 189
Query Formulation and Execution	Chapter 5, page 99	Chapter 10, page 203
User Interaction	Chapter 6, page 145	Chapter 11, page 359
Summary	Chapter 7, page 183	Chapter 12, page 383

The contents of the following chapters will present our contribution to provide building blocks that can be used to build digital library systems. All parts presented in Part III have been used to build complete systems. These systems act as a proof-of-concept and will be presented in more detail in Part IV.

As the selection of building blocks is based on usage scenarios, the description of the scenarios and the requirements that are imposed by them are essential to this part of the thesis. They will determine which criteria are appropriate to measure the suitability and benefit of any particular implementation of a building block.

9

Content Management

This chapter will describe our implementations for Content Management. We chose to have more than a single implementation because each of them has benefits and drawbacks, that we will discuss after we have have introduced them. The chapter therefore covers a broad spectrum of the functional building block of Content Management.

Section 9.1 presents a very flexible approach that is very scalable in terms of the size of the individual content hosted, the size of collections, the number of collections, and the number of users and requests by choosing an architecture that can utilize Grid-resources. In contrast, Section 9.2 presents tailored approaches for benchmark image collections. Section 9.3 will finally discuss the benefits and drawbacks of the different approaches.

9.1 Large-Scale Content Management using Grid Resources

The importance of Content Management becomes most visible when dealing with huge datasets that cannot be handled on individual nodes. Such datasets together with a federated infrastructure of distributed computing resources shared between several institutions are prevalent or even the core of Grid environments as described in [Foster et al., 2001]. Many approaches exist to manage files and content in general in so-called *Data Grids*. Our implementation is particular through its dedication to digital libraries with the inherent requirement not only to allow management of files, but also rich metadata associated with individual raw content together with ability to retrieve information. For this thesis, the documents we are interested in are images and the proposed building blocks must be able to manage not only the images themselves, but also perceptual features that have been extracted and will be used in retrieval. Rather than replacing established solutions for file management in Grid environments, our approach enriches them with functionality required for digital library applications. Yet, the approach is flexible enough to be applied also to other distributed environments to provide scalable management of content.

9.1.1 Motivation

As mentioned in Chapter 1.1, one challenging domain from the perspective of scalability for collection sizes in digital libraries for images are image-sharing websites. They gather up to billions of user-generated images, which can exceed by several orders of magnitude the number of documents in more curated digital libraries.[1]

Another challenging domain are settings, in which mainly automatically or periodically generated images need to be maintained. This is to a certain extend the case for medical imaging (as it will be the subject in Chapter 10.2); but even more prevalent in earth observation scenarios. In this section, we will present our prototypical implementation of content management that provides scalability for such scenarios. It has been used in the context of the DILIGENT project[2], that has been used for earth observations [Candela et al., 2007]. For this, the content is provided by ESA, the European Space Agency and stored on Grid resources spread over multiple sites across Europe using the EGEE infrastructure[3].

Figure 9.1 shows an example document: each of these documents does not exist in isolation; it is linked to metadata which includes information like the geographic area depicted in the image and the settings used for capturing the information. Additional information like image features or detailed measurements may get derived from the image. Documents like these may then be used in reports, which can contain generated parts which are updated whenever new measurements are available. Such reports are called *living documents* [Candela et al., 2007, p. 60]. If we analyze such documents using the Generic Storage Model proposed in Chapter 4.1.1, we can identify some raw content like the satellite image and texts in the report as well as relationships like the features related to the image or reports and the images they use. For the purpose of better structure and ability to retrieve, we can also find properties, e.g., the date when a satellite image was captured and the geospatial coordinates of the points on earth that correspond to the upper left and the lower right corner of image.

9.1.2 Requirements on the Implementation

The earth observation scenario imposes a set of requirements that the implementation of content management will need to satisfy. First of all, it must be able to deal with documents that are linked to other information as shown in the example in Figure 9.1. Such a functionality cannot directly be provided through the functionality that a file system offers. Furthermore, the model must be flexible enough to be adaptable to new data that

[1] Some subsets of 25,000 user-collected images will be used in Chapter 10.3 for evaluation. Such a small and static subset does not fully reflect the true size of such huge and ever-growing collections. It will therefore be subject of Chapter 9.2.

[2] DILIGENT stands for *DIgital Library Infrastructure on Grid ENabled Technology* and was an EU-funded IST project in FP6 from September 2004 – December 2007. It was followed by the D4Science and D4Science-II project in FP7. Project website is available at `http://www.d4science.eu/`.

[3] EGEE stands for *Enabling Grids for E-sciencE* and was a project active from 2004 – 2010. The infrastructure is now supported by the European Grid Infrastructure (EGI). Project websites: `http://www.eu-egee.org/` and `http://www.egi.eu/`.

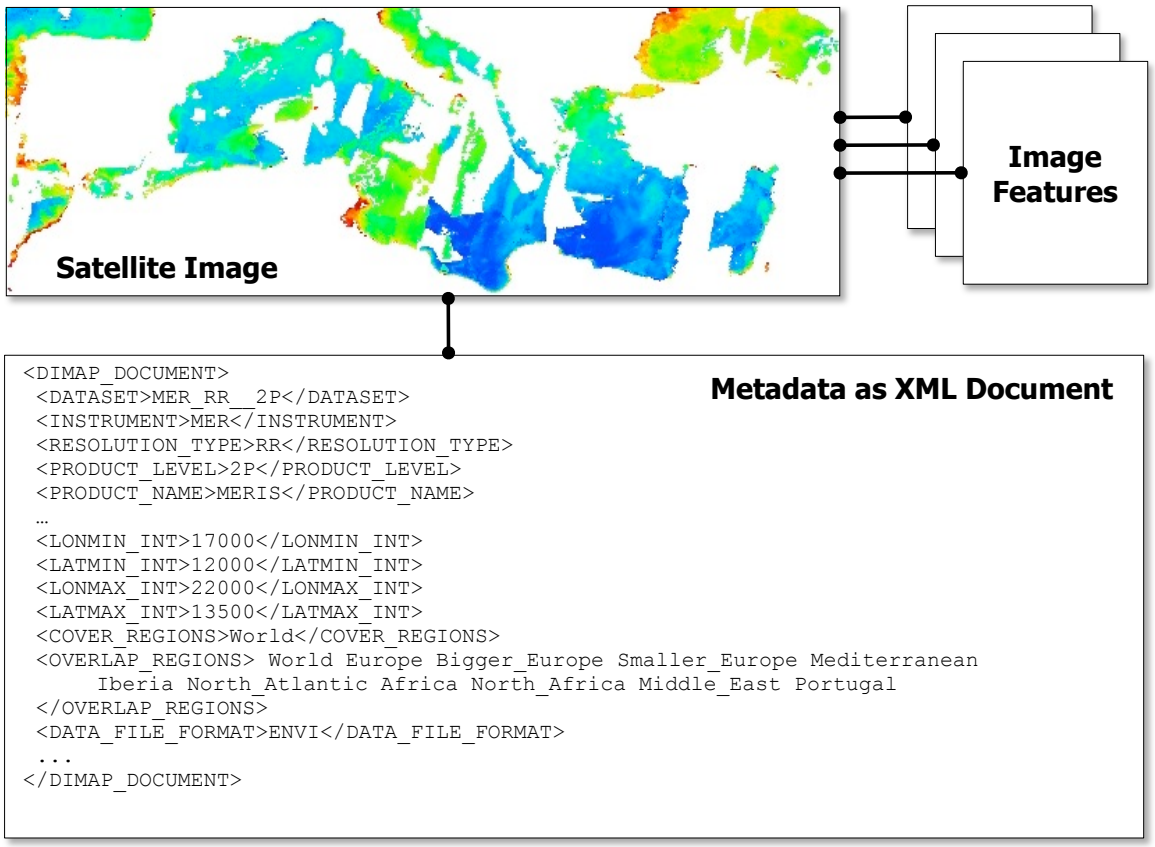


Figure 9.1: Example Document from Earth Observation: Satellite image is linked to a metadata record and image features.

may get generated through new observing instruments, experiments, and processing routines.

Some existing data already is hosted on Grid resources and the Grid infrastructure provides the security mechanisms for authentication and authorization and grouping users and resources to virtual organizations (VO) Therefore any new service for managing content has to be able to access this data and integrate into the Grid infrastructure. This means on one hand, that the implementation has to understand and use existing protocols used in the Grid community. On the other hand, it also means that in order not to duplicate the entire infrastructure, it has to reuse the existing resources of the infrastructure.

Last but not least, the implementation must provide scalability through distribution on many nodes: When new resources are added to the Grid, it must be possible to deploy new service instances to make use of these new nodes without the need to reconfigure the client services that access the content. Fault tolerance is essential in such distributed environment and therefore the possibility to transparently replicate the content and ensure its consistency must be considered from the early stages of design. The deployment must also allow scalability towards smaller scale infrastructures:

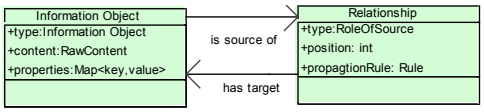


Figure 9.2: UML Class Diagram of DILIGENT Storage Model

For such huge infrastructures that are used in productive Grids, it is almost impossible to provide an infrastructure of identical size for development and testing purposes. The development and testing infrastructures must still be able to use all the features of content management, however, in contrast to the production infrastructure commonly not all features are needed at the same time: For development, it is most important that the API is available all the time and for testing, that the functionality that is being tested is available at the time of testing.

9.1.3 Storage Model and Architecture

The requirements imposed by the nature of the documents being used in the scenario are address through a flexible storage model. Figure 9.2 shows the DILIGENT Storage Model. As mentioned in Chapter 4.4 on page 90, it is closely related to Generic Storage Model. Compared to the Generic Storage Model from Chapter 4.1.1 shown in Figure 4.2(b) it does provide the possibility to assign arbitrary key-value-pairs as properties of information objects, but not for relationships. This has been changed in subsequent iterations of the software.[4] Nevertheless, it does provide references and can specify a type to describe the role of the source of the directed relationship through the "RoleOfSource" attribute, e.g., source object *is-member-of* a collection, source object *is-annotated-by* some annotateion, or source object *has-part* a sub-document. To express a particular ordering, the optional *position* attribute get assigned a numeric value – and therefore allow to specify, for instance, which is the first and which is the second part of a compound document. The model therefore is capable of handling documents like the example presented in Figure 9.1.

Finally, in order to maintain consistency easily, a particular attribute was added to the relationship to be able to define *propagation rules*. This allows to define the behavior of the system when particular information objects get modified / deleted, similar to referential integrity constraints in relational databases. Such propagation rules can be used and were used to implement requirements in a Content Model on top of the Storage Model like "when a document that has parts or annotations gets deleted, all its parts get deleted as well" or "a document has to be member of at least one collection". In DILIGENT, the content model and the storage model are separated into two layers, as shown in Figure 9.3. This is also based on technical considerations.

[4] Cf. the gCube documentation at `https://gcube.wiki.gcube-system.org/gcube/index.php/Storage_Management`; gCube is the name of the software product that was developed during the lifetime of the DILIGENT project and it's successors.

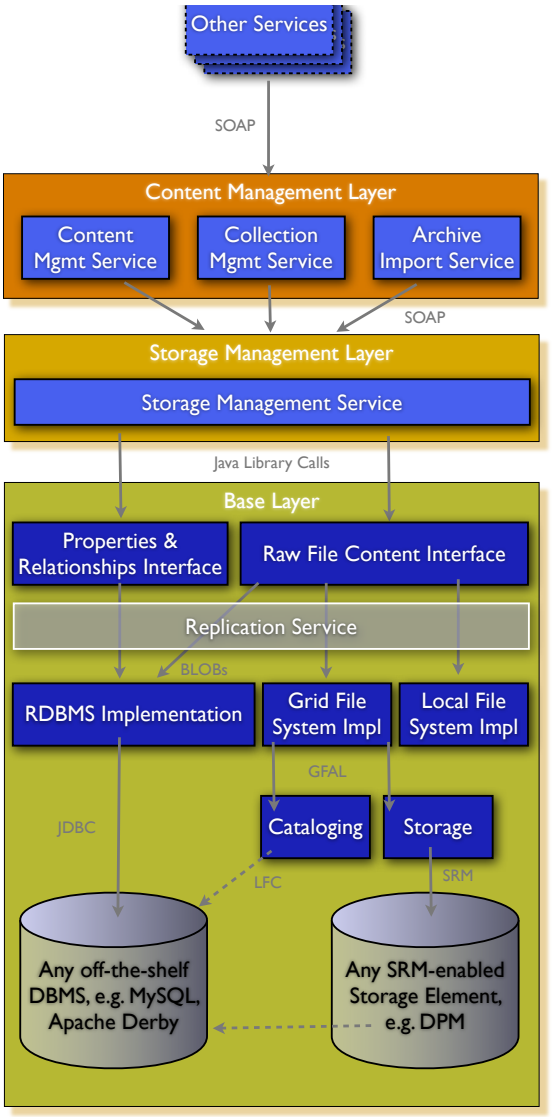


Figure 9.3: Overview of DILIGENT Content & Storage Management Architecture

9.1.4 Technical Framework

DILIGENT has been designed and implemented as a service-oriented architecture (SOA), therefore every major piece of functionality –and in particular every functionality that shall be reusable– is exposed to other services and clients as a web service. DILIGENT is based on Grid-technology, in particular the Globus Toolkit 4 (GT4)[5] in which web services are implemented based on the WS-Resource Framework (WS-RF)[6][OASIS, 2006b, OASIS, 2006a]. The service API is described in WSDL[7] and com-

[5]http://www.globus.org/toolkit/docs/4.0/
[6]http://www.globus.org/wsrf/
[7]WSDL stands for Web Service Description Language, a W3C standard for service description using XML [W3C, 2001]

munication between services uses SOAP[8]. Services are implemented in Java and hosted in a GT4 WS-Core container[9].

The building blocks are therefore implemented as a set of services, where the DILIGENT Storage Model is implemented by a single service, the *Storage Management Service*. It is the only service in the *Storage Layer*, which sits on top of the *Base Layer*. This in turn provides configurable backends to handle *Relationships & Properties* and *Raw Content*. It allows to manage all information according to the Generic Storage Model described in Chapter 4.1.1. The Content Model is implemented inside the *Content Layer* which provides a service to manage the individual content of documents (*Content Management Service*), another service to manage collections as a whole (*Collection Management Service*), and third service to perform bulk imports of harvested collections and archives (*Archive Import Service*). Each collection is published as a WS-Resource, thus allowing easy discovery as well as assisting in security management within the DILIGENT environment. Additionally, notifications can inform services about changes on managed content which is in particular of interest to maintain consistency with indexes for efficient search.

This separation in three different layers allows to reuse the implementation and easily replace parts where needed to adapt to different requirements for different digital libraries. For instance, if any digital library would require a Content Model that significantly differs from the one used in DILIGENT, it would be sufficient to replace the services within the content layer. It also allows to let other services use the storage management layer to add functionality: In the context of DILIGENT, metadata management services and annotation services use the storage management service as backend. The metadata management services in DILIGENT enrich the digital library functionality for documents by allowing transformations and searches on the metadata formats. The annotation services further enrich this functionality by allowing the users to add annotations of document as a whole or in parts. If, on the other hand, a technical different framework would be needed that is not based on WS-RF, also an adaptation of the storage layer would become necessary – but not of the base layer, as long as a binding to Java is still available. For adding support for different backends, it is not necessary to change the service implementations; these changes only affect the base layer.

As a backend for managing relationships and properties, we developed an implementation for relational database management systems that uses JDBC[10] and has mainly been used with MySQL and Apache Derby instances.[11]

[8]SOAP stands for Simple Object Access Protocol, a protocol based on XML messages which are commonly exchanged over HTTP(S) [W3C, 2000]

[9]`http://globus.org/toolkit/docs/4.0/common/javawscore/`, an open-source WS-RF-compliant web service container derived from Apache Axis (`http://axis.apache.org/axis/`)

[10]Java Database Connectivity (`http://www.oracle.com/technetwork/java/overview-141217.html`); provides a call-level API for SQL-based database access

[11]MySQL (`http://www.mysql.com/`) is an open-source database system that is used in a number of software products for the Grid-infrastructure, in particular the LCG File Catalog (LFC) and Disk Pool Manager (DPM), therefore in many cases is already installed on nodes handling storage for Grid-installations.
Apache Derby (`http://db.apache.org/derby/`) is a lightweight open-source DBMS entirely written in Java, which can easily be deployed on any node that provides a Java Runtime Environment; therefore making the deployment in particular easy for development purposes and testing of services that are

As a backend for raw content, the same implementation for relational DBMS can be used – in which case the content will be treated as binary large objects (BLOBs). However, as a dedicated implementation to exploit the vast set of resources available on the Grid, we also implemented another backend that uses the GFAL[12] which can be used for both, cataloging via LFC[13] and storage on an SRM-enabled Storage Element [Open Grid Forum, 2008] like DPM.[14] A third implementation simply stores the raw content to the local filesystem on the node hosting the service. Slightly different from the storage of raw content is its transfer, for which we implemented some additional handlers to support popular protocols like HTTP, HTTPS, FTP, GridFTP, a simple way to include the Base64-encoded content inside the SOAP messages, as well as the ResultSet framework used in DILIGENT [Simeoni et al., 2007a].

The use of existing technology used in Grid environments allows to integrate the digital library building blocks into the existing Grid infrastructure and therefore addresses the another requirement described in Section 9.1.2.

The framework also makes it easy to adapt the system to the current development trend towards cloud solutions, e.g., by implementing management of relationships and properties in a document-centric NoSQL database system like CouchDB [15] or MongoDB[16], or using a commercial offer like Amazon SimpleDB[17] or Google AppEngine Datastore[18]. For raw content, the equivalent choices would probably be HDFS from Apache Hadoop[19], Google GFS [Ghemawat et al., 2003], and Amazon Simple Storage Service (S3)[20].

9.1.5 Deployment

The last requirement described in Section 9.1.2 deals with scalability and the deployment of services is one key concern to achieve scalability. The architecture and framework that we use for our implementation of the building blocks allows to configure and deploy the services of the digital library system according to any need that will be experienced in productive, development, or testing infrastructure.

For instance, it is possible to deploy all services with a single instance on the same node with Apache Derby as a lightweight DBMS in embedded mode and using the local filesystem for raw content. This keeps network traffic between nodes low, which is in particular helpful to reduce latencies in slow networks where each additional message

based on content management without affecting the productive environment.
The usage of other relational DBMS is possible as long as a JDBC driver is available.

[12] Grid File Access Library (`http://wiki.egee-see.org/index.php/Data_Access_with_GFAL`)

[13] LCG File Catalog (`https://twiki.cern.ch/twiki/bin/view/LCG/LfcGeneralDescription`)

[14] Disk Pool Manager (`https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm`)

[15] `http://couchdb.apache.org`

[16] `http://www.mongodb.org/`

[17] http://aws.amazon.com/simpledb/

[18] http://code.google.com/appengine/

[19] `http://hadoop.apache.org/hdfs/`

[20] http://aws.amazon.com/s3/

over the network between services harms the end user impression. Such a setup can also be sufficient for small installations or in development and testing infrastructures, where nodes may run in virtual machines and co-deployment of services in the same container on the same node reduces the memory consumption significantly.[21]

For handling heavier loads, the deployment can be adapted. If there are, for instance, distinct user groups that work on different collections, the easiest solution certainly is to deploy an exclusive instance for each user community and isolate them into virtual organizations (VO) to partition the load at this level.If the load on particular collections still is too high, several instances of the Content Management Layer can handle the request. As these services are stateless, load-balancing is simple. The Storage Management Service can also run on several nodes, which then have to be configured to deliver the same content from the base layer, e.g., point to the same database server (or database cluster) on the network and distributed (Grid) file storage. The next step of preparing for even higher loads can be achieved by creating replicas of the content. For this, the Replication Service is placed in the Base Layer, to allow transparent (re-)configuration.[22]

9.2 Micro-Scale Content Management using Filesystem and Lucene

Not all collections require Grid-installation to be handled. Using the Generic Storage Model proposed in Chapter 4.1.1, we can analyze the structure of the content to better understand the needs for managing the collections.

Table 9.1 shows information about two particular collections that will be used for experiments in Chapter 10. As these are collections used for image retrieval benchmarks, they have to be considered unmodifiable – so no changes will occur on the data during their use, in particular no new content will get added or existing content deleted. This means that maintaining consistency as described in Chapter 4.3 is not an issue at all and it makes it also easy to give precise numbers.

The IRMA dataset contains medical images collected in clinical practice and is described in depth in Chapter 10.2.1. It was used for the automatic medical image annotation task at ImageCLEF 2007 [Deselaers et al., 2008a] and contains 10'000 training images in one directory. Each image belongs to exactly one class (class label) and this

[21] DILIGENT was not only used for earth observation (the so-called *ImpECt* scenario), but also by a community dealing with historical art drawings and paintings (the so-called *ARTE* scenario). The collections they dealt with were very focussed and ranged in sizes between 100 to 5000 images. Clearly, for these collection sizes, Grid-storage was not really needed and local file storage was sufficient. For this user community, the main benefit of the usage of DILIGENT was the ability to create virtual organizations to exploit security features of the Grid installation and be able to annotate images in the collections.

[22] There are approaches for handling replication in such settings. One of them is based on a freshness-aware database replication protocol [Akal et al., 2005], which has been adapted to the use in data grids and cloud storage and have been published for instance in [Voicu et al., 2010, Voicu and Schuldt, 2009a, Voicu and Schuldt, 2009b, Voicu et al., 2009a, Voicu et al., 2009b, Voicu and Schuldt, 2008]. They have been developed in the group of the author of this thesis, but are outside the scope of this thesis, in particular since the author of this thesis was neither involved in the details of their design nor their implementation.

Table 9.1: Analysis of two benchmark collections

Generic Storage Concept	IRMA dataset	MIRFLICKR-25000
Raw Content	10'000 training image files	25'000 image files
Relationships	Image - Collection (1) Image - Class (1:N)	Image - Collection (1) Image - Tag (N:M)
Properties	filename	Name Description Exif info (key/value pairs) Copyright owner License

information is available for all training images. The aim of the task is to classify new images automatically based on the knowledge captured in the training images.

The MIRFLICKR-25000 [Huiskes and Lew, 2008] consists of 25'000 images that have been collected from the online image sharing platform *flickr*[23]. One particularity of this collection is, that it contains only images that have been licensed under the family of *Creative Commons licenses*[24] (frequently abbreviated as ©), which allow the usage of the images without the need to contact the legal copyright holder. Most require to name the copyright owner of the image (*Attribution* or ⓘ). Some of them additionally disallow commercial use (*Noncommercial* or $), disallow derived works (*No Derivative Works* or =), or require to publish derived works under the same license (*Share Alike* or ↻). In order to comply with the license, it is therefore important to keep track of the copyright owner and which license he or she assigned to the image. All the images of the benchmark collection are provided in one directory with some available metadata, in particular the Exif information as it was uploaded with the image to *flickr*, the name and description that the owner specified, and the tags that either the owner or any other user of the platform assigned to the image.

In both use cases, there is only one collection with no sub-collections and no hierarchical structure of the content. Moreover, for experiments with the benchmark collections, there is no need to combine such a collection with any other collection; this would even cause a breach of the benchmarking protocol. This leads to a situation, where handling of relationships can basically ignore the membership to collections as long as there is a separation of different datasets (e.g., by configuring the system differently to load one or the other dataset).

For the image tags in the MIRFLICKR-25000 dataset, each image can have multiple tags and each tag can be associated with several images, thus forming an N:M relationship [Chen, 1976, pp. 19f] in terms of the *Entity-Relationship-Model*. In contrast, in the IRMA dataset, each image can and has to be member of exactly one class and each class can have many members, thus forming a 1:N relationship. Another relationship that

[23] `http://www.flickr.com/`, information on this service was already given in Chapter 1.1 on page 5; a screenshot of a shared image and information usually found on the website was presented in Figure 1.3 on page 12

[24] `http://creativecommons.org/`

is not present in the dataset itself, but will be required, is the relationship between the image and the features extracted from it (N:1).

For the purpose of minimal complexity, we developed small Content Management implementations for such collections. In both cases, the images themselves –therefore the raw content– have been stored in a single directory in the local file system. Whenever access to the original content (images and associated metadata) is needed, it can be achieved by directly accessing the files from disk. As the collections are considered unmodifiable, no support has been added to maintain consistency between the content and information used in retrieval (cf. Chapter 4.3).

9.2.1 IRMA dataset maintained using just the file system

For each different kind of feature, the extracted features are stored one after the other in a simple binary representation. In our case, this representation was the Java Object Serialization [Sun Microsystems, Inc., 2003]. As the class membership is a 1:N relationship between image and class, the information can be stored as a property of the image. Furthermore, as the only other property is the filename and both are fairly simple and consume few bytes, these two properties can be stored together with the feature. Using the Object Serialization of Java, a simple wrapper class holds the filename, class label, and the wrapped feature as one object that gets written as a stream to a file. When a search is executed as in Chapter 10.2, the stream of objects is read again, thus recovering the filename and class label without any need for additional lookups or data structures.

9.2.2 MIRFLICKR-25000 dataset maintained using Lucene

Features extracted from the images are stored again in files, with the filename added to identify to which document they belong. The key consideration for using a different approach for the remaining information in the MIRFLICKR-25000 dataset is, that tags and description are text fields for which retrieval should be possible and efficiently supported. Therefore the starting point was to use a popular open-source full text search framework, that was flexible enough to store also all other properties and the relationship with tags. Apache Lucene Core[25] provides such a framework and the Java implementation is lightweight and well-documented and has a rich set of built-in and external language tools, e.g., for stemming and spell checking.

Lucene can store several *fields* for each document. Each field is named and can hold several values. Named fields allow to store properties; the possibility to store several values in a single named field allows to store all tags assigned to an image in a single field. Lucene constructs an inverted index [Büttcher et al., 2010, pp. 33ff] from all content of the fields. Moreover, it also allows to return all values of a named field for a particular document. This functionality is sufficient to manage the relationships between images and tags.[26]

[25] `http://lucene.apache.org`

[26] LIRe (Lucene Image Retrieval [Lux and Chatzichristofis, 2008]), available for download at `http://www.semanticmetadata.net/lire/` provides an GPL-licensed open-source library that uses Lucene also to store extracted features and perform retrieval. As this focuses only on the retrieval part, the

9.3 Discussion

The previous sections followed two very different approaches in implementing Content Management building blocks:

- Section 9.1 focussed on the implementation of a very generic storage model in the storage layer, which also abstracts from backend implementation inside the base layer. On top of this, the content layer adds semantics to particular properties and relationships and dedicated services to perform certain operations on document, collections, and external archives.
- Section 9.2 materializes the content representation directly in a particular technology – either files that store the list of extracted features and/or a text-retrieval engine.

The latter approach is significantly less complex and even provides some support to enhance speed of query execution (as will be discussed further in Chapter 10.5) as, for instance, the full-text engine provides (compact) inverted indexes for fast term-based retrieval. Even considering scalability, the used software Lucene is not inappropriate: It is used in very demanding setting and can (in particular when combined with Apache Solr[27]) also be distributed on multiple nodes; the same is true when the filesystem is backed by a distributed filesystem, e.g., even a Grid-filesystem or using a data cloud.

The main benefits of the approach presented in Section 9.1 are the following:

1. Both approaches in Section 9.2 were only possible because the managed documents were fairly simple: There was no document which consists of several parts. Furthermore, both datasets consisted each of a single collection and this collection did not have any structure into sub-collections. Although it is certainly possible to encode (sub-)collections with additional code, it adds complexity to what-used-to-be a simple implementation.[28]

2. The situation gets even more complex when we consider also modifications of the managed content, e.g., updating the content of a document and re-extracting features, deleting a document, adding a new document to a collection, adding new information as a property to a document for an application that uses content management. In particular the information stored in files is not that easy to alter: Any change that would add or remove some bytes to any document except the last in the file would require adaptation for any subsequent document.[29] The collections

raw content of documents still has to be stored and managed outside the Lucene backend, hence cannot satisfy by itself all functionality to implement the entire content management building block but provides already much of the functionality when content is stored outside the digital library (cf. Chapter 4.2).

[27] `http://lucene.apache.org/solr/`

[28] We added some limited support to our simple implementations in order to cope with the bigger MIRFLICKR dataset consisting of one million images (MIRFLICKR-1M [Huiskes et al., 2010]), separated into 100 folders containing 10'000. This splitting of the collection is commonly beneficial as the performance of many file systems degrades significantly when too many files are stored in a single directory. This was still a fairly simple setting – yet it required to modify the implementation as it was no longer possible to use only the filename alone as an identifier, but had to include the path to the file.

[29] Lucene in such a situation would write to different segments (like databases would use pages) and when index optimization is triggered, the index gets compacted again.

in Section 9.2 were unmodifiable benchmark collections, therefore was no need to maintain consistency between the information used in search and the information that the user gets presented when opening a document from the result list. Commonly, being able to modify a collection is part of the core functionality of content management and consistency has to be achieved, either

- by shielding the content from direct access to the underlying backend and allowing the access only via dedicated operations of content management (this means the raw content all metadata is stored and managed *inside* the the digital library as described in Chapter 4.2)
- by monitoring changes to the content through periodic checks or hooks and update any derived information (this is necessary for raw content and metadata that is stored and managed *outside* the the digital library and was detailed in Chapter 4.3).

3. When we consider Chapter 4.7, the situation is reversed: Here the simple implementation can already assist query formulation and execution by providing efficient access to all features in sequential order when all features of the same type are stored in a single file and Lucene even provides a query language to formulate search predicates with boolean expressions. Such functionality can be added to the layered implementation and may even reuse functionality that the backend provides, in particular when a relational database manages the relationships and properties: Databases provide with SQL a very powerful query language and can index (one/low-dimensional) data efficiently, but it needs to expose this functionality to upper layers and wrap it.

Table 9.2 summarizes very roughly the comparison of the different implementations. The "*or*" in some table entries indicates that the base functionality cannot fully provide the aspect and additional code needs to be written in order to improve the situation within the bounds of what is possible.

This comparison is far from being an exhaustive evaluation of the implementation possibilities mentioned in Chapter 4.4; in particular since also hybrid solutions are possible, e.g., using Lucene as the backend for a layered implementation to manage properties and replicate all features of same type stored inside the content management backend to a single file to assist sequential scans. We did not include any quantitative analysis in the comparison like lines of code needed for the implementation or runtime behavior as this depends very much on the particular language of implementation, personal coding skills and time invested in optimizing code, as well as the used technology and environment for performing the runtime tests.

Table 9.2: Comparison of Content Management Implementations

	Layered Implementation (Chapter 9.1)	Simple Files (Chapter 9.2.1)	Files + Lucene (Chapter 9.2.2)
Complexity of Implementation	simple model, but additional layers add code	little code for base functionality	little code for base functionality
Support for Relationships	fully supported as part of storage model	none *or* limited to hierarchical relationships	none *or* limited to hierarchical relationships
Support for Properties	fully supported as part of storage model	limited as part of feature *or* filename convention	fully supported as named fields in Lucene
Maintaining Consistency	inherent when content stored inside DL	none *or* needs monitoring of file system	none *or* needs monitoring of file system
Scalability	very good with appropriate backends	strong limits by used file system	good with Lucene extensions
Assistance for Search	little built in, may need auxiliary data structures	built-in support for sequential scans	built-in support for scans and index lookups

10

Query Execution

In Chapter 2.2 we showed the very different properties and requirements that the *Task Input and Aim* imposes on any image-related search task. As we are interested in defining and implementing reusable building blocks, there will be some common functionality that can be used in any of the particular tasks. The evaluation of how well it serves the task, however, will always depend on the particular task that one tries to solve.

This chapter will use the following structure:

- Section 10.1 will define a common starting point that will be used in any of the tasks that we will address in the sections that follow afterwards. This includes the implementations for the search primitives in similarity search: range search and k-th nearest neighbor search.
- Section 10.2 – Section 10.4 will focus on particular tasks to which a dedicated approach will be described and evaluated:
  - Classification of medical images in Section 10.2,
  - Searching known images with user-drawn sketches in Section 10.2, and
  - Determining geolocations of images in Section 10.4.

  By focussing on particular tasks, we can define quality criteria to measures how well our approach handles the needs of the user.
- Section 10.5 will then take another look on the shared functionality and will present techniques to reduce the time required to execute queries. As the functionality is shared among all tasks, it can improve the user experience in any of them by delivering the search results faster and through this let the user finish the task in less time.

This structure separates the aspects of retrieval performance: Quality of the results and the time it takes to deliver them. By first introducing the general implementation and tailored approaches that solve the user's tasks and then adding the optimization later we follow the rules on optimization described by [Jackson, 1975][1]:

[1] Quoted after [Bloch, 2003, p. 162].

Rule 1. Don't do it.

Rule 2 (for experts only). Don't do it yet – that is, not until you have a perfectly clear and unoptimized solution.

So let's start with the clear and unoptimized solution...

10.1 Starting Point: Basic Implementation of Search Primitives

For performing any similarity-based search, the (dis-)similarity between the query and the content in the collections have to be computed to determine the results. The straight forward implementation would simply compute the distance to every image in the collection as shown in Pseudo Code Listing 10.1.[2]

```
SEARCH(𝔔, Docs, Φ, Δ)
1  Res ← ∅
2  for each 𝔯 ∈ Docs do
3          dist ← Δ(φ(𝔔), Φ({𝔯}))
4          Res ← Res ∪ {(dist, 𝔯)}
5  SORT(Res)        ▷ Sort into ascending order based on distance
6  return Res
```

Pseudo Code Listing 10.1: Simple query execution

Usually, the user is not interested in all the distances between the query and the documents in the collection, but only in those with low distances. In Chapter 5.4 range search and nearest neighbor search have been introduced, which use either an absolute criterion (the range on the distance) or a relative criterion (only k nearest neighbors) for restricting the list of search results. Equation (5.25) and Equation (5.26) introduced in Chapter 5.4 can be used to reduce the results determined through Pseudo Code Listing 10.1. However, the semantics of the equations can also be incorporated already while performing the search.

Pseudo Code Listing 10.2 implements a range search, with the value of *range* defines the upper bound on the distance on Line 5. This is a straight forward implementation of Equation (5.25) which also uses just an upper bound to define a range. In a strict sense

[2] Pseudo code follows the format of [Cormen et al., 2001] to present code in a commonly known, compact form. Source code closer to real implementations tends to add verbosity by the overhead to name the used libraries. A number of algorithm could be expressed also in a declarative formula. By using pseudo code even for simple algorithms, the similar structure between several algorithms becomes more visible as only individual lines change. Notation assumes that simple tuples of image *img* and distance *dist* can be expressed as $(dist, img)$ and $(a, b) \leftarrow (c, d)$ a shorthand for $a \leftarrow c, b \leftarrow d$ – thus avoiding the need for dedicated data structures or multiple arrays to keep both values together.

```
RANGESEARCH(𝔔, Docs, Φ, Δ, range)
1  Res ← ∅
2  For each ℜ ∈ Docs
3       do
4          dist ← Δ(Φ(𝔔), Φ({ℜ}))
5          if dist ≤ range
6            then Res ← Res ∪ {(dist, ℜ)}
7  ▷ Optional: SORT(Res) to return sorted order
8  return Res
```

Pseudo Code Listing 10.2: Simple range search execution

of the definition of a range search, the final results do not have to be ordered. For real world applications, when results are presented to an end user, they usually should be ordered by distance; only when the range search delivers just an intermediate result, the ordering can commonly be ignored.

```
NEARESTNEIGHBORSEARCH(𝔔, Docs, Φ, Δ, k)
 1  Res ← LIST of size k
 2  For each ℜ ∈ Docs
 3       do
 4          dist ← Δ(Φ(𝔔), Φ({ℜ}))
 5          if length(Res) < k
 6            then
 7                 INSERT-SORTED(Res, length(Res) + 1, dist, ℜ)
 8            else
 9                 (dist_k, img_k) ← Res[k]
10                 if dist < dist_k
11                   then
12                        INSERT-SORTED(Res, K, DIST, ℜ)
13  return Res
```

Pseudo Code Listing 10.3: Simple nearest neighbor search execution

The k-th nearest neighbor search (kNN) is described in Pseudo Code Listing 10.3. It uses INSERT-SORTED from Pseudo Code Listing 10.4 for iteratively adding each new value to list, maintaining the order of the list. This is based the inner loop of INSERTION-SORT from [Cormen et al., 2001, p. 17]. With the parameter *pos* it can either add a new

INSERT-SORTED(A, pos, key, img)

```
1  i ← pos − 1
2  while i > 0 and dist_i > key with (dist_i, img_i) ← Res[i]
3        do A[i + 1] ← A[i]
4           i ← i − 1
5  A[i + 1] ← (key, img)
```

Pseudo Code Listing 10.4: Sorting of a single entry based on inner loop of INSERTION-SORT from [Cormen et al., 2001, p. 17]: Every entry at a position less than *pos* is assumed to be sorted already.

value to the existing list if $pos = length(Res) + 1$ or replace an existing value if $pos \leq length(Res)$.[3]

With this elementary implementation of search primitives, we can now adapt the approach to particular problem domains.

10.2 *k*NN-based Hierarchical Medical Image Classification

The generated class labels as the results of *Image Classification* as described in Chapter 2.2.2 on page 32 cannot only be used for *Retrieval by Class*, but also used like any other metadata for faceted searches in the context of *Known Image Searches* as well as *Themed Searches*.

As already mentioned in Chapter 2.4 on page 64 and pointed out in [Enser, 2008], medical image retrieval is one of the example domains where the image is hardly ever considered in isolation, but tightly linked to the patient, health record, case, and/or

[3]This algorithm can certainly get implemented more nicely by using a data structure which always maintains the order based on the distance – for instance based on HEAPSELECT as in [Press et al., 2007]. When using some other dynamic data structure, it becomes important that this data structure cannot just add items efficiently, but also delete or better replace previous entries to reduce unnecessary memory overhead. This is something that you get for free when using a sorted list of k entries. Furthermore, as we will see in Section 10.5.2, this complete ordering can be used to derive the threshold for the Early Termination strategy for which –in the case of a nearest neighbor search– the distance of the k-th best entry has to be used for which it is necessary to know the worst value within the k best values seen so far; just keeping the single best value on top of the data structure is not sufficient, hence half-sorted trees (a.k.a. Heap) have commonly worse runtime performance than a simple, fully-sorted list.
INSERTION-SORT by itself is not known as an efficient sorting algorithm as it has a worst-case complexity of $O(n^2)$. In the case of kNN search $n = k$ for the sorting part, but it gets called from within NEARESTNEIGHBORSEARCH for every image in *Docs* that has a distance smaller than the k previously seen values. The combination of NEARESTNEIGHBORSEARCH and INSERT-SORTED has therefore a worst-case complexity of $O(k \times |Docs|)$. The implementation can act as a basis for future optimization as k is usually much smaller than $|Docs|$ and the condition in line 10 usually prevents many of the invocations of INSERT-SORTED in realistic search settings.

other domain-specific metadata. Therefore the availability of such metadata and its accuracy can become crucial for many search tasks.

In clinical practice, medical images in digital form are usually stored in a picture archiving and communication system (PACS), for which commonly the Digital Imaging and Communications in Medicine (DICOM) format is used as a container and communication protocol (cf. [Bidgood et al., 1997, Mildenberger et al., 2002]). The DICOM header can contain metadata that can be relevant for retrieval, e.g., the tag *Body Part Examined*. However, as a study has revealed in [Güld et al., 2002], the quality of DICOM header information is far from ideal with an error rate above 15% for the tag *Body Part Examined*. According to this study, the reason why so much of the information is wrong –and therefore would be misleading when relied on in searches– is mainly driven by two aspects:

1. Flaws in the way the DICOM standard requires information to be provided: For the example of the tag *Body Part Examined*, the standard allows only to set one value. The categories defined by the DICOM standard *CHEST*, *ABDOMEN*, *NECK*, and *HEAD* as well as *LEG*, *ANKLE*, and *FOOT* are connected when observing the human body; certain imaging modalities will frequently cover more than a single body part, thus making it impossible to label the images optimally within the standard (cf. [Güld et al., 2002]). Using a better suited classification scene can resolve or at least reduce this problem, e.g., using the hierarchical IRMA code [Lehmann et al., 2002, Lehmann et al., 2003a, Lehmann et al., 2003b].

2. Automatic tagging through modality in presence of conflicts in the overall aim: The DICOM header is usually written by the modality based on the parameters that have been set when taking the image[4]. The main aim of the clinical personnel is to achieve best image quality with little manual adjustments – therefore often applying an examination protocol outside its normal context (cf. [Güld et al., 2002]), which may result in metadata being stored that is not accurate and therefore not ideal for retrieval. This problem can be detected if sufficiently good automatic image classification algorithms do exist that can fix or at least warn about combinations of image content and metadata that appears implausible. Such a automatic classification of medical images has been described in [Lehmann et al., 2005].

In the following, we will describe a solution built essentially from components as mentioned in Chapter 5 to provide automatic classification of medical images with the hierarchical IRMA code with some modifications to improve classification accuracy. This follows the setup of the ImageCLEF automatic medical image annotation task [Deselaers et al., 2008a] that was part of the ImageCLEF benchmark challenge from 2005 to 2009 and uses the dataset of 2007.[5] The following paragraphs will describe in

[4]Similar as Exif information [JETIA, 2010] is recorded by common digital cameras; also here errors can get introduced, e.g., the wrong date and time set for the camera or for cameras with interchangeable lenses, sometimes not all lenses provide full / correct information, in particular about the aperture and focal length. However, a significant difference is, that based on the parameters set by the clinical personnel, the medical imaging modality choses presets which correspond to certain tags.

[5]Much progress has been made in the recent years and certainly dedicated classification approaches can achieve even better accuracy. In the line of the idea of building blocks, this nevertheless shows that

more detail how the hierarchical classification in this particular task is performed, which will become important for the evaluation in which an error scores measures how well an automatically assigned label fits to the correct hierarchical code of an image.

10.2.1 Hierarchical Classification: The IRMA Code

To ease retrieval of medical images, not only accurate labels are needed, but they must also cover the aspects relevant for retrieval to the needed degree of detail. Exploiting hierarchical codes allows to select the appropriate level of detail at the time of querying rather than at the time the image is indexed. It also allows that a system may express in an automatically assigned code how uncertainty remains: All parts of the hierarchy for which the determined labels are certain can be named while the uncertain parts can be expressed through wildcard characters.

Medical images even of the same patient and the same body region can differ a lot depending on the modality used for acquisition of the image. Figure 10.1 shows a small sample of the variety that is common in clinical practice. The IRMA code therefore tries to isolate aspects into four different axes, each of which is encoded by a sequence of "digits" / alphanumerical characters which are described in [Lehmann et al., 2002] as:

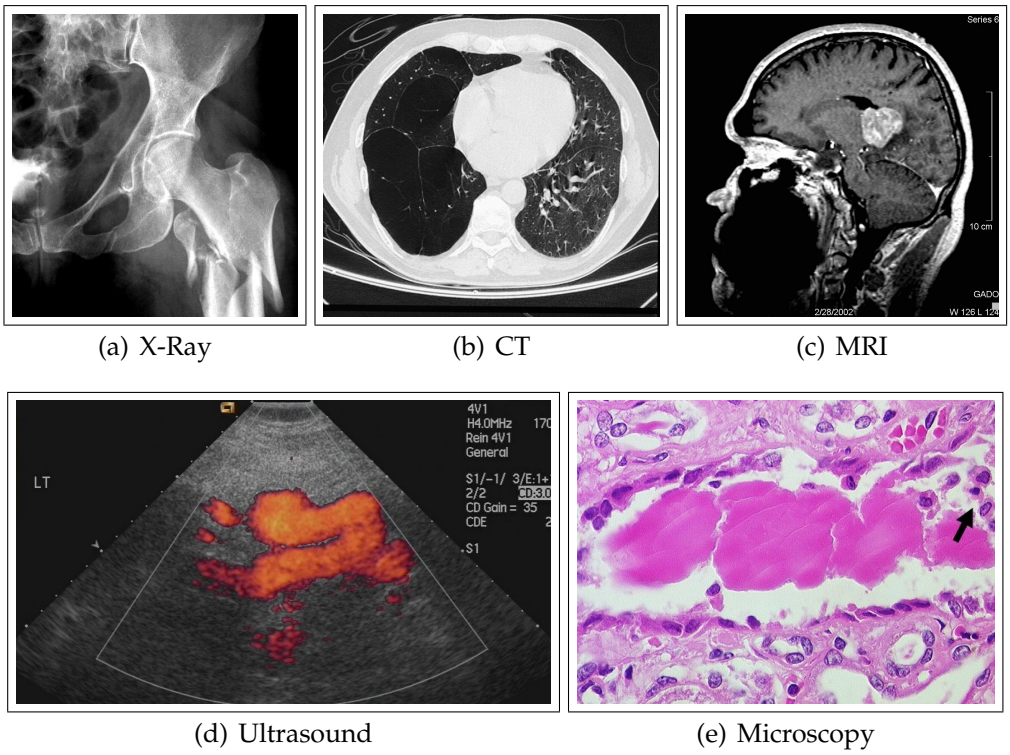
- *Technical Code (T-code, 4 digits)*: The technical axis is used to identify the modality, starting with imaging physical techniques (e.g.: 1 = x-ray, 2 = ultrasound, 3 = magnetic resonance, 4 = optical, ...). The code in total uses 4 digits, of which each following provides more details. The second digit adds the modality position (e.g.: 11 = plain film projection radiography, 12 = fluoroscopy, 13 = angiography, 14 = computed tomography, ...), the third digit specifies the technique (e.g.: 111 = digital, 112 = analog, 113 = stereometry, 114 = stereography, ...) and the fourth position of T-code assesses subtechniques (e.g.: 1111 = tomography, 1112 = high energy, 1113 = low energy, 1114 = parallel beam, ...).

- *Directional Code (D-code, 3 digits)*: The directional axis describes the orientation starting with the common orientation (e.g.: 1 = coronal, 2 = sagital, 3 = transversal, 4 other) and giving more a detailed specification in the second position (e.g.: 11 = posterior-anterior (pa), 12 = anterior-posterior (ap)). With the third digit, functional orientation tasks of the examination can be described (e.g.: 111 = standing, 112 = lying, 113 = inclination, 114 = reclination, ...).

- *Anatomical Code (A-code, 3 digits)*: The anatomical axis allows coding of the anatomical region. In total, nine major regions are defined (e.g.: 1 = total body, 2 = head-/scull, 3 = spine, 4 = upper extremity, ...) with up to three positions within the code (e.g.: 3 = spine, 31 = cervical spine, 311 = dens).

- *Biological Code (B-code, 3 digits)*: The biological axis is used to determine the organ that is imaged. For example, fluoroscopy of the abdominal region may access the cardiovascular or the gastrointestinal system. On the top-level of the functional

a) a baseline implementation is possible reusing existing components and b) better components can and should be used to replace existing ones when they get available.



(a) X-Ray (b) CT (c) MRI

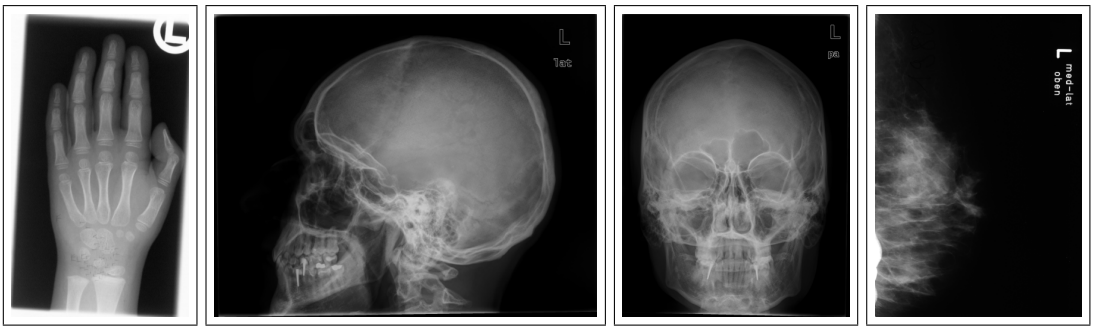
(d) Ultrasound (e) Microscopy

Figure 10.1: Medical images from different modalities: (a) shows a fracture of the femur bone using X-ray and therefore a projection of the 3D object into 2D. (b) shows the computerized tomography (CT) of the lung, (c) the magnetic resonance image (MRI) of the head – both always show only cross-sections (slices) of 3D objects. (d) shows a Doppler ultrasound image, which contains colors to indicate measurements instead of the visual appearance of the object itself. Such coloring is also common in functional MRI (fMRI). (e) shows the microscopic image of kidney tissue. (All images are taken from the query subjects used in the *ImageCLEF 2005 Medical Image Retrieval* benchmark task [Clough et al., 2005, Hersh et al., 2006].)

> IRMA-code, ten organ systems are specified (e.g.: 1 = cerebrospinal system, 2 = cardiovascular system, 3 = respiratory system, 4 = gastrointestinal system, ...) each of which having up to three digits to exactly identify the organ in question (e.g.: 1 = cerebrospinal system, 11 = central nervous system, 111 = metencephalon).

Any 0 indicates "unspecific"; due to the hierarchical coding any digit following a 0 in any axis must be 0 - until all digits of the axis are filled. Figure 10.2 shows some examples from the dataset used for the ImageCLEF medical annotation benchmark task with the IRMA code and the full textual explanation. The images in the dataset are based on a randomly-selected set of images from clinical practice that have been labeled by domain experts and anonymized. Due to the great importance that X-ray (still) has, the dataset contains *only* X-ray images; so only a small subset of the potential codes of the technical axis are found in the dataset.

For retrieval as well as evaluating the hierarchical classification, the wildcard character "*" is introduced (cf. [Deselaers et al., 2008a]): Similar to the value for unspecific



(a) (b) (c) (d)

Figure 10.2: Example X-ray images with IRMA code:
(a): *1121-110-414-700* - technique: x-ray, plain radiography, analog, overview image; direction: coronal, posteroanterior (PA), unspecified; anatomy: upper extremity (arm), hand, left hand; biosystem: musculosceletal system, unspecified
(b): *1121-220-230-700* - technique: x-ray, plain radiography, analog, overview image; direction: sagittal, lateral, left-right, unspecified; anatomy: cranium, neuro cranium, unspecified; biosystem: musculosceletal system, unspecified
(c): *1121-120-200-700* - technique: x-ray, plain radiography, analog, overview image; direction: coronal, anteroposterior (AP, coronal), unspecified; anatomy: cranium, unspecified; biosystem: musculosceletal system, unspecified
(d): *1124-410-620-625* - technique: x-ray, plain radiography, analog, low beam energy; direction: other orientation, oblique, unspecified; anatomy: breast (mamma), left breast, unspecified; biosystem: reproductive system, female system, breast
(All images and codes taken from the IRMA dataset used for ImageCLEF 2007 Medical Annotation Task [Deselaers et al., 2008a].)

classes indicated by "0", this allows anywhere in the hierarchy to state that the classification along this axis will not get further detailed. The semantics, however, is slightly different as "0" means that it does not make sense / it is not possible to further elaborate the classification, whereas in retrieval "*" means that there is no need to separate – all subclasses are wanted. In automatic classification, "*" can be used to express uncertainty, that is, the classification along the axis until the "*" is certain / with high probability, but further distinction into subclasses is too uncertain.

10.2.2 Distance Measure for Medical Image Classification

The Image Distortion Model (IDM) as described in Chapter 5.2.3 on page 122 is a deformable model and has originally been proposed in [Keysers et al., 2004] for optical character recognition and successfully applied to medical image retrieval [Güld et al., 2005, Keysers et al., 2007]. For this, the images are reduced to a smaller, common size; e.g., 32×32 pixels. We will denote this downscaling as ϕ_{ds}, for which the feature space $\mathcal{F}$ is a subset of the set of all images $\mathcal{I}$. As a shorthand, we denote $\phi_{ds}(\mathfrak{Q}) = Q$ and $\phi_{ds}(\mathfrak{R}) = R$ for the two images $\mathfrak{Q}, \mathfrak{R} \in \mathcal{I}$. Then, pixels in in the reduced versions can get compared with each other. As a simple base-

line, also other distance measures can be used - for instance the Euclidean distance (cf. [Güld and Deserno, 2007]). In the following, we will present a modification of the Image Distortion Model compared to [Keysers et al., 2007] that allows for some more flexible parameterization.

For clarity and easier discussion, we will use the following generic formula that based on a pixel distance computation function p computes the overall distance:

$$\delta(Q,R,p) = \sqrt{\sum_{x=0}^{height} \sum_{y=0}^{width} p(Q,R,x,y)} \tag{10.1}$$

The Euclidean distance results then in using p_{euclid} in Equation (10.1) to compute the distance of individual pixels:

$$p_{euclid}(Q,R,x,y) = (Q(x,y) - R(x,y))^2 \tag{10.2}$$

Warp Range for Displacements

For computing the distance with some image distortion, we can use the following refinement, in which $C(x-x', y-y')$ adds some penalty for displacements:

$$p_{IDM}(Q,R,x,y) = \min_{\substack{x' \in [x-w,x+w], \\ y' \in [y-w,y+w]}} (Q(x,y) - R(x',y'))^2 + C(x-x', y-y')) \tag{10.3}$$

Each displacement may get penalized with some costs that are associated with this displacement. These costs are computed using the cost function C, which in contrast to the original proposal of IDM also includes the possibility to multiply the differences by a factor:

$$C(Q,x,y,R,x',y') \quad = \quad \left(Q(x,y) \; - \; R(x',y')\right)^2 \; \times \; m_{|x-x'|,|y-y'|} \; + \; a_{|x-x'|,|y-y'|} \tag{10.4}$$

M is a matrix containing constant factors $m_{i,j}$ for penalizing the displacement, A another matrix with constant addend $a_{i,j}$. As in many programming languages, we start counting the row and column indices of the matrices at 0, thus indicating no displacement. For this reason, A is set to zero at row 0, column 0, such that no costs are added when pixels are not displaced ($a_{0,0} = 0$) and also M is set to a factor of 1.0 at row 0, column 0 ($m_{0,0} = 1.0$). Table 10.1 shows two pairs of instances of such matrices used to penalize displacement. The combination A_{369} and M_{369} does not use factors on the distance, therefore all entries of M_{369} are set to one.

Threshold for Individual Pixels

The maximum distance that a single pixel can contribute equals the complete difference between a black pixel with gray value of 0 and a plain white of 255. Such differences only occur in extreme situations since usually there are hardly neither plain white nor black areas. However, for instance in case of padding in slightly rotated images, such

Table 10.1: Example matrices for costs associated with deformation.

$$A_{369} = \begin{pmatrix} 0 & 3 & 6 & 9 \\ 3 & 6 & 9 & 12 \\ 6 & 9 & 12 & 15 \\ 9 & 12 & 15 & 20 \end{pmatrix} \qquad M_{369} = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix}$$

$$A_{4816} = \begin{pmatrix} 0 & 4 & 8 & 16 \\ 4 & 8 & 16 & 32 \\ 8 & 16 & 32 & 48 \\ 16 & 32 & 48 & 64 \end{pmatrix} \qquad M_{4816} = \begin{pmatrix} 1.0 & 1.2 & 1.4 & 1.6 \\ 1.2 & 1.4 & 1.6 & 1.8 \\ 1.4 & 1.6 & 1.8 & 2.0 \\ 1.6 & 1.8 & 2.0 & 2.5 \end{pmatrix}$$

differences in images belonging to the same class in the ImageCLEF2007 data set do exist. Such significant but local differences should therefore not negatively affect the classification if they occur infrequently. In order to limit the maximum contribution of a single pixel, [Keysers et al., 2003] proposes the use of a threshold.

We therefore introduce a threshold t and an intermediate function $p_{p,t}$ that can be applied on any pixel distance computation function:

$$p_{pt}(Q,R,x,y) = \begin{cases} p(Q,R,x,y) & \text{for } p(Q,R,x,y) < t^2 \\ t^2 & \text{for } p(Q,R,x,y) \geq t^2 \end{cases} \tag{10.5}$$

We use t^2 instead of t here in order to be in line with the square root that will be taken in Equation (10.1).

Local Context

The retrieval quality of IDM is significantly increased, if we do not restrict IDM to single pixels, but consider also their local context [Keysers et al., 2004]. The local context is defined by an area of pixels around the central pixel that differ in their column and row value by not more than the local context range lc. IDM with local context computes the average distance between those pixels in the area with the corresponding pixels of the reference image.

$$p_{IDMlc}(Q,R,x,y) = \min_{\substack{x' \in [x-w,x+w], \\ y' \in [y-w,y+w]}} \frac{\sum_{\hat{x}=-lc}^{lc} \sum_{\hat{y}=-lc}^{lc} ((Q(x+\hat{x},y+\hat{y}) - R(x'+\hat{x},y'+\hat{y}))^2 + C(x-x',y-y'))}{(2lc+1)^2} \tag{10.6}$$

[Keysers et al., 2004, Keysers et al., 2007] uses a local context range $lc = 1$, hence using a 3×3 pixel area for the IDM with local context. In our approach, we extended the local context range up to $lc = 3$. Most of the runs that we submitted to ImageCLEF2007, we used $lc = 2$, our best run uses $lc = 3$. The area covered by the original parameters and the increased sizes are illustrated in Figure 10.3.

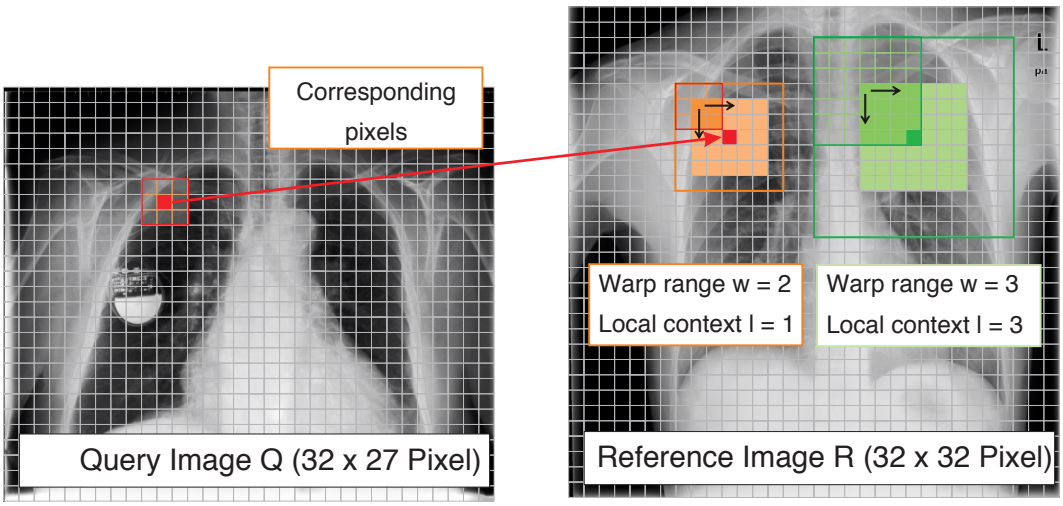


Figure 10.3: Image Distortion Model for two medical images, illustrated for one pixel and its local context in image Q and the corresponding area to which it may be warped in orange color. The small arrows indicate that the local context area in the reference image R will move to minimize the cost of deformation. The green area in image R shows the effect of choosing higher values for the w and lc.

Pixel Intensities and Gradients

The local context is taken from the image directly such that the directionality of edges is preserved in the local context of pixels. Such edges can be detected using any edge detection algorithm, e.g., a Sobel filter. For the computation of IDM, either the gray values of the pixels can be used directly, or their gradients (detected edges), or a combination of both.

Notice that the shape of an edges is to a certain extent affected when the aspect ration of an image is changed, for instance, if all images would get scaled to a 32×32 size independent of the original aspect ratio of images. Since edges and their directionality have shown in experiments to influence the retrieval quality, we preserve the aspect ratio of images during scaling.

[Güld and Deserno, 2007] uses a scaling of $X \times 32$ pixels, thus using a fixed height while preserving the aspect ratio. For images in landscape orientation, this may lead to a width significantly greater than 32 pixels, whereas portrait pictures would have less than 32. In our approach, we reduce the difference in pixels by scaling the image such that the longer side has not more than 32 pixels. Therefore, images in portrait orientation will have the same size as if we used $X \times 32$, but landscape pictures will be restricted to a width of 32 pixels. In order to make images of different width or height comparable, we identify each corresponding pixel by scaling based on there relative position in the image as shown in Fig. 10.3.

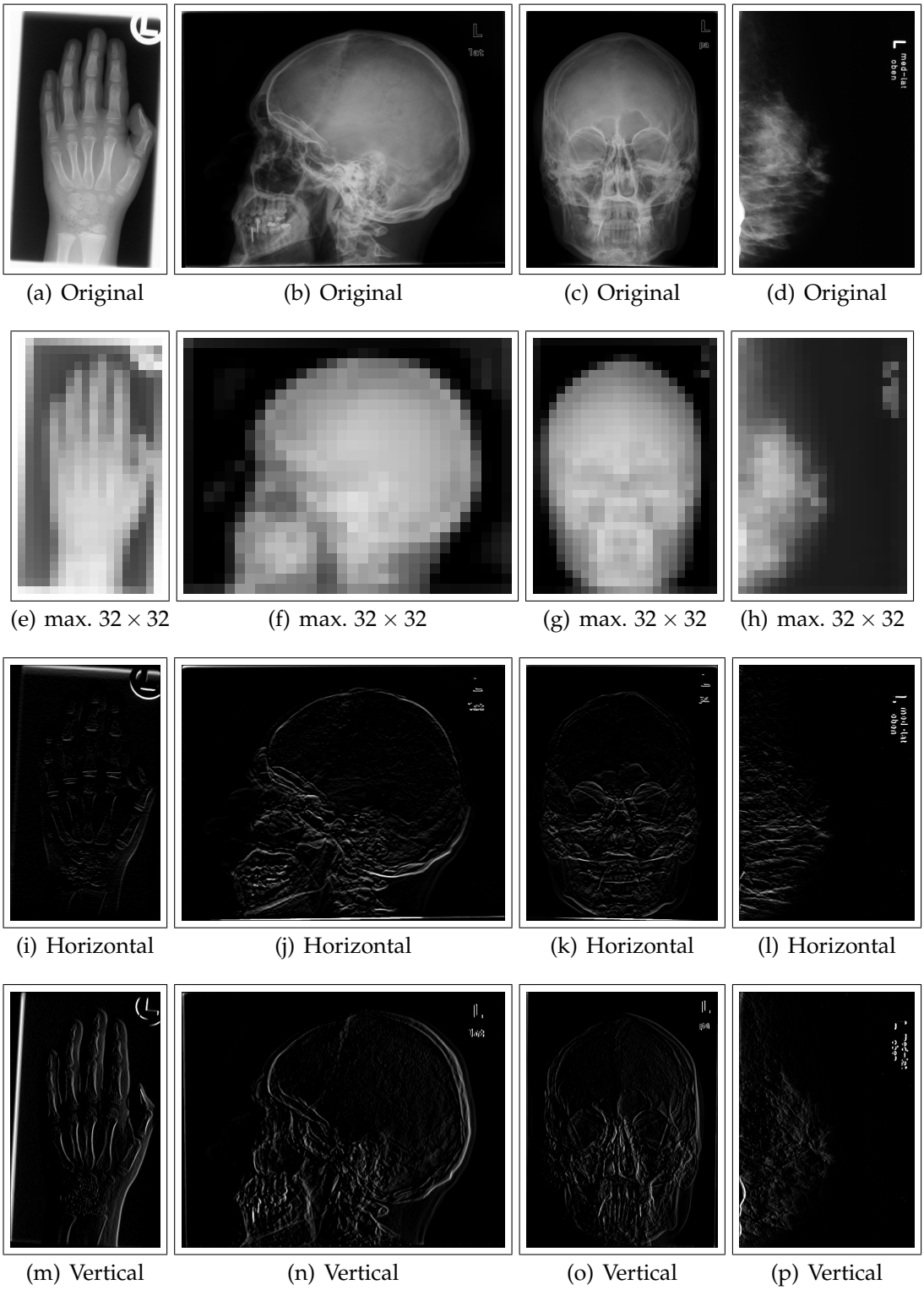
The individual steps and intermediate result images are shown in Figure 10.4 and 10.5, in particular for the operations of detecting edges with the Sobel edge detector and scaling using bicubic interpolation. Notice that are quite a number of algorithms to de-

tect edges and perform scaling – all of them may achieve slightly different results.[6] To achieve precisely the same results, it is necessary to follow exactly the same processing pipeline with the same algorithms and same parameters, applied in the same order; otherwise there will be at least some deviations. Notice also, that just from visually judging the results one may get some helpful insights but not necessarily good estimations on how the actual retrieval performance will behave: For classification tasks, it is important how good a feature separates the instance of one class from instances of other classes. With respect to this criterion, it may very well be the case that overemphasizing parts of an image while ignoring other parts (therefore shifting the *Matching Tolerance*) can achieve better results than equally weighting all available information from the images. These details are included here in order to allow maximal repeatability of the results and discussion.

To improve the speed of retrieval, [Keysers et al., 2004, Thies et al., 2005, Güld et al., 2005] propose to use downscaled versions of the image of either 16×16 or 32×32 pixels using the pixel intensities with the Euclidean distance and apply the more complex IDM only on the 500 nearest neighbors.[7] Therefore an additional feature extraction step for the reference collection as well as every query image needs to be performed. Moreover, the result quality degrades, when the best matches are not among the 500 nearest neighbors with regard to the Euclidean distance. To keep the loss in retrieval quality as low as possible while still achieving fast results, we apply the Euclidean distance using the same scaling as we use for IDM when filtering is performed. Minimal degradation in retrieval quality was achieved when both layers, intensities and gradients, are used and weighted for this modified Euclidean distance in the same way as for IDM, therefore turning theoretically into IDM with parameters $w = 0$, $lc = 0$, $A = (0)$ and $M = (1)$. Although we would be able to use IDM with these parameters, we keep a separate implementation to avoid any potential negative impact on execution times due to overhead such as additional bound checks. During query execution we no longer need to extract two different sets of features of the query images, since both distance functions can use the same set.

[6] See for instance [Fisher et al., 2004] and in particular `http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm`.

[7] Such a hierarchical filtering approach was also used in [Simard, 1993], but mentions also the problem of identifying the minimum number of candidates to keep in early stages.



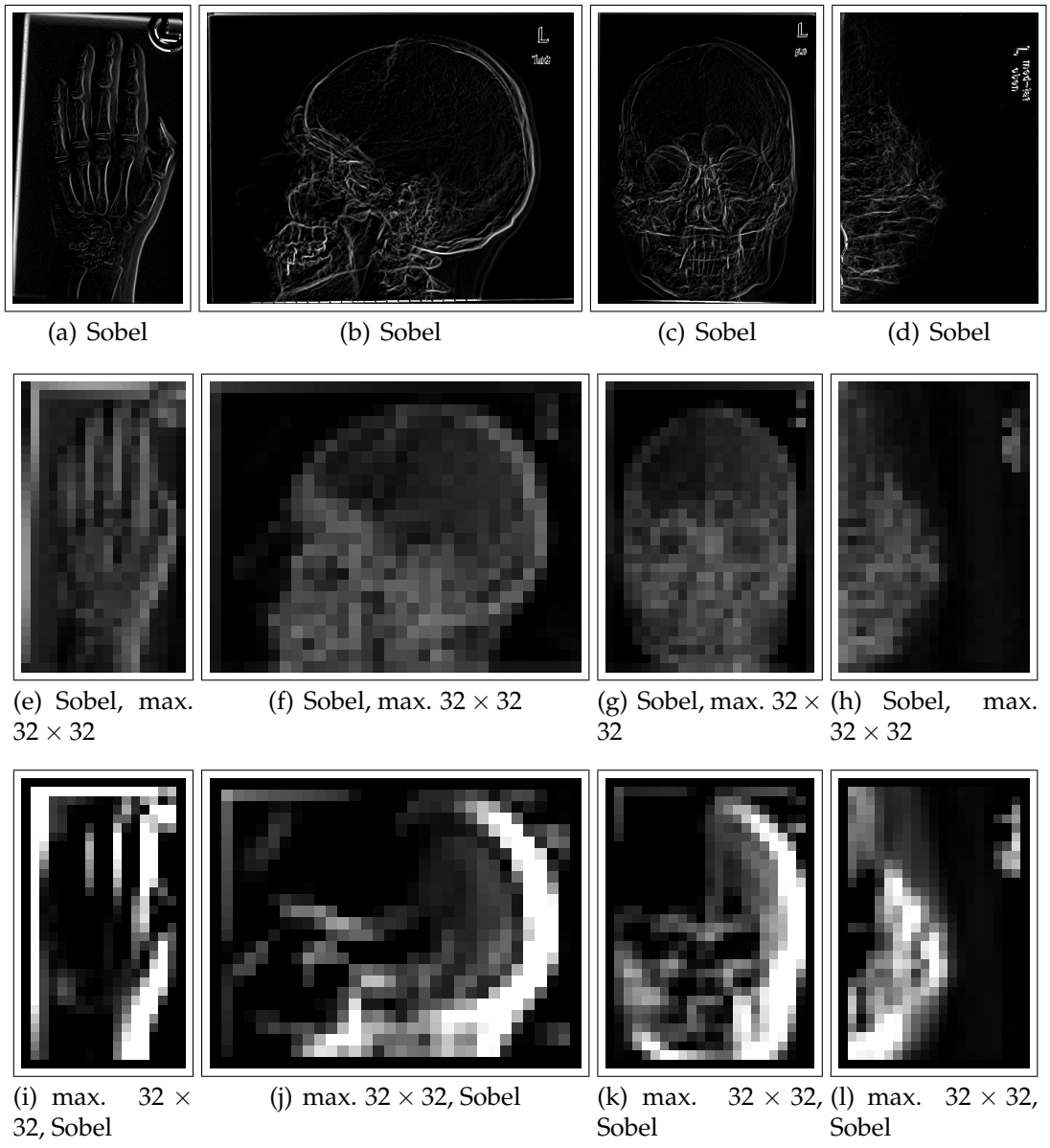
(a) Original (b) Original (c) Original (d) Original

(e) max. 32×32 (f) max. 32×32 (g) max. 32×32 (h) max. 32×32

(i) Horizontal (j) Horizontal (k) Horizontal (l) Horizontal

(m) Vertical (n) Vertical (o) Vertical (p) Vertical

Figure 10.4: Image processing steps before applying the image distortion model: The original images ((a)–(d)) are either scaled directly to the desired size of 32 pixels for the longer edge ((e)–(h)) or edges detected. (i)–(l) shows the result of applying a horizontal Sobel filter, (m)–(p) a vertical Sobel filter.



(a) Sobel (b) Sobel (c) Sobel (d) Sobel

(e) Sobel, max. 32×32 (f) Sobel, max. 32×32 (g) Sobel, max. 32×32 (h) Sobel, max. 32×32

(i) max. 32×32, Sobel (j) max. 32×32, Sobel (k) max. 32×32, Sobel (l) max. 32×32, Sobel

Figure 10.5: Image processing steps before applying the image distortion model (continued): (a)–(d) shows the combined filter for horizontal and vertical edges. (e)–(h) show the result when these filtered images are scaled to 32 pixels for the longer edge with bicubic interpolation. In contrast, (i)–(l) show the result when the order of operations is inverted: First the images are scaled as in Figure 10.4(e)–(h), then the edges detection filter is applied – and results in much stronger edge responses in some places, but also due to the loss of detail available at the edge detection step, some missing edges, e.g., towards the left and right side of the hand in (i), at the forehead of the skull in (j), the top-left part of (g) and for what should be prominent horizontal edges in (l).

10.2.3 Classifier based on k nearest neighbor search

In order to be able to classify images, the *labels* and what characterizes instances of the class with a particular label need to be known. In our case, the space of labels $\mathfrak{L}$ corresponds to all possible IRMA codes described in Section 10.2.1 on page 208. To perform this classification automatically, it can be sufficient to have a training dataset $Train \subset \mathcal{I}$ of which the correct labels are known, e.g., by manually annotating all images in this set by domain experts as ground truth information.

Let Λ be the function that returns for any image $\mathfrak{I} \in Train$ the known label:

$$\Lambda : Train \rightarrow \mathfrak{L} \tag{10.7}$$

As the labels of the training set are known, this can be seen and implemented as a simple lookup operation.

For determining the class of the query image of which the label is not yet known, [Keysers et al., 2007] propose the nearest neighbor rule, that is, to take the class of the reference image with minimal distance (1-NN). In addition, [Keysers et al., 2004] proposes to take into account the class information of the three nearest neighbors, where the information of each is equally weighted. The ImageCLEF runs performed in [Güld et al., 2005] also take the 5 nearest neighbors into account, these resulted in scores worse than taking only the 1-NN.

Let *Res* be the set of k pairs of retrieved images and distances as returned by Pseudo Code Listing 10.3 on the training set given the query image $\mathfrak{Q}$ and the combination of selected features Φ and distance measure Δ:

$$Res \leftarrow \text{NEARESTNEIGHBORSEARCH}(\mathfrak{Q}, Train, \Phi, \Delta, k)$$

Pseudo Code Listing 10.5 provides a simple implementation in which each of the nearest neighbors is voting for its label. In the end, the label will get returned that received most votes – or, if maximum number of votes have been achieved by several labels, the label of the nearest neighbor is returned. The latter can occur quite easily for small values of k, e.g., if $k = 2$ and the two labels are not the same. Pseudo Code Listing 10.5 keeps the labels in the order of the appearance within the k nearest neighbors. By assigning *max* to the lowest index and therefore the nearest neighbor on line 2, this puts the label of the nearest neighbor –and therefore also the class with lowest distance– in a better position: Whenever a situation is encountered with several labels having the same number of votes and the class of the nearest neighbors is among them, it will get selected.[8]

[8]Other alternative solutions could randomly pick any of the candidates with maximum votes or pause as the situation is undecided, potentially picking more neighbors until there is a decision. Note that using data structures such as a hash table or a half-sorted tree (a.ka. Heap) for keeping the list of labels and achieved scores usually results in such a situation in which the selection of a label in case of same scores can become runtime-dependent – and therefore appear in a theoretical analysis similar to picking a value randomly. However, none of these approaches seem to adequately maintain the intention behind a nearest neighbor approach, that is, closer neighbors should be more important. The approach presented in Pseudo Code Listing 10.5 and the alternative that will be presented in Pseudo Code Listing 10.8 therefore seem to be a better options.

```
kNNClassify(Res, k)
 1  labels ← List of size k
 2  scores ← List of size k
 3  numUniqueLabels ← 0
 4
 5  for each (dist, img) ∈ Res
 6       do
 7          l ← Λ(img)
 8          if l ∈ labels
 9            then
10                 i ← Find(labels, l, numUniqueLabels)
11                 scores[i] ← scores[i] + 1
12            else
13                 numUniqueLabels ← numUniqueLabels + 1
14                 labels[numUniqueLabels] ← l
15                 scores[numUniqueLabels] ← 1
16  return labels[MaxScoreIndex(scores, numUniquelabels)]
```

Pseudo Code Listing 10.5: Simple unweighted kNN classification

```
Find(A, key, k)
1  i ← 1
2  while A[i] ≠ key ∧ i ≤ k + 1
3       do i ← i − 1
4  return i
```

Pseudo Code Listing 10.6: Simple sequential search / lookup for the index of a key in a list among the first k entries.

The distance cannot only be used as an indicator in case where the same number of neighbors were found in several classes, but also for weighting the votes differently. This brings one desirable property that a range search would provide to the kNN: In case some of k nearest neighbors are already quite far away, their votes contributes less to the overall solution. Pseudo Code Listing 10.8 shows a variant of kNNClassify that weighs the class information based on the inverse square of the distance [Esters and Sander, 2000]. For this, the highlighted lines 11 and 15 have been changed.

So far, these classifiers did not exploit the hierarchy of the IRMA code. And, in fact, if the nearest neighbor is very close to the image in feature space that should get classified, the results are fine. But when the nearest neighbors are already quite far away, the reliability of the classification is reduced. In such a situation it might be worthwhile

```
MAXSCOREINDEX(scores, numUniqueLabels)
1  i ← 1
2  max ← 1
3  while i < numUniqueLabels
4        do
5           i ← i − 1
6           if scores[i] > scores[max]
7             then
8                  max ← i
9  return max
```

Pseudo Code Listing 10.7: Simple look to determine the index of the entry with highest score.

```
KNNCLASSIFYWEIGHTED(Res, k)
 1  labels ← LIST of size k
 2  scores ← LIST of size k
 3  numUniqueLabels ← 0
 4
 5  for each (dist, img) ∈ Res
 6        do
 7           l ← Λ(img)
 8           if l ∈ labels
 9             then
10                  i ← FIND(LABELS, L, NUMUNIQUELABELS)
11                  scores[i] ← scores[i] + 1/(dist∗dist)
12             else
13                  numUniqueLabels ← numUniqueLabels + 1
14                  labels[numUniqueLabels] ← l
15                  scores[numUniqueLabels] ← 1/(dist∗dist)
16  return labels[MAXSCOREINDEX(scores, numUniquelabels)]
```

Pseudo Code Listing 10.8: Weighted kNN classification

to use the hierarchy and the possibility to express uncertain parts with '$*$' whenever the nearest neighbors have different labels and would therefore disagree in their votes.

Pseudo Code Listing 10.9 generates the such a code with '$*$' from two IRMA codes. The method is called SUPERCODE in analogy with superclasses which can provide the generalization of several classes in biology and object-oriented programming. Notice that it processes the IRMA codes for each axis independently through the outer loop on Line 1, such that when it processes the individual symbols inside the inner loop on

```
SUPERCODE(codeA, codeB)
 1  for i ← 1 to 4
 2       do
 3          axisA ← codeA[i], axisB ← codeB[i]
 4          sameSoFar ← true
 5          for j ← 1 to LENGTH(axisA)
 6               do
 7                  if sameSoFar and axisA[j] = axisB[j]
 8                    then
 9                         newAxis[j] = axisA[j]
10                    else
11                         sameSoFar = false
12                         newAxis[j] = ∗
13          newCode[i] = newAxis
14  return newCode
```

Pseudo Code Listing 10.9: Generation of IRMA Code with wildcards from two IRMA Codes

Line 5, it will set all remaining symbols for this axis to '$*$' as soon as the first disagreement occurs on the axis. This method can be used in cases where the classification with another classifier seems "unreliable", which is the case for kNN classifiers if the nearest neighbor is already far off.[9]

Pseudo Code Listing 10.10 provides an implementation where t_c defines the threshold for using generating the "super code". Since the number of pixels in the query image contributes to the sum computed in Equation 10.1, we normalize the absolute distance by dividing it through the number of pixels in the image before comparing it to the threshold t_c on Line 2. This implementation also allows to set the number of nearest neighbors used in generating the super code (k_c) independently from the number of nearest neighbors to used for the weighted kNN classifier from Pseudo Code Listing 10.8 (k), which can help to set the values such that neither the generated supercode gets too generic nor the weighted k classifier gets too little input for good results.[10]

[9] Of course, not only the nearest neighbor (1-NN) contributes to the quality of the classification, but also the other neighbors in a kNN classifier. Theoretically also thresholds on the other neighbors could be defined to even further control the effect. In practice, however, the weighted kNN classifier handles fairly well in which the distances of the nearest neighbors differ significantly.

[10] A different group at the ImageCLEF 2007 followed a similar approach to the supercode named the "common code" rule [Güld and Deserno, 2007]. They didn't use a hybrid solution, instead submitted runs either using only kNN classifiers or the "common code" generated from the 100 nearest neighbors. The latter did perform worse than there implementation using kNN classifier and our implementation using Pseudo Code Listing 10.10. Considering that the class distribution shows that the training set contains only 10 instances of some classes, 100 nearest neighbors will frequently belong to several different classes – which results in a fairly generic "common code".

```
kNNClassifySuperCode(Res, k, k_c, t_c)
 1  (dist, img) ← Res[1]
 2  if dist / (width(img) * height(img)) > t_c
 3     then
 4          newCode ← Λ(img)
 5          for i ← 2 to k_c
 6               do
 7                    (dist, img) ← Res[i]
 8                    newCode getsSuperCode(newCode, Λ(img))
 9          return newCode
10     else
11          return kNNClassifyWeighted(Res, k)
```

Pseudo Code Listing 10.10: Classification using Supercode or the weighted kNN classifier

10.2.4 Evaluating Errors in Hierarchical IRMA Codes

As described in Chapter 2.2.2 on page 32 and detailed in Equation (2.6), one essential measure for the quality of classification is the error rate. This would simply check whether the assigned label for an image matches the ground truth and count all instances where this was not the case (*false negatives* F^- and *false positives* F^+ in case of binary classifiers that accepts/rejects a single class) and compute the ratio of all images evaluated.

The error rate can be computed for the IRMA code, but this would mean that no matter at which position in an axis the error is made, it would be considered equally bad. Furthermore, it wouldn't differentiate whether only the code for one axis or for all four axes was wrong or if there have been even several mistakes in multiple levels of the same axis. Last but not least, any simple evaluation scheme using error rates by checking only for identical labels cannot capture the semantics of wildcards like "*": Based on the definition of that wildcard on page 210, it can only occur in the assigned label and not in the ground truth; hence, assigning wildcards in case of uncertainty would be considered as bad as just guessing wrong.

In order to evaluate the hierarchical code without these undesirable properties, a different error scoring theme has been proposed for the ImageCLEF 2007 medical annotation benchmark task [Deselaers et al., 2008a], which computes the score per axis of the code and aggregates the score. The score is normalized such that any code where not a single position in the code matches gets an error of 1, if the entire code is correct, an error of 0. Furthermore, the branching factor and therefore the difficulty to "guess" the code correctly at each level of the code is considered, as well as the wildcard character that is assigned 0.5 times the maximum error at this position – thus rewarding if a classifier specifies a wildcard when uncertain rather than simply guessing wrong but

still providing enough incentive to propose a class if this is correct with a fairly high probability.

In order to develop, tune and benchmark several approaches, three sets of images has been released for ImageCLEF 2007:

Training Set: A set of 10'000 images of which the correct IRMA code is known, such that the Λ function as declared in Equation (10.7) can get implemented.

Development Set: A set of 1'000 images with class labels that can be used to develop and tune the approaches. Participants of the benchmark can evaluate their approach with this set before handing in any results to avoid situations in which misconfiguration harmed the results.

Test Set: Another set of 1'000 images that is used for benchmarking. The correct labels are only announced after all participants handed in their results for the benchmark.

In theory, their would be no need for a dedicated development set as one could always use a subset of the training set, and e.g., perform a k-fold cross-validation. However, by providing a specific set of images as development set, it is easier to compare also the results achieved on the development set and –in combination with the results achieved on the test set– may better allow to identify tendencies rather than just the single result achieved with only one particular validation set.

In order for these results to be meaningful, all sets should be similar w.r.t. on which classes are present, the kind of deviations between images (intra- and interclass variability), and the distribution of classes.[11] As shown in Figure 10.6, the three sets show a very similar distribution. In total there are 116 unique classes / IRMA codes; all of them present with at least 10 images in the training set, one of them not present in the test set, six of them not present in development set. None of the classes not appearing in the test or development set has more than 11 instances in the training data set. The maximum number of instances of one class found in the development set is 1927 of the code 1123-127-500-000[12], median number of instances is 37.

10.2.5 Effectiveness of Classification

Using these sets of images and labels, we can experimentally evaluate the effectiveness of the classifiers. As both sets for evaluation, development and test set, contain 1'000 images and the maximum error score per image is 1.0, the worst aggregated score for an run over an entire set is 1'000.[13] At the same time, if only wildcard characters were used, the error score would be 500 and therefore any helpful approach to automatic

[11] In other words: Assuming any set of images is generated by sampling from a distribution function, all three sets should sample the same distribution function.

[12] 1123-127-500-000 = technique: x-ray, plain radiography, analog, high beam energy; direction: coronal, anteroposterior (AP, coronal), supine; anatomy: chest, unspecified; biosystem: unspecified

[13] Notice than when only the 116 classes present in the training set are considered, the worst score per image is not 1.0 but only about 0.916 as all images in the sets are X-ray images and therefore the first digit of the technology axis always remains "1", so by picking any of the 116 classes, this can never be wrong.

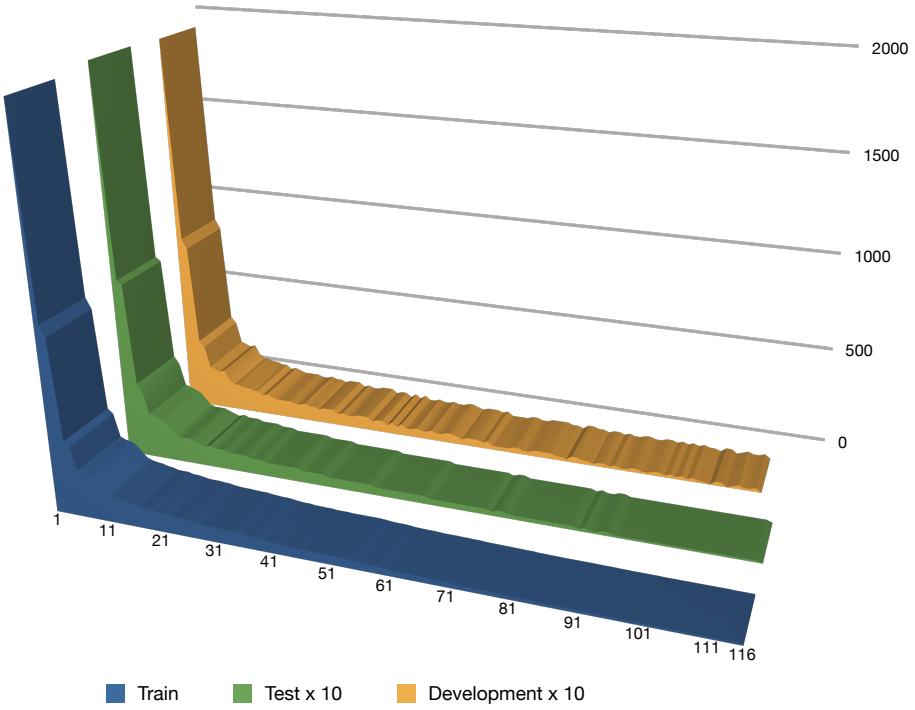
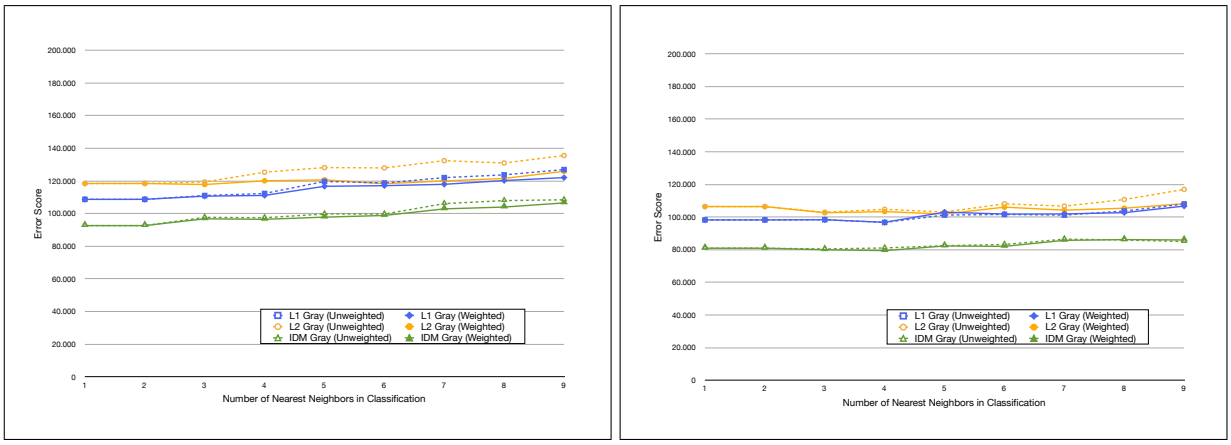


Figure 10.6: Distributions of images to IRMA codes: The number of instances for each of the 116 classes, sorted on the number of instances in the training set. Numbers of instances in Test and Development set have been multiplied with the factor 10 to compensate for the smaller overall set size. *(IRMA codes of classes not shown, just numbered. Bars shown in 3D as they overlap so closely, that in common 2D line or bar charts, the lines are basically indistinguishable.)*

image classification has to achieve an aggregated error score well below this value.[14] As some classes have in total only ten instances in the training set, any classification based on nearest neighbors would be biased towards classes with many instances if k is set too high. Even with values of $k < 10$, we can observe the tendency that smaller values seem to perform better.

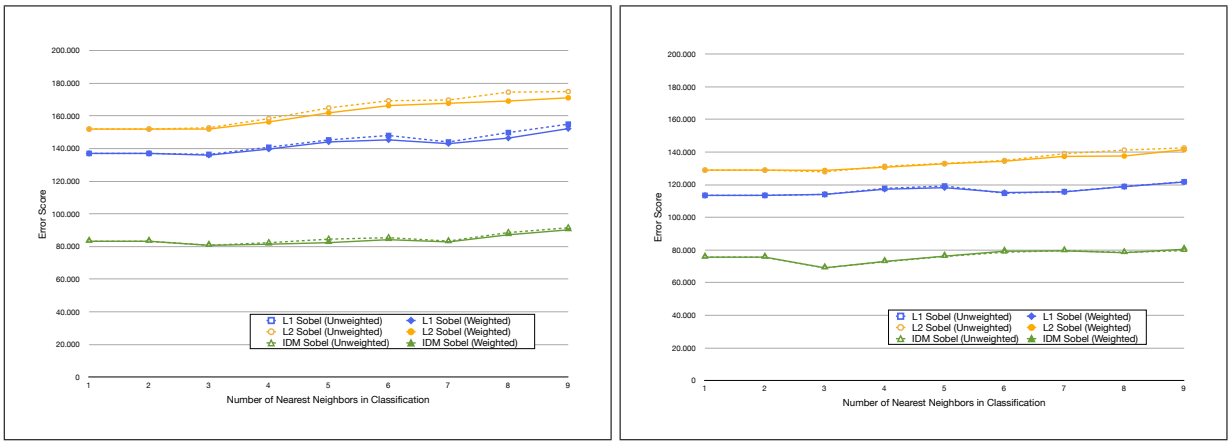
Figure 10.7 shows the results when using only the gray intensities of the images scaled to 32 pixels at the longer edge. As distance measures, IDM and variants of L_1 (Manhattan distance) and L_2 (Euclidean distance) are used that perform the same computation for corresponding pixels as IDM. IDM was used with a warp range w of 2 and a local context lc of 1. The error score has been computed on both, the development and the test set. The solid lines show the result when the weighted variant of the k Nearest Neighbor Classifier as shown in Pseudo Code Listing 10.8 is used; the dotted lines

[14] If there very only two classes which do not share any code along the four axes, guessing randomly should also achieve a score of 500. With a total of 116 different classes, guessing randomly has to be expected achieve a score significantly above 500. In this particular setting, when picking randomly any of the 116 classes, empirically we observed an average score of 590.404 when performing 100 runs each of the two evaluation sets. The score was reduced to 547.312 when the class distribution on the training set was considered instead of a uniform distribution of the classes.



(a) Development Set (b) Test Set

Figure 10.7: Results using only gray intensities for L_1 (blue), L_2 (orange), and IDM with warp range of 2 and local context of 1 (green): Dotted lines show the results when using the unweighted classifier as in Pseudo Code Listing 10.5, solid lines the weighted variant of Pseudo Code Listing 10.8. The scores achieved on the development set in (a) are in general worse (higher values) than on the test set in (b).



(a) Development Set (b) Test Set

Figure 10.8: Results using only gradients for L_1 (blue), L_2 (orange), and IDM with warp range of 2 and local context of 1 (green): Dotted lines show the results when using the unweighted classifier as in Pseudo Code Listing 10.5, solid lines the weighted variant of Pseudo Code Listing 10.8. In most cases, the scores achieved on the development set in (a) are again worse (higher values) than on the test set in (b). The exception is L_2 which performs very poor on the test set.

show the result when Pseudo Code Listing 10.5 is applied. As the implementation of Pseudo Code Listing 10.5 assures that in case of equal number of votes the class with lower distance is selected, the results for $k \leq 2$ are identical with the weighted version of Pseudo Code Listing 10.8. For $k > 2$, in the vast majority of cases, the weighted classifier of Pseudo Code Listing 10.8 outperforms the unweighted version. However, this

effect is much more visible for L_1 and L_2 with $k \geq 5$ than it is with IDM – which generally leads to lower scores and therefore better results. Similar results can be observed in Figure 10.8: IDM performs better than L_1 and L_1 still outperforms L_2. In most cases, the weighted classifier performs better than the unweighted version.

Before continuing the discussion, we want to remind of the error counting scheme and give some more intuitive understanding to the numbers: The worst score found in Figure 10.7 and 10.8 is 174.900 for L_2 with a unweighted classifier with $k = 9$ using only gradients on the development set. The best score found is 69.031 for IDM with a weighted classifier with $k = 3$ using gradients on the test set. Theoretically, this could be achieved by guessing only very few wrong class labels for images with a maximum error score of 1.0, thus having an error rate of only 6.9% to 17.4%. In reality, most misclassified images result in error scores significantly below 1.0 because only some digits along some axes are wrong. The worst score corresponds effectively in an error rate of 41.5%, the best score corresponds to an error rate of 20.9%. Any improvement of the error score can either be achieved by either reducing the number of misclassifications or –in case of misclassifications– make mistakes that result in a smaller error score as the classes are closer related in the hierarchical structure. In contrast, error rate would only honor the first kind of improvement – avoiding misclassifications. When comparing two error scores or lines in the graphs, small differences of less than 1.0 in the error score could be subject to just a small number of the 1'000 images in the evaluation set being affected – in the extreme case only a single image. As the sets are realistic images from clinical practice and not a synthetically generated set, the effect could always be caused by the quality of the image rather than the approach itself as some images may be "misleading" / not ideal cases to represent a class. However, if a similar trend is found on two independent sets of 1'000 images and robust against some minor changes of parameters, chances are not bad that the difference is caused by one approach performing better than the other. And when applying such an approach to real world images, also not all images will be ideal.

Effectiveness of Gray Intensity, Edges, and Weighted Combinations

As illustrated in Figure 10.4 and 10.5, there are several possibilities to derive edge information from the images for searches; even if the same edge detector (Sobel filter) is used, the image processing pipeline can have some impact on the features and the search results. Figure 10.9 shows the results for three different variants:

- *Combined:* Use a single layer in which both, vertical and horizontal Sobel filter results have been aggregated as the gradient magnitude.[15] This variant is shown in Figure 10.9 as "Sobel (Combined)".

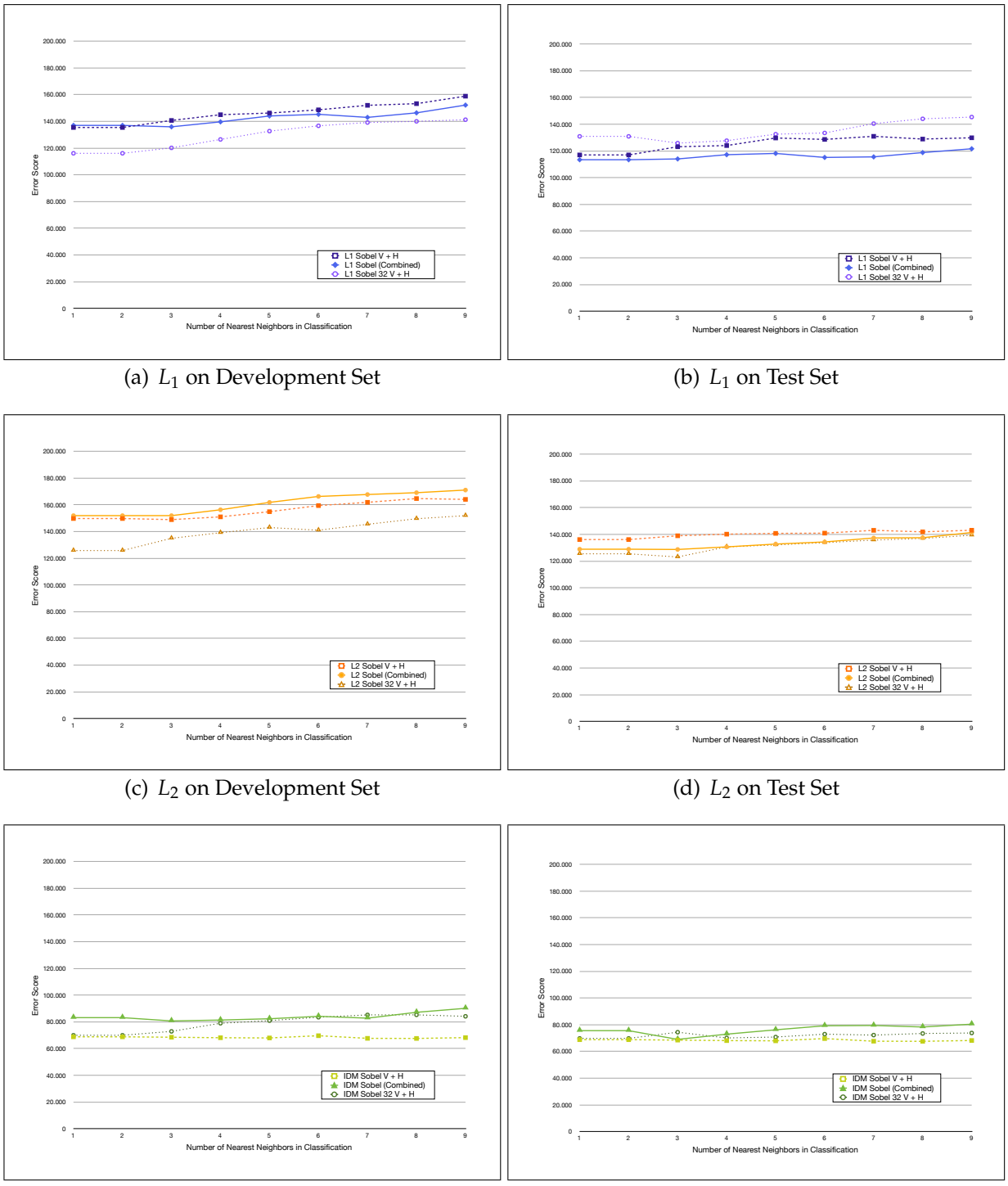
- *Two Separate Layers:* Use separate layers for the edge information from the horizontal and vertical Sobel filter. The distance will be computed based on pixel positions on both of these layers and minimized jointly in IDM. For the Minkowski Norms L_1 and L_2 there is no need to "optimize" the distance and computation can

[15] This corresponds to the formula $|G| = \sqrt{Gx^2 + Gy^2}$ rather than the approximated magnitude $|G| = |Gx| + |Gy|$ in [Fisher et al., 2004] on `http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm`.

be performed as if two vectors would be concatenated. For IDM, such a simple approach would no longer preserve the desired interpretation of a deformation / displacement of pixels. Therefore it is preferable to use separate layers in which each pixel will have one value per layer similar to common color models like RGB channels; optimization will find the displacement of pixels and it's local context that is optimal with respect to all layers instead of displacing the pixel differently on different layers. One variant shown in Figure 10.9 that uses this setup and otherwise is identical to the combined variant is "Sobel V + H".

- *Order of Processing Pipeline:* Either first shrink the image to the desired size, then apply the Sobel filters or first process the filters and then shrink the image. Due to the different amount of information preserved and image artifacts of the interpolation of the picture, result in different values as described on page 214 and shown in Figure 10.4 and 10.5. In order to analyze the effect of the processing pipeline, Figure 10.9 has the variant "Sobel 32 V + H" that first scales down the images to at most 32×32 pixels and then applies the Sobel filters and stores the results in different layers. Both other variant apply the Sobel filters first and then scale to the desired size.

Figure 10.9 shows the results on the development and test set. As observed on Figure 10.8, IDM performs in general significantly better than L_1 and L_1 performs in general better than L_2. The three different distance functions have been split on separate graphs to reduce clutter. On the development set, scaling down the images first and then applying the Sobel filter performs slightly better than first filtering than shrinking – even though that this corresponds to a higher loss of information detail. The reason for this behavior seems to be that some (otherwise) misleading information from some of the images of the set was removed as this behavior did not occur to the same amount on the test set – or was even reversed for L_1.



(a) L_1 on Development Set

(b) L_1 on Test Set

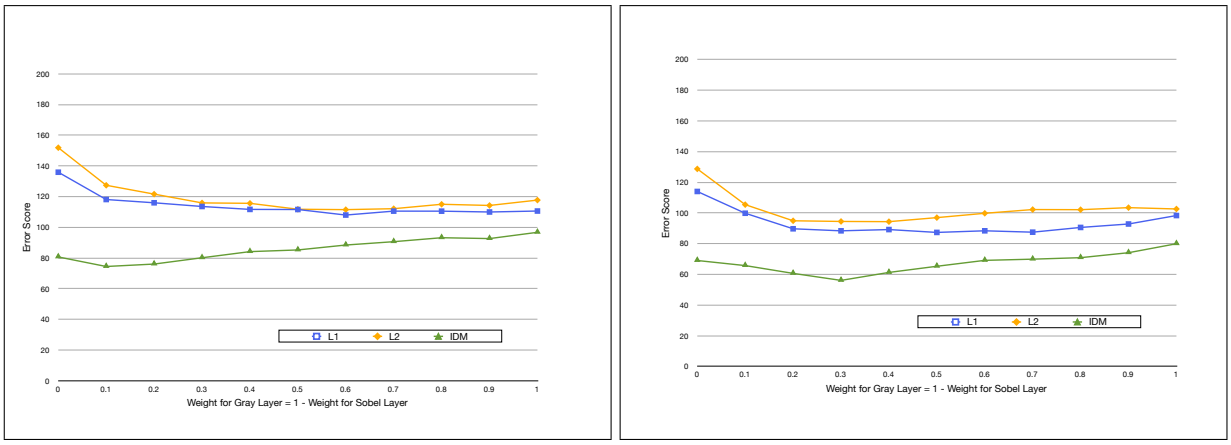
(c) L_2 on Development Set

(d) L_2 on Test Set

(e) IDM on Development Set

(f) IDM on Test Set

Figure 10.9: Results using different variants to compute gradients for L_1, L_2, and IDM with warp range of 2 and local context of 1 (green): *Combined* uses the a single layer that combines the horizontal and vertical Sobel filter; $V + H$ uses two separate layers – one for the result of the vertical Sobel filter and one for the horizontal filter. In both cases, scaling to the maximum size of 32 pixels for the longer axis is performed after the filtering was performed with interpolation. In contrast, *32* $V + H$ scales down the images first and applies then the Sobel filters. All results use the weighted kNN classifier based on Pseudo Code Listing 10.8.



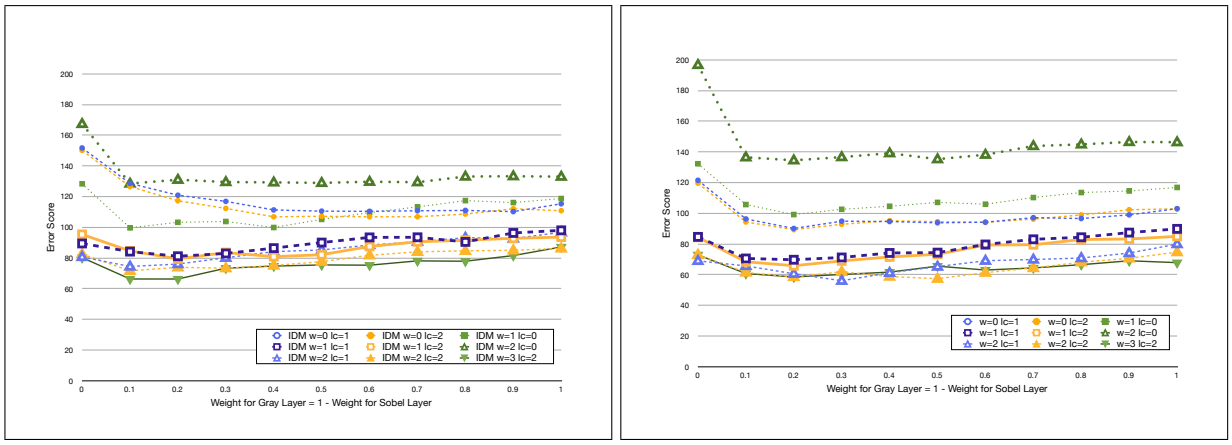
(a) Development Set (b) Test Set

Figure 10.10: Results of using a combination of gray intensities and gradients with L_1 (blue), L_2 (orange), and IDM with warp range of 2 and local context of 1 (green) for weighted classifier with $k = 3$: Combinations depend on the weight assigned to the two layers with $w_I + w_S = 1$.

The weighted variant of the kNN classifier was used, best performance was commonly achieved for k either 1, 2, or 3 with 1 and 2 delivering identical results as described on page 224. $k = 3$ sometimes outperformed smaller values, but in general, showed similar performance while being more robust w.r.t. the choice of the variant of applying the Sobel filter: In the graphs, the points at $k = 3$ are in most cases closer together than for $k < 3$. Using the combined version with a single layer did achieve very similar results as using separate layers with the biggest difference found for IDM - where this also results in different layers for the local context and therefore preserves the orientation of edges rather than just their intensity. However, even for IDM the results at $k = 3$ did not differ very much; considering the fact that storing the information on separate layers in general comes at the cost of twice the storage needed and also performing twice as many computations to determine the distances, these added costs may not appear justified by the small gains that appear achievable.

In contrast, using two layers with different sets of informations seems more interesting: This allows to adjust the use of image intensities and gradients at the same time. However, in contrast to using two layers of vertical and horizontal edge information, the particular values on the two layers will have very different meaning and as they have been generated in different ways, it would be mostly coincidental if the best combination would equally weight both layers.

Figure 10.10 shows the results for varying combinations of weights. Let w_I be the weight for the distance on the layer of gray intensities and w_S the weights for the gradients on the layer on which the combined Sobel filter was applied. We set $w_I + w_S = 1$, therefore the extreme positions in Figure 10.10 correspond to only the gradients ($w_I = 0, w_S = 1$) and only the intensities ($w_I = 1, w_S = 0$). As can be observed easily, combinations of weights perform better than any of the layers alone.



(a) Development Set (b) Test Set

Figure 10.11: Results of using a combination of gray intensities and gradients with IDM for varying values of warp range w and local context lc for weighted classifier with $k = 3$: Combinations depend on the weight assigned to the two layers with $w_I + w_S = 1$.

Fine-Tuning IDM

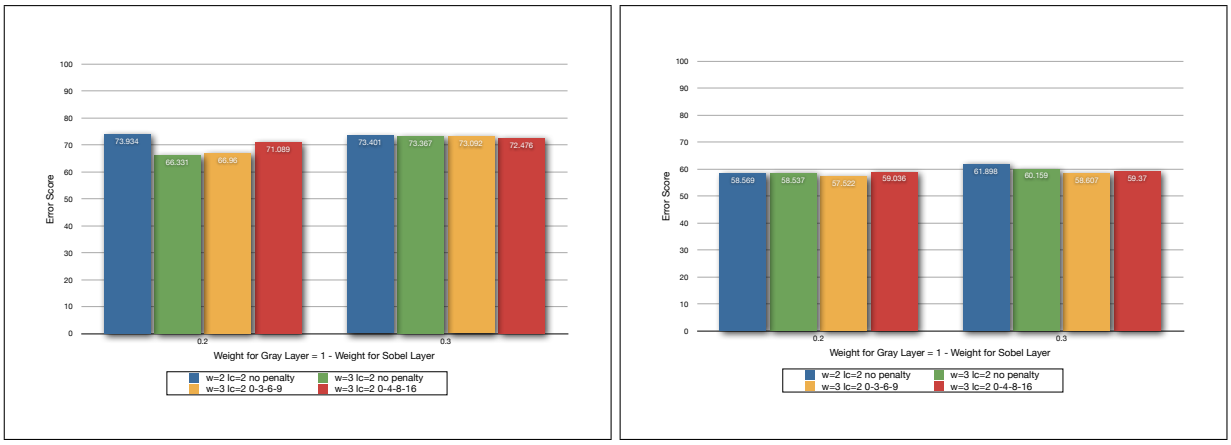
IDM constantly outperforms the more simple distance measures L_1 and L_2. As our implementation differs mainly in the matching tolerance[16], we investigate further what makes IDM perform better as there are some parameters that can be tuned. Figure 10.11 shows the same results for varying values of the warp range w and the local context lc. Notice, that greater values of w *increase* the matching tolerance as pixels are allowed to be further displaced. In contrast, greater values of lc *decrease* the matching tolerance as greater patches have to matched, thus adding more constrains to matching.

Very poor results are achieved for $lc = 0$, with higher warp ranges delivering worse results. The results are very poor as they are even worse than L_2 in Figure 10.10. Therefore fairly unconstraint matching does not lead to good results.

The significantly better results are achieved with combinations $w \in \{2,3\}$ and $lc \in \{1,2\}$; with the overall best result on the development set being achieved by $w = 3$ and $lc = 2$ which also performs well on the test set. However, on the test set, the overall best score is achieved with $w = 2$ and $lc = 1$ for the weight $w_I = 0.3$ – for other values of w_I, $lc = 2$ with $w \in \{2,3\}$ still perform better. Thus, it is the combination of w and lc that deliver an appropriate matching tolerance for this problem of medical image classification.

In order to fine-tune the deformations within the warp range, a cost function as presented in Equation (10.4) can be used to penalize displacements to pixels which are far away over near ones. Figure 10.12 shows the results for IDM with $w_I \in \{0.2, 0.3\}$, the combinations for which IDM delivered best results in Figure 10.11. The local context has been kept fixed at 2 to display only the effect caused by the costs for displacement. "No penalty" indicates that any displacement was considered equally well; "0-3-6-9"

[16] L_2 is essentially IDM with $w = lc = 0$



(a) Development Set (b) Test Set

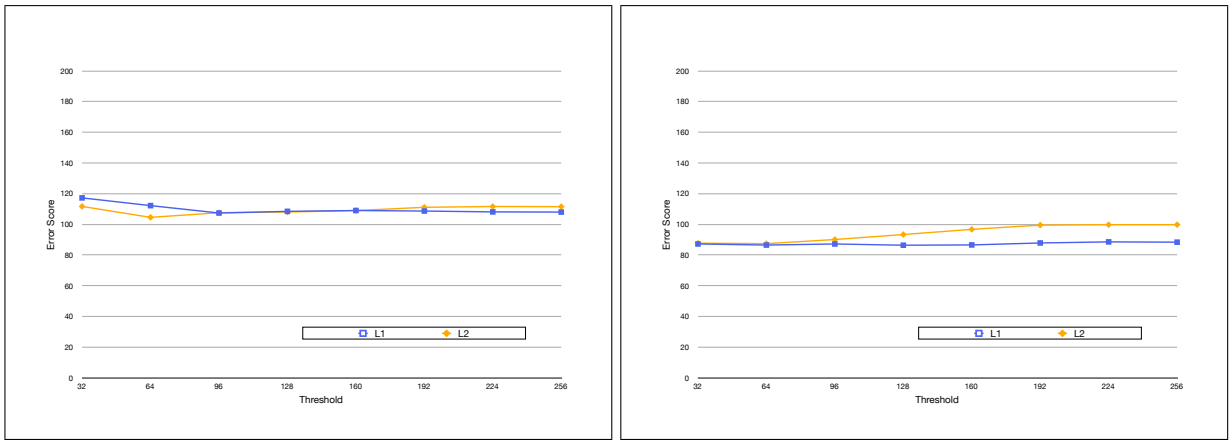
Figure 10.12: Results of adding a penalty based on the costs for displacements presented in Table 10.1 for IDM with a local context $lc = 2$.

indicate that A_{369} and M_{369} from Table 10.1 was used, for "0-4-8-16" it was A_{4816} and M_{4816}, respectively.

If we look back at the performance of the simpler distance measures L_1 and L_2, we see that the matching tolerance is very limited due to the inability to cope with deformations. The matching tolerance can get adjusted by using a threshold on the impact of individual pixels as presented in Equation (10.5) on page 212: Such a threshold makes the approach less sensitive to strong local differences that can be caused by small displacements that IDM would compensate by deforming the image. As differences on the level of individual pixels are computed based on the gray intensities or gradients which are in our implementation captured as 8 bit integer values, the difference for any two pixels will always be at most 255. If both layers are used, intensities and gradients, each layer can have at most 255 and we should use the weights for the layers as in Figure 10.10.

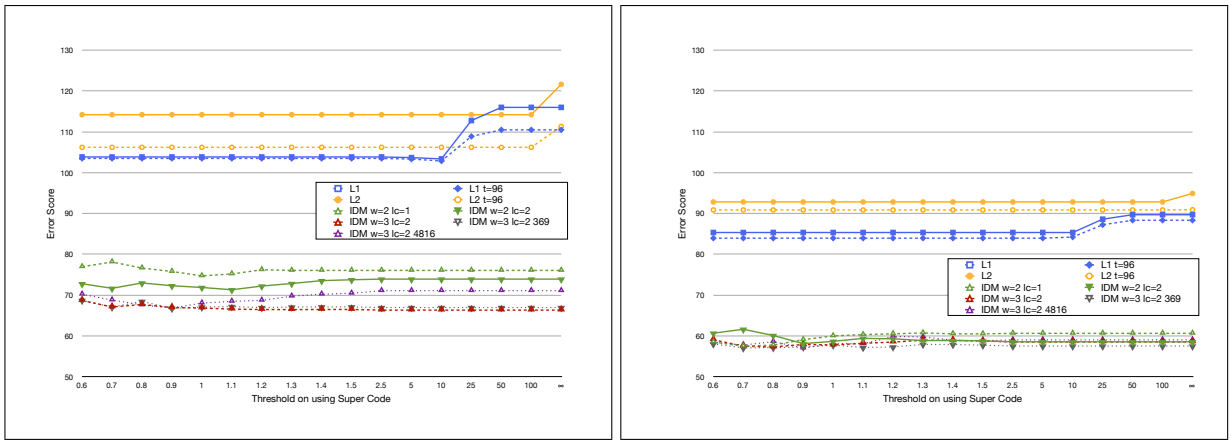
Figure 10.13 shows the results for L_1 and L_2 when thresholds are used. Both layers, gray intensities and gradients, are used with $w_I = 0.6$ and $w_S = 0.4$ – which is a choice at which both distance measures performed well in experiments on the development set and validation with the test set confirms this finding. Thresholds are displayed for increments of 32; the highest value 256 is equivalent to not using a threshold and therefore shows the same results as Figure 10.10. For other weights, similar results are achieved: For most threshold values, the results are better than when applying no threshold; the exceptions being very low thresholds of 32 and 64 on the development set – which performed worse than not applying a threshold as very low thresholds remove the discriminating power of the distance measure.[17] L_2 benefits more from applying a threshold than L_1, getting very close to the results of L_1. This can be explained with the fact that L_2 is more sensitive to differences in individual values as it takes the squared

[17]This becomes even more apparent when considering the extreme case: At a threshold of 0, every difference would exceed this value, hence the distance measure would return the same distance score for all image comparisons as it is dominated by the threshold and therefore entirely independent of the images compared.



(a) Development Set (b) Test Set

Figure 10.13: Results of using thresholds on the difference between individual elements in the vectors of gray intensities and gradients for L_1 and L_2 with $w_I = 0.6$ and $w_S = 0.4$.



(a) Development Set (b) Test Set

Figure 10.14: Results of using Pseudo Code Listing 10.10 with $k_c = 2$, $k = 3$, $w_I = 0.2$ and $w_S = 0.8$ for different distance measures and thresholds t_c. *(Error Score is only displayed in range 50 – 130, Y-axis doesn't start at 0. X-axis displays selected values for t_c in range 0.6 – ∞.)*

difference rather than the absolute difference in consideration. The results presented in Figure 10.13 show that a threshold of 96 performs always better than not using a threshold.

Exploiting Knowledge about the Hierarchy

So far, the all experiments did not exploit the knowledge about the hierarchical code. Pseudo Code Listing 10.10 on page 221 implements a hybrid solution that depending on the threshold t_c switches between generating the supercode of k_c nearest neighbors and a weighted kNN classifier. Figure 10.14 shows the results for $k_c = 2$, meaning that

the supercode was only (if at all) generated from the two nearest neighbors. Values of $k_c > 2$ resulted in overgeneralizing the supercode in most cases, therefore assigning a label in the hierarchical code that is less specific and therefore delivering higher Error Scores than using just the weighted kNN classifier that "guesses" a specific label from the code. In this experiment, w_I was set to 0.2, therefore w_S was 0.8, meaning that four times more emphasize was given to the edges generated with the Sobel filter than on the gray intensities. Similar results are achieved also for other values; this particular value has been selected as it delivered good results in Figure 10.11. Notice that the Y-axis in Figure 10.14 does not start at 0 and ends at 130 instead of 200, as it was in previous results, as some results are very close and even hard to see on this reduced range of values. In line with the idea behind the implementation of Pseudo Code Listing 10.10, strongest improvements are achieved where the kNN classifier alone –which is equivalent to $t_c = \infty$– would deliver unreliable results. This is also due to the fact that if the nearest neighbors agree on a class, generating the supercode would not differ at all in its result from using either the weighted or the unweighted kNN classifier.

For any distance measure, setting the threshold t_c too low will result in a supercode which is too generic and therefore resulting in higher Error Scores. This is best visible in Figure 10.14 for all used settings of IDM with very low values of t_c, e.g., $t_c = 0.6$. Setting the threshold too high will result not using the supercode at all, therefore achieving identical results as the weighted kNN classifier. This is visible in Figure 10.14 for IDM for $t_c \geq 2.5$ and for L_1 for $t_c \geq 50$. Between these extremes, the generated supercode can outperform the weighted classifier.

As the value of the distance depends strongly on the used distance measure, values of t_c in IDM have to take into account that IDM averages the distance over the local context, hence delivering significantly smaller values in particular when strong impact is given to the Sobel-filtered layer. Additionally, IDM optimizes and therefore reduces the absolute values of distances inside the warp range – another difference compared to L_1 and L_2. When comparing L_1 and L_2, as L_2 computes the squared differences instead of the absolute differences, also the values for t_c need to be significantly higher.

Official Participation to ImageCLEF 2007

We submitted runs to the ImageCLEF 2007 benchmark competition for which we used an Image Distortion Model with a warp range of $w = 3$ and the two different combinations of the cost matrices presented in Table 10.1, A_{369} for additive costs and M_{369} as factors and A_{4816} together with M_{4816}. Notice that at the time of submission, the correct classes of the images in the test set was not known to the participants and only limited time between the release of the dataset was available to tune the system based on the development set. For these runs, the gray intensities was given twice the importance as the Sobel-filtered layer, therefore $w_I = \frac{2}{3}$ and $w_S = \frac{1}{3}$.

The combination 4816, which assigns higher costs, performed a little better when using only a 5×5 pixel area ($l = 2$) as local context. For $l = 3$ we submitted only a single run using the 369 matrices – which turned out to be the best of our runs. A "c" appended to the number of k nearest neighbors indicates that the IRMA-code aware

classifier was used with the two nearest neighbors generating a common code, if the normalized distance was below 1.0.

As a baseline, the run with label (a) was performed using only IDM with $w = 2$ and filtered by a sieve function with 500 candidates based on the Euclidean distance ("Sieve 500L2"). The sieve function was proposed for the use with IDM in [Thies et al., 2005], not to improve classification quality, but to reduce execution time. In contrast to [Thies et al., 2005] which uses L_2 on the gray images reduced to 32×32 pixels, we used the same features (images scaled to 32 pixels the longer side, preserving the aspect ratio; both layers, therefore gray intensities and gradients with $w_I = \frac{2}{3}$ and $w_S = \frac{1}{3}$) and a per-pixel threshold of $t = 96$ in Equation (10.5) and for L_2. (b) uses the same parameters except for the Manhattan distance as the filter ("Sieve 500L1"). The column improvement displays the relative improvement compared to the baseline run (a).

Table 10.2: Scores of runs on the ImageCLEF2007 MAAT test data set

Parameters	Score	Improvement	Label
$W = 2$ $LC = 1$ $k = 1$ + Sieve 500L2	66.50		(a)
$W = 2$ $LC = 1$ $k = 1$ + Sieve 500L1	66.34	0.24%	(b)
$W = 2$ $LC = 1$ $k = 1$	66.17	0.50%	(c)
$W = 2$ $LC = 1$ $k = 5$	65.45	1.58%	(d)
$W = 3$ $LC = 2$ $k = 5c$ 369	65.09	2.12%	(25)
$W = 3$ $LC = 2$ $k = 3$ 369	63.44	4.60%	(24)
$W = 3$ $LC = 2$ $k = 3c$ 369	62.83	5.15%	(23)
$W = 3$ $LC = 2$ $k = 5c$ 4816	61.41	7.65%	(22)
$W = 3$ $LC = 2$ $k = 3c$ 4816	60.67	8.77%	(21)
$W = 3$ $LC = 2$ $k = 3$ 4816	59.84	10.02%	(20)
$W = 3$ $LC = 3$ $k = 3$ 369	59.12	11.10%	(e)
$W = 3$ $LC = 3$ $k = 3c$ 369	58.15	12.56%	(19)

The achieved scores presented in Table 10.2 show that:

- Increasing the warp range and local context improves retrieval quality.
- $k = 3$ outperforms $k = 5$. Further experiments showed that $k = 1$ is inferior when the inverse square of the distance is used.
- Expressing uncertainty in the IRMA-code pays off only when a threshold on the distance is set very carefully. In our best case, this improved the score by 1.6%. For the combination A_{4816} / M_{4816} the threshold on the normalized distances of 1.0 was not ideal.
- Using the sieve function proposed in [Thies et al., 2005] slightly degrades retrieval quality.
- Overall the score was improved by 12.56% from 66.50 to 58.15.

In total, 68 runs have been submitted for this task in ImageCLEF 2007 [Deselaers et al., 2008a]. Out of these, our best run achieved rank 19, our worst run that we have submitted achieved still rank 25. The labels for runs in Table 10.2 that use only numeric values indicate the rank which they achieved in the official ranking of all 68 runs.

It shows that using the fairly basic implementation of search primitives can already achieve acceptable results. Fine-tuning the *Matching Tolerance* by adjusting the warp range and local context, the costs for displacements, the number of nearest neighbors and weighting in classification, and the threshold to generate a supercode in case of high uncertainty can improve the quality of classification.

If better results are needed, in line with the idea of exchangeable building blocks, the basic implementation can get replaced with one that is more tailored towards the task; for instance with one of the approaches that use also SIFT-features and a support vector machine (SVM) that performed best in the ImageCLEF 2007 Medical Image Annotation Task [Deselaers et al., 2008a, Tommasi et al., 2008a]. That approach used bag-of-words approach with modified SIFT descriptors extracted at random sampling points of the medical images and intensionally are extracted only at a single octave to drop the –for this task undesirable– scale-invariance and also dropped the rotation-invariance. Best results were achieved with a SVM that integrated these local features with a reduced representation of the image [Tommasi et al., 2008a].

10.3 Sketch-Based Known Image Search

10.3.1 On Query by Sketch as a Strategy to Solve Image-Related Search Tasks

Query by Sketch (QbS) as such is a strategy in searching for images, therefore just one tool amongst others that the user can pick to perform a search task. Before going into the details of our approach, we therefore want to quickly recapitulate on some aspects of the various tasks we have analyzed in Task Input and Aim (Chapter 2.2) and how they relate to QbS.

Known Image Search

A fairly common case of searching for images has been described in Chapter 2.2.1: The search for known images where the image itself is known, but not available as an image itself to user at the moment as its storage location as well as other meta information may be forgotten at the time of search. Chapter 2.2.1 on page 29 mentions already that *Query by Sketching (QbS)* can be a powerful tool to assist the user in such situations; partially due to the fact, that it is one of the tools that remains applicable when the prerequisites for other strategies are unavailable, e.g., no image available for Query by Example (QbE) or not enough metadata like annotations or tags available or known for the sought image.

Classification

In a generic setting, Query by Sketch could theoretically be used for Retrieval by Class (Chapter 2.2.2) as soon as the class to which the query sketch belongs has been identified correctly. In practice however, selecting a class based on a sketch is highly unreliable and inefficient for great majority of concepts.

In fact, it's so inefficient and unreliable, that there is a well known game concept based on this situation known as "Pictionary". In this game, each turn starts with a person whose turn reading a word from a card (which takes very little time) and then starting to draw something on a board. The drawing should either represents the concept behind the word or –if that seems easier– provides hints for the other game participants what the concept might be, e.g., by illustrating another concept with a name that sounds similar or represents part of the word that is written on the card. The other participants in the game have to guess what the word was, therefore shout out what it could be and the person drawing stops whenever the word from the card was said. Again: Shouting the word takes very little time as well as listening to it and signaling whether it was the correct word. The aim of the game is that the audience guesses the word as fast as possible. Would sketching the concept be as efficient as saying the word aloud or writing it down and wouldn't sketching lead to false interpretations, there would not be any fun in playing this game.[18]

Sketches can still be very effective to illustrate concepts in restricted domains. For instance, sketches or sketch-like gestures can be used as shorthands for particular concepts – a technique not only helpful in searching, but also in many aspects of communication, like warning signs or technical drawings to describe the package contents and illustrate how to assemble furniture.

In [Shrivastava et al., 2011], sketches are used to retrieve either cars or bicycle images from the PASCAL VOC dataset [Everingham et al., 2010]; however, the retrieval is not a pure retrieval by class as the article emphasizes that the top retrieved images show "the target objects in a very similar pose and viewpoint as the query sketch". That property is mainly attributed to known image search and may also apply to themed searches. In classification, any member of the correct class should be considered equally good; if pose and viewpoint are relevant to be a correct result, this requirement has to be part of the class definition.[19] Elastic template matching as in [Del Bimbo and Pala, 1997] can also be used as to assist object detection based on a user sketch, for instance to detect bottles in paintings given a template sketch.

Another approach to classify user-drawn object sketches is described in [Eitz and Hays, 2011], which reports an accuracy of 37% for selecting the right category out of 187 objects

[18]Which actually, can be done using computing devices - for instance online at `http://www.isketch.net` or attaching the uDraw devices by video game publisher THQ to popular game consoles: `http://www.thq.com/us/udraw-gametablet/` or the (temporary) hype smartphone game DrawSomething: `http://omgpop.com/drawsomething`.

[19]This "moving the pose and viewpoint into the class definition" is a basic assumption of *Exemplar-SVM* [Malisiewicz et al., 2011] which intentionally deviates from traditional object category assignment towards similarity measurement through large-scale negative mining. [Shrivastava et al., 2011] use such Exemplar-SVM for discriminative training with a single positive example and randomly selected subset of the images in the dataset which are used as negative examples – independent of how similar this random sample turns out to be.

categories of the LabelMe dataset. This is a good achievement considering the difficulties even human participants to the game Pictionary have. However, it is still far below anything that would commonly be achieved by letting the user either enter the class name using a keyboard or speech recognition or select the class from a list of class names or representative images from each class – and drawing such a sketch will certainly take considerably more time. This would even more be the case for [Schindler and Suter, 2008, Ferrari et al., 2010], where only 5 and 9 different object classes have been used. In latter two approaches, sketches can be used, but the main emphasize lies not in sketched-based retrieval, but the detection and localization of instances of the object classes based on a learned shape model. In [Vuurpijl et al., 2002], outline sketches are used to retrieve human annotations that are also associated with outline sketches in scanned historical paintings. Also here the reported accuracy for 4 different query classes (tree, human, table, horse/donkey) within the top k ranked results varied between 98.3% (horse, 520 samples, $k = 1$) and 53.5%(tree, 3000+520 samples, $k = 10$), thus being far below the 100% accuracy if the user had selected the concept out of a list of four entries[20]. The main added benefit in [Vuurpijl et al., 2002] was the involvement of users to annotate content. Also in the evaluation descriptors for content-based sketch retrieval in [Heesch and Rüger, 2002] a classification scenario is used: The test dataset consists of 238 black and white sketches from 34 categories. Relevance is defined by retrieving images that belong to the same category, thus being a classification task. Precision varied for the best single descriptor (Fourier descriptor) between 70% (lowest recall) and 15% (recall 100%); combining several features resulted in a moderate increase of precision with a precision of 72% (lowest recall) and 19% (recall 100%). Precision values are not directly translatable to classification accuracy, however, using a 1-NN classifier as a baseline, the precision at lowest recall usually corresponds closely to 1 - error rate (cf. also Chapter 2.2.2). Thus, also this report indicates classification accuracy that is significantly below 100% and therefore as another example, that sketches are not very reliable as class input in retrieval.

Themed Search

In Themed Search as described in Chapter 2.2.3, the user has an idea about what the ideal image would look like – independent of whether such an image exists or not. The aim is to find at least one image that suits the preferences of the user well enough.

One critical issue in applying Query by Sketch for such tasks is, that the mental image is frequently centered around concepts like objects, people, animals and actions that are performed in the image as well as emotions that are caused by viewing such images. As mentioned in Chapter 2.2.3, [Jörgensen, 1998] describes perceptual, interpretational, and reactive attributes that have to be considered in such tasks. Sketches alone can usually not capture all of them adequately and even of those parts that can be expressed through sketches, the user might frequently run into similar problems in communicating a concept through a sketch as for Retrieval by Class. An extreme situa-

[20] And assuming that all human annotations are accurate. [Vuurpijl et al., 2002] reports that within their quality control of the web-based system used in a museum context, 95% of the annotations were "cooperative".

tion to illustrate the problem can frequently be experienced when a young child draws an image and presents it to an adult: The adult responds with something like "Oh, that's beautiful! What is it?" – showing the problem of understanding a painted scene just from a sketch. In this particular instance very prominently, since the person drawing the picture radically simplifies concepts also because the needed skills and methods to draw the concepts better have not yet been acquired.

Nevertheless, the popular saying "a picture is worth a thousand words" is true for many Themed Search tasks, in particular to describe spatial arrangements. There are at least two recent approaches, Sketch2Photo [Chen et al., 2009] and Photosketcher [Eitz et al., 2009b, Eitz et al., 2011b], that take user-drawn sketches as an input to perform a particular kind of Themed Searches: Photo montage, which is a task to compose new images from existing ones as described in Chapter 2.2.3 on page 41. Sketch2Photo does not only consider the user's sketch to identify the desired concepts in the image, but relies on textual information provided by the user to achieve the better accuracy for this task. Therefore it does not use just the strategy of QbS to solve the task, but a mix with a keyword-based strategy which is likely to perform better than any strategy alone as it combines the strength of both strategies to compensate for the weaknesses when one of the strategies is used in isolation: The textual information describes the concept that shall be represented through an image part, the sketch mainly defines the placement inside the composed image. Visual aspects are considered for picking ideal candidates from the set of images that are defined by the textual description of the concept.

10.3.2 Query by Sketch for Known Image Search in the MIRFLICKR-25000 Dataset

As this thesis is dedicated towards similarity search and content-based image retrieval, from the tasks that can be supported by QbS, we decided that Known Image Search is the task we want to address with our approach. The focus on a particular task allows us to define the criteria based on which we can measure success since the task does not only define what the input is –in case of QbS this will be at least a sketch–, but also the desired aim: return the sought image at a good rank. For the evaluation, we will be using the MIRFLICKR-25000[21] dataset [Huiskes and Lew, 2008, Huiskes et al., 2010] that has been collected from the popular online image sharing platform *flickr*[22]. The collection consists of photographs that have been selected based on their license (only Creative Commons-licensed pictures)[23], and on the flickr measure of "interestingness" which takes into account how many people watched the image, commented on it, tagged it, picked it as favorite on flickr at the time when the collection was created. The images in the benchmark collection are accompanied with metadata on the photographer, the title and tags of the image, Exif metadata [JETIA, 2010], and license information. It is therefore a fairly realistic set of images that people would collect themselves and would

[21] `http://press.liacs.nl/mirflickr/`

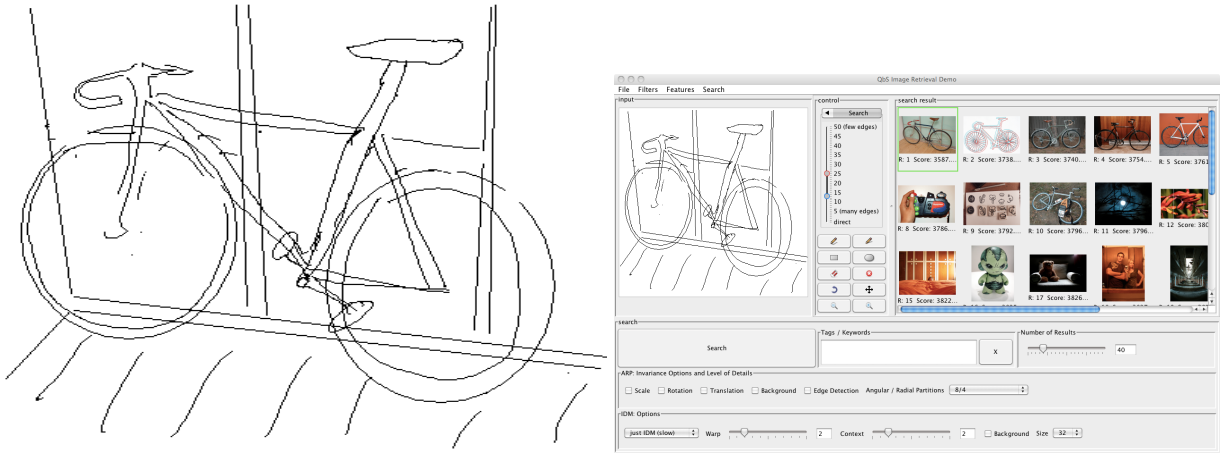
[22] `http://www.flickr.com/`

[23] Creative Commons-licenses is a family of licenses that do allow the reuse of the works, cf. Chapter 9.2 on page 197 and `http://creativecommons.org/`.

(a) im1660.jpg (b) im10853.jpg (c) im18707.jpg (d) im18797.jpg

Figure 10.15: Some images from the MIRFLICKR-25000 dataset

(a) Known Image im18707.jpg (b) Segmented



(c) Sketch (d) QbS Result

Figure 10.16: Known Image, Segmented View, and Sketch: the original image (a), a segmented and annotated version (b), a sketch of the image (c), and the search results (d).

like to search for. Figure 10.15 shows four example images that have been selected for the purpose of evaluation.

It's important to keep in mind the difference between Known Image Search (Chapter 2.2.1) and Object Detection (part of Classification in Chapter 2.2.2 on page 34) as

those two get frequently blurred when textual search is performed as a strategy: The task we will try to solve is to find exactly one previously known image, for instance the image in Figure 10.16(a). A textual description of this task may be: "Find the image that shows a blue bicycle standing in the corner of a room with brown wooden floor and white wall to the left and a wall (or door) made of frosted glass behind the bicycle." This image, like all images in the dataset, was shared on the image sharing platform Flickr.[24] It was accompanied by the following description:

> **Version 1.0 release candidate 1**
> Here's the "release candidate" version of the bike. Nitto moustache handlebars with red cloth tape; this time the color is a better match for the accents on the frame, so I'll stick with it for a while. I've also swapped the cog for a 16-tooth 3/32" one (previously a 15-tooth 1/8" one). The chainline is just a hair better and the chain is quieter with it, and 42/16 works better for urban riding than 42/15, at least at my level.
> *Tags: bike, bicycle, vintage, singlespeed, fixedgear, fixedwheel*

This textual information exclusively focuses on the main object depicted in the image, but not the entire scene shown in the image – which makes it a less helpful for searches; in particular if the user is interested in the image and does not remember the describing text. The most helpful information might probably be, that the image contains a bicycle – just like about 170 other images among the 25'000 images do.[25]

Of course, if the a computer vision system would be able to analyze every image in the collection with a high degree of image understanding or if all images had been annotated manually as in Figure 10.16(b), the search task could be solved easily – as long as there are not too many very similar images in the dataset. However, such detailed annotations will frequently be not available and in a generic setup, computer vision algorithms might also not be able to provide deep enough image understanding.[26] And furthermore, if there are not too many very similar images in the dataset, a search based only on the appearance of the image without any detailed image understanding might be sufficient.[27] Figure 10.16(d) shows a sketch of the image and Figure 10.16(d) the

[24] This image was shared by Petteri Sulonen at http://www.flickr.com/photos/primejunta/623867918/.

[25] In the dataset, 111 images are tagged with "bike"; 87 are tagged with "bicycle". Out of these, 37 have both tags, the boolean query "bike OR bicycle" returns 161 images. Additionally, 3 images are tagged with the wrong spelling "bycicle". Tags are not limited to one language, so one should also add "bicicleta" (spanish, portuguese, romanian; 9 hits), "Fahrrad" (german; 6 hits), "velo" (french, swiss german; 3 hits), and so on as well semantically related terms, e.g., that describe just particular brands or types of bicycles.

[26] This shall not be confused with the fact, that for *non-generic problems*, there have been quite a number of very successful approaches to object detection and automatic image annotations. But general solutions are much harder – see for instance [Pavlidis, 2009a].

[27] One particular kind of "objects" that are present a lot in the MIRFLICKR-25000 collection, shouldn't be a surprise: People. There have been many approaches to detect and identify people and in particular faces in images; some of the have already been mentioned in Chapter 5.1.2. However, in the particular case of this dataset, there will be a near-zero probability that any of the users evaluating the collection will know any of the people in the collection as there are no pictures of famous people, from the circle of friends of the users, and alike. In a personal image collection this will certainly be different and in such collections, the ability to trace easily pictures of particular people is certainly of great importance. For

search results. Notice, that this search was performed purely based on the visual appearance – the search was not restricted to images that have been tagged or otherwise annotated. Such a restriction could be added if needed. In this particular search, it was not: the sought image has been found as the first result already – and Query by Sketching can therefore also be used in cases where metadata is unavailable or not sufficient for text-based retrieval. For the MIRFLICKR-25000 collection, 2'128 out of the 25'000 images (or 8.51%) do not have any tags – even though the collection contains only images with highest "interestingness" shared on a very popular website. Less popular or private image collections are expected to have even less tags available. Therefore, there are many situations in which keyword search alone cannot deliver satisfactory results.

10.3.3 Challenges in QbS for Known Item Search

The main challenges in Query by Sketching for Known Item Search relate to the aspects of *Matching Tolerance* that have been introduced in Chapter 2.3. These can be summarized as:

- *Query Input:* The user has limitations in both, memory of the known item and the ability to draw it. For the latter, the available input device has great impact. Novel input devices like Tablet PCs or interactive paper as presented in Figure 2.3 have to be considered a necessity rather than just a gimmick to reduce frustration of the end users.

- *Color vs. Black-&-White:* As pointed out already on page 51, although users tend to focus a lot on color information, it is quite hard to match colors closely – in particular when a user has to choose the color from memory. Therefore the tolerance given to the color must be fairly high, which makes it hard to use it as the single information to use.

- *Single Edges vs. Complete Contours:* In many cases it might be simpler to draw only some of the most prominent edges rather than to sketch the correct contour outline, in particular for real world 3D objects that the user has to mentally project onto 2D. In [Vuurpijl et al., 2002], 5 out of 26 users reported explicitly in their feedback, that they were very limited by the requirements that they had to draw closed outlines and the system did not allow intermediate pen-lifts. For *Known Image Search*, the user may remember the pose of the object and its placement inside the image; drawing it, however, still remains remains a non-trivial task even for experienced users.

- *Degree of Detail:* As the aim of the task is, to find again the known image, the user shouldn't be required to draw many details to be able to perform a search. Thus, the system must be able to tolerate many deviations between the sketch and the

generic collections like the MIRFLICKR-25000 collection, the best effort that face detection could achieve is, to mark areas in images in which faces appear. Such information can easily be integrated in the search by a filter predicate. Currently, our prototype does not include any dedicated functionality to detect and search for faces.

sought image due to the omission of details in the sketch. Such omission might not only be caused by "laziness" of the user, but also due to the inability to remember these details. It is therefore crucial to be able to distinguish between empty and unknown areas in the sketch as detailed on page 53.

- *Dealing with Issues of Scale, Rotation, Translation, Illumination:* In particular when there are areas left blank in a sketch –either because it should remain empty or because its content in the image is unknown– it gets hard to place objects at the correct position and at the right scale. For the scale, this does not only include the finding the correct relative size w.r.t. to other objects, but also drawing with the correct aspect ratio. In known images, the rotation of sketch and sought image usually does not mismatch arbitrarily: Like translation, the user usually has a rough idea where to place the individual parts of the image and at which angle. However, minor changes in translation and rotation may occur for reasons of inaccurate drawing as well as lack of exact memory. Additionally, as detailed on page 58, rotations by multiples of 90 degrees are of interest if orientation information is missing in image files. The illumination can cause issues when colors change their appearance or when edge detection has to deal with very dark areas or the borders of shadows.

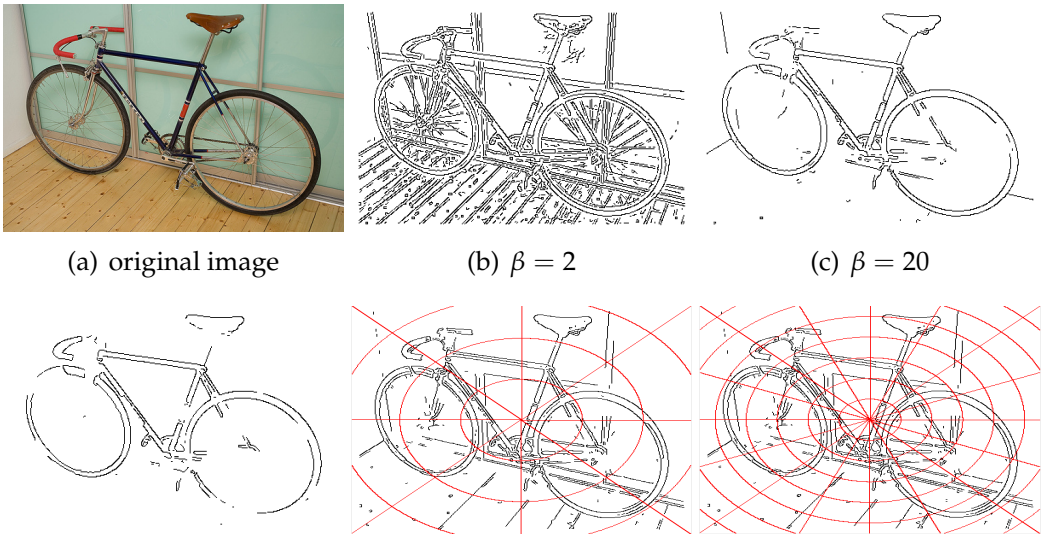
10.3.4 Retrieval Process

To address the challenges for QbS in our prototype implementation, we use mostly edge information that has been acquired from either a Tablet PC or interactive paper. The user starts to draw some edges, but not necessarily closed contour outlines. The QbS system provides a fast query mechanism to retrieve some of the top-ranked results to give the user a fast feedback about the retrieval result that can be expected from the final or full result. Then, the user is able to add further details to refine the search and remove misleading parts of the sketch without having to redo the complete sketch.

The study in [McDonald and Tait, 2003] has shown that query by sketch is able to assist the users in a best possible way if it is used in cases where the user has seen the item before search. Therefore, she will not only be able to provide the system with the sketch as input, but also some additional information on the invariances that have to be considered by the system and on how to distinguish between empty and unknown areas.

In contrast to directly comparing images with each other –as it is the case in CBIR, e.g., by comparing pixel intensities,– sketches and images always have to receive some preprocessing in order to make them comparable. This also limits to a certain extent the features and distance functions described in Chapter 5.2 and 5.3 that a) can be used with sketches and b) provide the needed invariances and support for the distinction between empty and unknown areas to address the challenges in for sketch-based Known Item Search.

QbS focuses on edge information and does not rely on semi-automatic segmentation or annotations of the image database, since the latter usually require significant work at the time of inserting images to the collections which users are frequently not willing to invest: Images may be tagged with keywords, which are available for retrieval, but this



(a) original image (b) $\beta = 2$ (c) $\beta = 20$

(d) $\beta = 30$ (e) ARP 8/4 (f) ARP 16/8

Figure 10.17: Edge Detection for ARP with several β values: An example image (a), edge maps for various values of β in (b)-(d), partitioning with $\beta = 10$ exploiting 8 angular and 4 radial partitions (e) and 16 angular and 8 radial partitions (f).

is intentionally left optional. Edge information corresponds best to perceptual features that capture shapes as described on 118, however there are still some limitations.

QBIC's shape features [Niblack et al., 1993, Flickner et al., 1995] or MPEG-7 CSS [Sikora, 2001, Bober, 2001] require segmentation of objects. The potential lack of tags implies that any feature that requires to be trained on several instances from the same classes like Edglets [Wu and Nevatia, 2007] or the Boundary Fragment Model (BFM) [Opelt et al., 2008] cannot be used. Furthermore, user-drawn edges will usually differ in their details from edge information in real images so much that interest point detectors like the ones used in [Matas et al., 2004, Bosch et al., 2007, Agarwal et al., 2004, Deselaers et al., 2005, Teynor et al., 2006] will become very unreliable.

In its basic setup, QbS is based on the use of edge maps as defined in [Chalechale et al., 2004] for the use with Angular Radial Partitioning (ARP). Sketches as well as images in the collection available for search are examined at a resolution of 400×400 pixels, which is a much higher resolution than QVE [Hirata and Kato, 1992] or IDM [Keysers et al., 2007] would commonly use. From color images, only the gray intensities are used – or more precisely, the *Lightness* of the image captured in the L^* channel in *CIE 1976 (L^*, a^*, b^*) color space*.[28] The generation of edge maps is controlled by the threshold value β as illustrated in Figure 10.17, where low values preserve many edges while high values retain only very few and prominent edges. To better compensate for effects due to level of detail and edge detection, we do not define a single value

[28]Frequently, conversion of color images is performed in RGB color space with the gray value v being either $v = 0.299 \times R + 0.587 \times G + 0.114 \times B$ or just $v = \frac{R+G+B}{3}$. However, by setting v to the value of the L^* channel in CIELAB which tries to scale values better to be in line with human perception than RGB, we achieved better control over the amount of edges preserved by setting the threshold β which ultimately led to better search results.

of β for the entire collection, but rather extract the edge maps from the images at several values between 2 and 50.

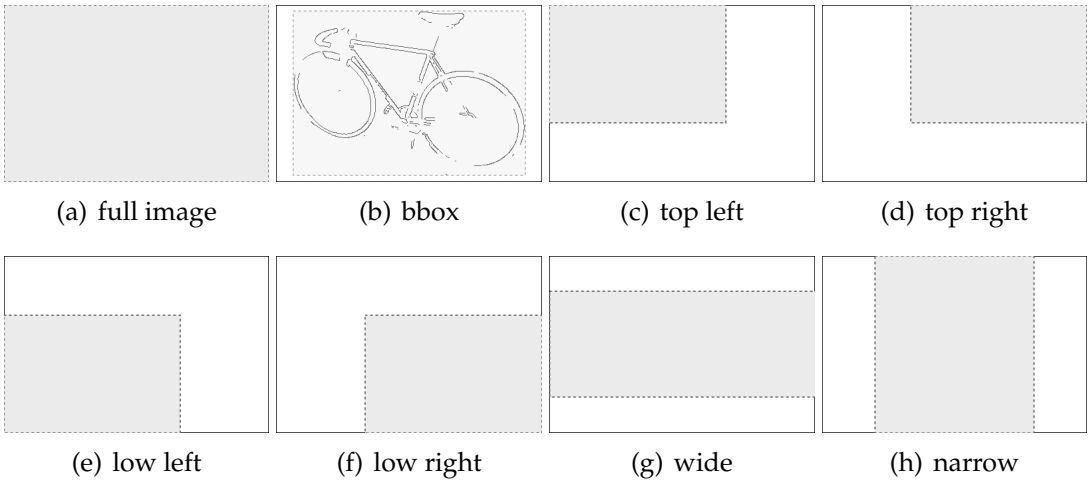
The basic setup incorporate two different sets of features and corresponding distance measures. First, Angular Radial Partitioning (ARP) is used as a compact, fast way to retrieve images when rough sketches and spatial layout is sufficient to separate the desired known item from all other images in the collection and robustness against many invariances are needed due to the deviations of the sketch w.r.t. to the known item. Second, we reused our implementation of the Image Distortion Model (IDM) used in Section 10.2 for medical images and applied it on the same edge maps. IDM is used as a more complex, computationally more expensive solution whenever the user needs a more thorough comparison between the sketch and the other images. For IDM, the user has to provide a sketch that is detailed and located close enough to expect meaningful results. In addition, in both cases the user can restrict the comparison to images selected on their metadata, e.g., only consider images that have been tagged with certain keywords.

Similarity using ARP

We support ARP at various resolutions. By default, ARP is used with 8 angular and 4 radial partitions (as depicted in Figure 5.9(h)). In this case, the number of edge pixels inside the sketch and the edge map are counted, which will generate a 32-dimensional feature vector. In order to distinguish empty from unknown areas as defined in Chapter 2.3.3, the vectors are compared with a weighted Manhattan distance as defined in Equation (5.16). In the easiest case of treating all non-edge areas as empty areas, all weights will be set to 1 which is equal to using an unweighted Manhattan distance. In another simple case where partitions without any single edge pixel are treated as unknown areas, such partitions of the sketch will cause a weight of 0 and otherwise , therefore ignoring any deviation between the sketch and compared images for this partition while still summing up the absolute differences in counted edge pixels in all non-empty partitions. The first case is the only case considered in [Chalechale et al., 2004], the latter case is equivalent to the approach proposed in [Eitz et al., 2009a] for a completely different descriptor.

The user can choose between the two cases, and therefore has the freedom to either sketch the full image or to focus on drawing only the parts of the image she remembers well, treating all other areas as unknown. The approach can be easily extended to let the user also define explicit relevance of areas as this information can be incorporated into weights on partitions. Such regions of interest [Springmann and Schuldt, 2008] can be easily selected using input devices used for sketching.

In particular, when only parts of the image are considered relevant, it becomes important to also ask the user whether she assumes the sketch to be placed in the appropriate position or request from the system to be less sensitive to translation. In cases where the entire image is used to define empty areas around the searched sketch, translation invariance might be important, for instance when searching for trade marks. As long as not too many edge pixels move to a different angular radial partition, ARP will not be very sensitive. For translations that are further off the position of the sketch, a heuristic



(a) full image (b) bbox (c) top left (d) top right

(e) low left (f) low right (g) wide (h) narrow

Figure 10.18: Image regions for translation (a)-(f) and scale invariance (g),(h)

can be applied: the original image is cut into several regions as mentioned in Figure 5.6 that still cover enough pixels to give meaningful search results. The ARP features are extracted for each of these regions (Figure 10.18(c)-(f)), as well as for the image as a whole (Figure 10.18(a)) and a bounding box on all non-empty pixels (Figure 10.18(b)). During search, the sketch will be compared to all these regions, thus compensating to a certain extent for translation.

As most of these regions are smaller than the full image, comparing the sketch or parts of the sketch can also provide some invariance to scale. Additional regions for different aspect ratios (Figure 10.18(g)-(h)) assist when the scaling does not maintain proportions.

Small degrees of rotation will also not shift too many edges from one partition to another, such that ARP will not be very sensitive to such deviations either. However, if the user expects even more rotation, invariance to rotation is provided by applying the one-dimensional fast Fourrier transform (1D FFT) on the features as proposed in [Chalechale et al., 2005].

Since the ARP feature vector is rather compact, it is feasible to extract and store several variations of parameters in order to enable invariances. Therefore, multiple sub-images for translation invariance and aspect ratios are processed and the 1D FFT for rotation invariance is applied and stored separately for each chosen β value. For simple, regular searches the query feature vector from the user's sketch will be compared to exactly one feature vector for each image in the collection. Invariant searches will compare the distances with all corresponding representations, but only select the best-matching version of the image features for an individual image to compute its distance. If each version of the image is treated as an image (or object) of its own, this is equivalent to applying the Fuzzy-Or $\Delta^{or}_{MO}(Q, r)$ from Equation (5.22).

Similarity using IDM

For IDM, the edge map image is scaled down to smaller sizes. As default size, we use 32 pixels for the longer side. Since the user is expected to draw the edges in the sketch,

edge detection is never applied to the sketch. This means that the scaling has to be performed after the edge detection. Moreover, it is preferable to use an interpolation for scaling down and to apply thresholding afterwards to return to a binary image (edge / non-edge) rather than not using interpolation in scaling, which may drop a lot of the edges.

The user can parameterize the search with IDM to fit her needs very precisely. With the warp range, the user can specify how far edge pixels are allowed to be misplaced by translation, scaling, or rotation. This misplacement is measured in terms of the number of pixels in the reduced resolution of the image, e.g., at most 32 pixels on the longer side. Big warp ranges result in a higher invariance. A warp range of 3, for instance, would allow an edge to be misplaced by 3 of the at most 32 pixels in every direction which is roughly +/-10%. The size of the local context defines whether individual pixels (local context of 0) or patches (local context > 0) are matched. Big local contexts result in small invariances as bigger patches must fit to achieve low distances. A local context of 2, for instance, would describe a patch of size 5×5 pixels.

To deal with invariances w.r.t. the value β used in extraction, the same strategy as for ARP is used. And also similar to ARP, unknown areas can be handled by ignoring non-edge pixels in the scaled down sketch. They only affect the distance score as part of the local context of an edge pixel — and more complex relevance judgements of areas in the sketch can be easily integrated as weights.

Integrating Filters

If the images are attributed with metadata (e.g., tags or keywords) and the user remembers them, search can be reduced to images that contain these metadata. In line with the extension described in Chapter 5.4.1 on page 140, this can easily be supported as a filter predicate. Similar to Equation (10.7) for returning the class label of medical images, let $\mathfrak{T}$ be the set of all tags and Θ the function that returns all tags associated with any image inside the collection:

$$\Theta : \mathcal{I} \rightarrow \mathcal{P}(\mathfrak{T}) \tag{10.8}$$

The predicate to test if a particular image has been associated with a tag t can then be implemented as:

$$P_{hasTag}(img) = \begin{cases} \textbf{true} & \text{if } t \in \Theta(img) \\ \textbf{false} & \text{otherwise} \end{cases} \tag{10.9}$$

However, it is important to notice that in many cases the images in the collection are not attributed with (good) keywords. In particular, private image collections like the pictures taken with digital cameras which are not shared online are frequently lacking such metadata. Even for resources that are shared online, there is often no shared or controlled vocabulary such that the user searching for one particular image may simply not know the 'right' keywords to find an item in such a collection. [Bischoff et al., 2008] showed also that not all tags can be used for search. Therefore, it is important that the QbS approach does not require such information to achieve satisfactory results but is able to exploit it, if available.

(a) im1660.jpg (b) im10853.jpg (c) im18707.jpg (d) im18797.jpg

Figure 10.19: Images selected for evaluation

10.3.5 Evaluation of Sketch-based Known Image Search

To measure how satisfying search results are in the context of Known Image Search, the rank of the sought item is of key importance as described in Chapter 2.2.1: Good results achieve low ranks. They satisfy the user needs if the rank is below the number of result images that the user is willing to browse – which certainly also depends on the number of images in the entire collection. In the case of the MIRFLICKR-25000 collection, the collection contains 25'000 images.[29]

Selection of the Known Items for Evaluation

We have performed known item search on four images that have different characteristics with respect to difficulties in drawing a sketch of them.

im1660.jpg in Figure 10.19(a) shows basically a single object, which is located in the center of the image. Moreover, the background is rather homogeneous and different from the foreground, which results in the ability of edge detection to find some value of β where all of the edges in the background are removed while still many edges of the foreground are preserved. In this particular case, at a value of β=7, none of the clouds contribute to edges anymore while almost all details of the plane are still available. Such characteristics makes it comparably easy to retrieve this image from a sketch as the common intuition of users to draw just the desired object works very well and also placing an object directly in the image center is easier than estimating a displacement from the center. As the background disappears even for comparably low values of β, a bounding box region (as in Figure 10.18(b)) can also compensate for translation and scaling.

im10853.jpg in Figure 10.19(b) is considerably more challenging as it contains several distinct objects of interest (e.g., houses, mountains). There are plain or homogenous

[29] As described in Chapter 2.2.1, the absolute rank of a known item provides some intuitive understanding whether the known image search can be considered successful, as the rank has to be within the number of result images that the user is willing to browse. For measuring the quality of different systems for many queries and across various collections of different size, the rank needs to get normalized. On measure to achieve this is the Average Normalized Modified Retrieval Rank (ANMRR) as described in [Manjunath et al., 2001]. However, as we described in [Springmann et al., 2010a, pp. 16ff], for a collection of fixed size and searching for a single known image, the resulting ANMRR value between 0 and 1 is less intuitive than the ranks itself and does not provide additional information.

Figure 10.20: Tablet PC running the QbS software for Sketch Acquisition

areas like the sea or the sky as well as areas with strong contrasts at high frequency like the rocks and vegetation on the mountains. The spatial distribution is not very hard to estimate and draw, however there might be issues with placing the coastline too high or too low and scaling, in particular matching the aspect ratio and relative sizes of objects.

im18707.jpg in Figure 10.19(c) contains again one main object, but this time, it is rather difficult to separate it from the image background. There is no value of β at which the details of the bike would still be preserved while the floor and walls would already disappear. Notice also that segmentation in image processing as well as the classification of bicycles based on visual features in the area of pattern recognition is considered challenging due to the property that most of the object's area does not block the line of sight to the background. The image perspective is also non-trivial as there is no planar view on the bike as, e.g., a frontal or lateral view.

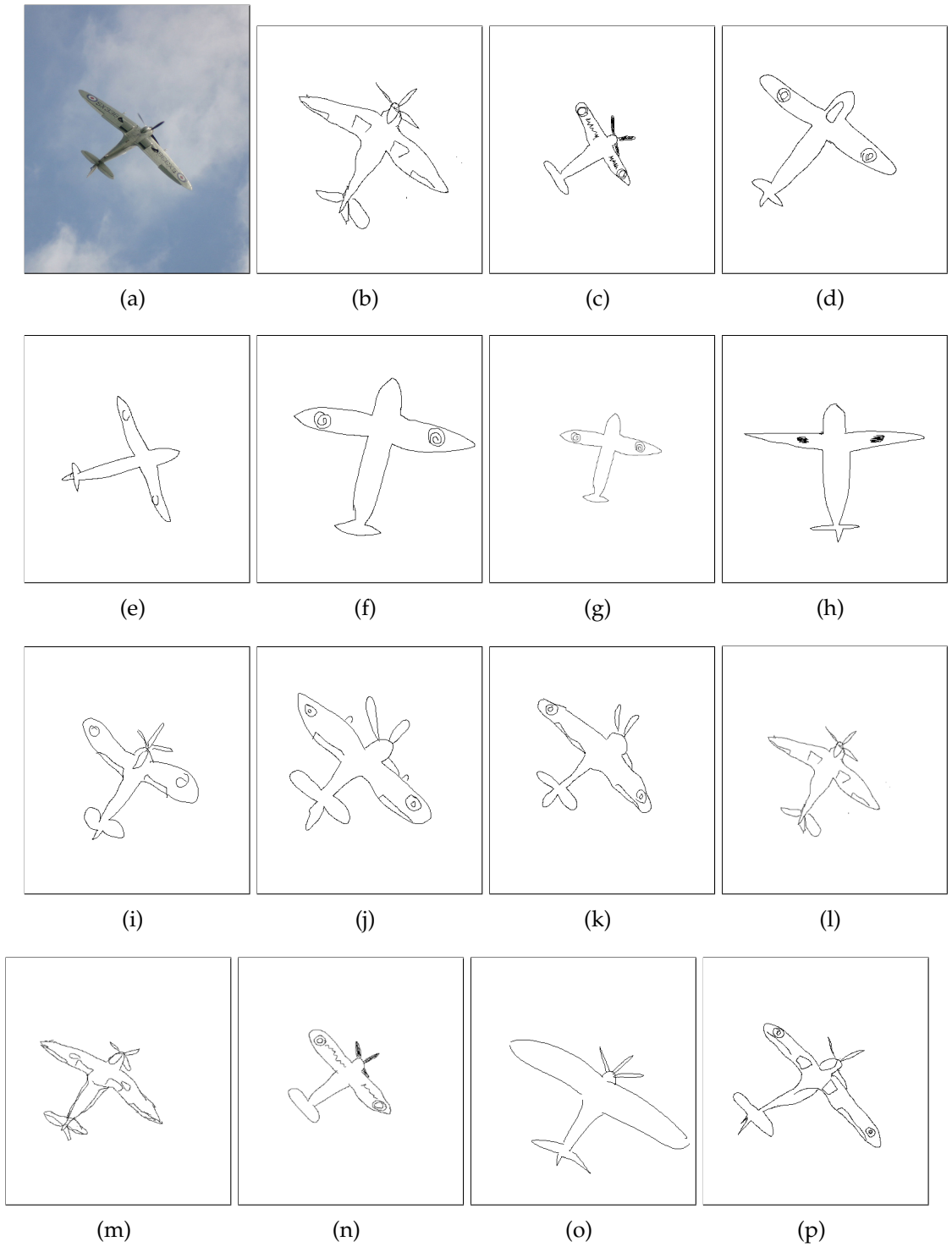
Finally, im18797.jpg in Figure 10.19(d) contains several prominent objects, although the hand and figure of the Eiffel tower are dominant. These objects are neither placed directly at any image border nor directly in the center of the image. Significant parts of the image are blurred, but colors still differ too much to generate areas which would appear to edge detection as homogenous areas. There is some (almost) empty area in the lowest part of the image which commonly makes it harder in the sketch to place the non-empty areas in correct proportion. Finally, none of the major visible lines are either horizontal or vertical; all are rotated a bit against those orientations.

Sketch Acquisition

For the purpose of evaluation, we have collected sketches for these four images from a group of seven people with different sketching abilities. The individuals were given time to familiarize with the QbS system running on a Lenovo ThinkPad X200t Tablet PC system as shown in Figure 10.20. Furthermore, they were then requested to search for the known items which were shown to them in printed form. We collected a total of 15 sketches for im1660.jpg (Figure 10.19(a)), 14 for im10853.jpg (Figure 10.15(b)), 15 sketches for im18707.jpg (Figure 10.19(c)), and ten sketches for im18797.jpg (Figure 10.19(d)). All sketches obtained are shown in Figures 10.21, 10.22, 10.23, and 10.24.[30]

Out of the four images, none was the only image either showing that kind of object in this collection or the only one using this particular kind of composition. As we use features that do not take color into account, the results are also not biased due to the selection of images using exceptional colors or color distributions.

[30] All sketches used in the evaluation are available for download on our website:
`http://dbis.cs.unibas.ch/downloads/qbs/QbS-User-Sketches.zip/view`



(a) (b) (c) (d)

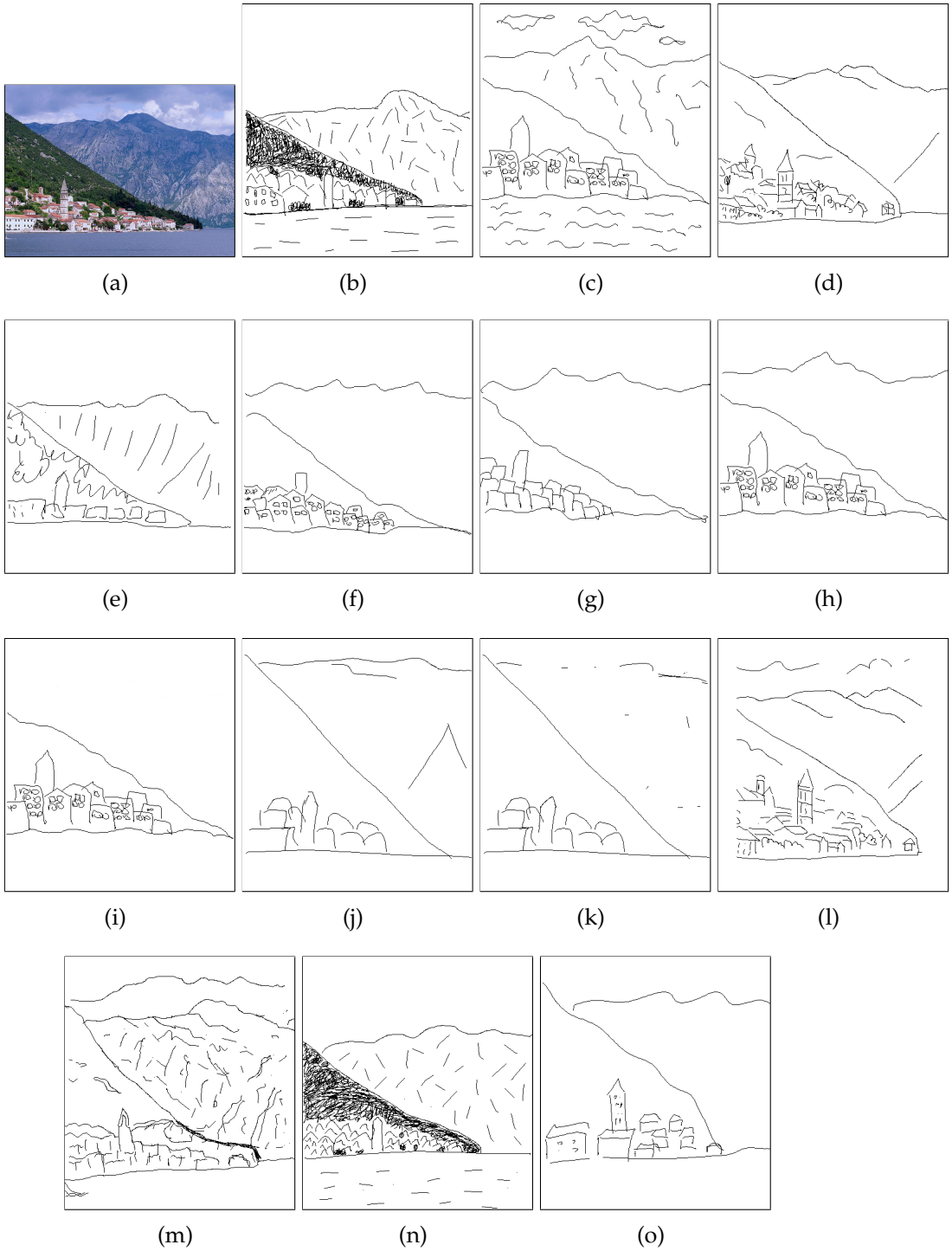
(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

im1660.jpg: 'Supermarine Seafire MKXVII' by Alex Layzell, License: CC BY NC ND

Figure 10.21: Sketches for image im1660.jpg used for evaluation.



(a) (b) (c) (d)

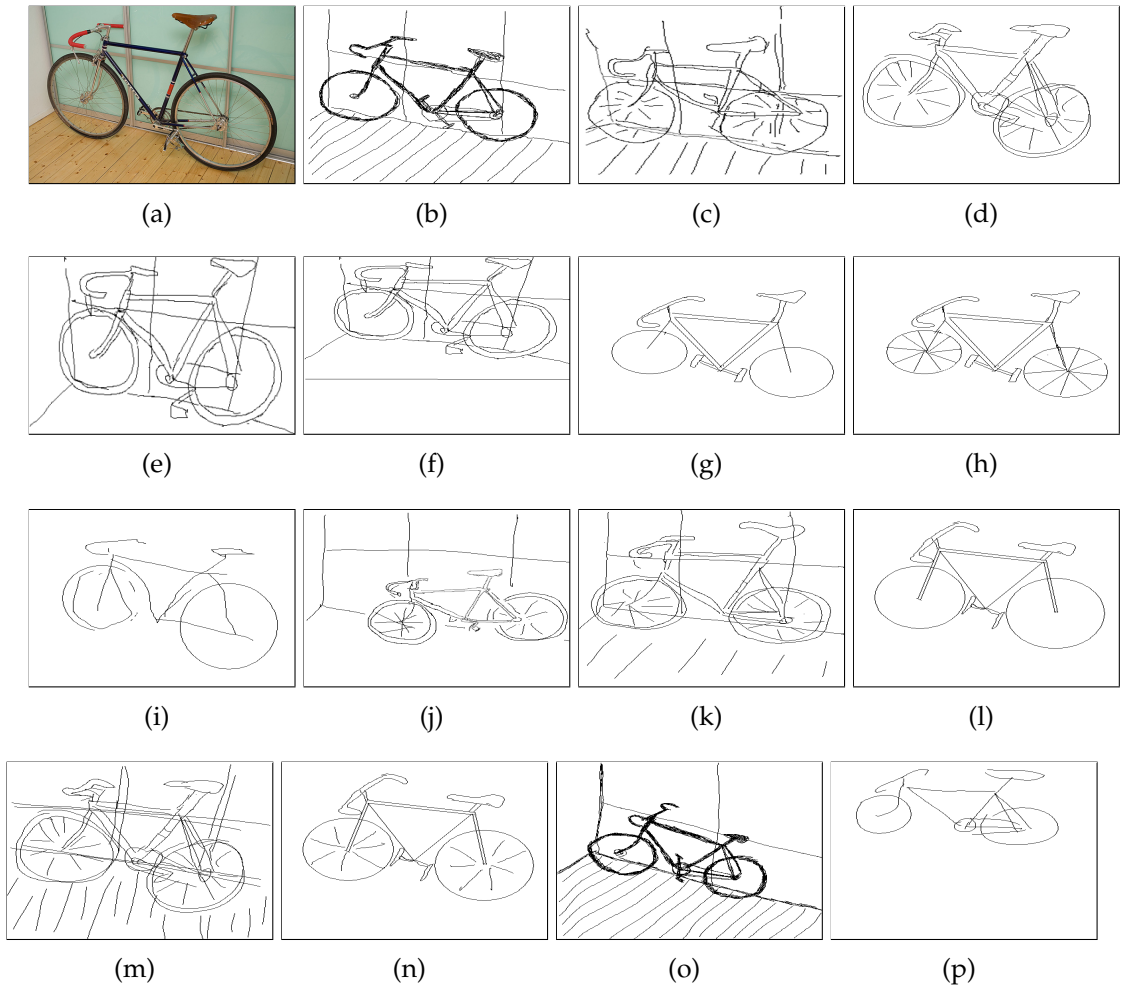
(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o)

im10853.jpg: 'Perast - Montenegro' by Milachich, License: CC BY NC

Figure 10.22: Sketches for image im10853.jpg used for evaluation.



(a) (b) (c) (d)

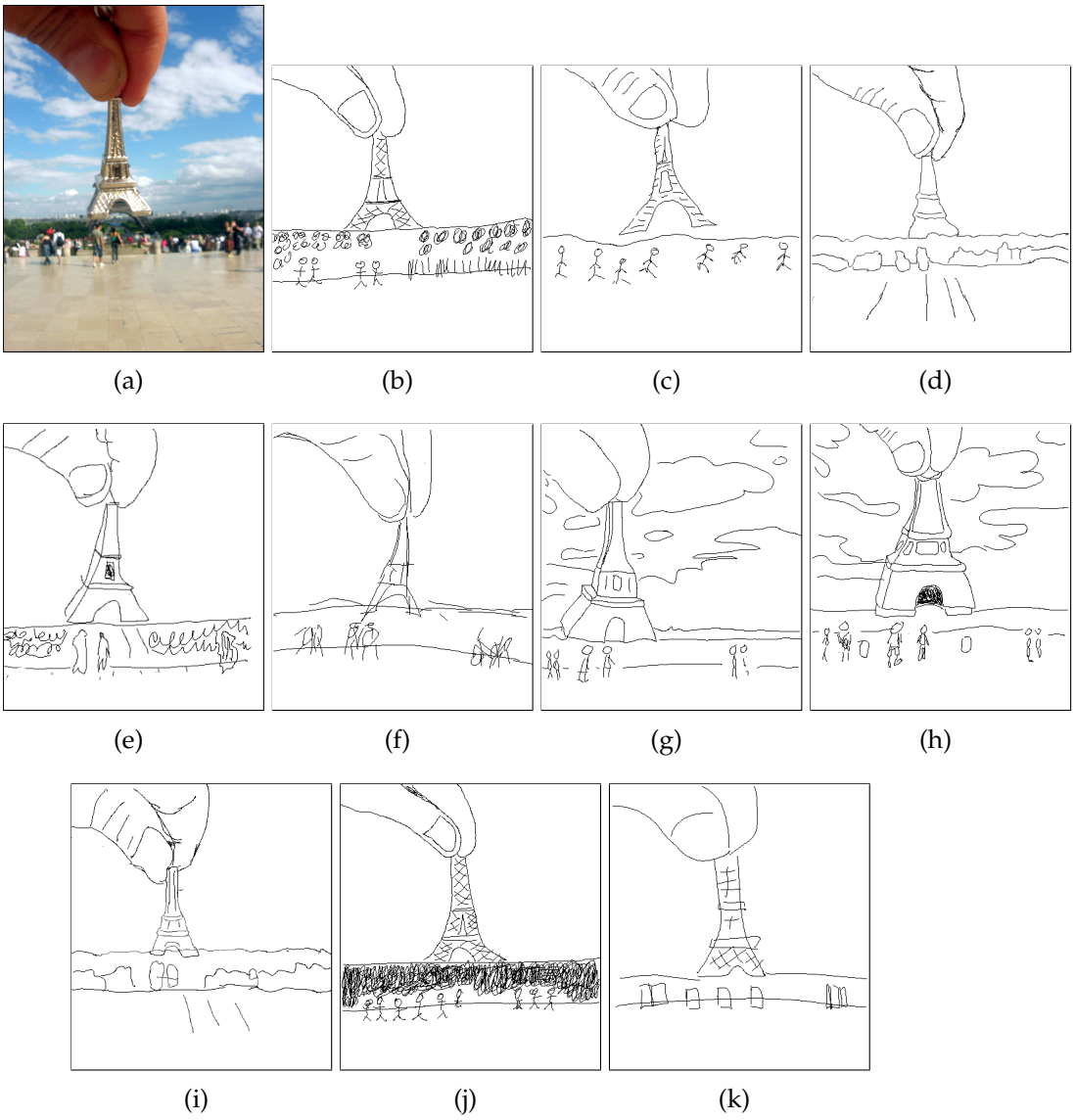
(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

im18707.jpg: 'Version 1.0 release candidate 1' by Petteri Sulonen, License: CC BY

Figure 10.23: Sketches for image im18707.jpg used for evaluation.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k)

im18797.jpg: 'If Eiffel In Love With You' by Gideon, License: CC BY

Figure 10.24: Sketches for image im18797.jpg used for evaluation.

Selection of Tags for Known Items

We measured the individual performance of ARP and IDM. In addition, we assessed the use of text search accompanying each algorithm. For the keyword search, we used the two most appropriate tags that flickr users associated with the four images. The selection of the keywords was performed with feedback that we received from the people contributing sketches when we showed them the list of tags associated with the images. Through this, tags that were highly specific like "singlespeed" for im18707.jpg (Figure 10.19(c)) and "trocadero" for im18797.jpg (Figure 10.19(d)) have not been selected, even though they would have been appropriate – as their relation to the image where unknown to most of the users. The complete selection of tags that were chosen in our evaluation are presented in Table 10.3. It is worth mentioning that when the tags where presented to the people participating in the study, they were presented in the original order of the MIRFLICKR data set. The tags displayed in Table 10.3 are ordered, having the least specific tags on top. This means that the tags that have been assigned to most images are listed first, the tags that would filter out most images are listed last. This order has been chosen as it nicely shows that the more specific tags are never selected by the users — they are too specific and therefore unknown to people who did not tag these images themselves.[31]

Table 10.3: Selected Tags used in Text Search Filtering

Image	FLICKR Tags (Selected Tags are highlighted in **bold**)
im1660.jpg	canon, 2008, usa, spring, uk, old, europe, digital, eos, england, museum, rebel, world, **flying**, war, us, aircraft, show, flight, **plane**, air, 2, britain, engine, military, 300d, display, arm, airshow, royal, gb, wwii, east, ww2, british, united, english, fighter, force, aerobatic, raf, ii, european, aeroplane, duxford, kingdom, aerobatics, imperial, fleet, iwm, supermarine, seafire, trainer, piston, warbird, airworthy, anglia
im10853.jpg	blue, water, beach, **sea**, sun, sand, life, fun, **mountain**, beauty, montenegro, milachich, balkan
im18707.jpg	vintage, **bike**, **bicycle**, fixedgear, singlespeed, fixedwheel
im18797.jpg	**paris**, ring, **eiffeltower**, key, trocadero

[31] This experience is important as in cases where users search for their own images with their own tags or keywords, e.g., for personal image collections, they may use more specific terms and therefore get better search results using text retrieval. However, personal image collections or other collections that are not shared between people are commonly annotated more sparsely than the (selected) images that are shared and that can be annotated by several people by exploiting the "wisdom of the crowd".

Retrieval Quality

In a first step, we ran experiments with a small number of sketches to identify the most useful settings. Initially, we tested a large number of combinations of parameters because multiple options for invariances could be enabled or disabled independently and because parameters like the β value for edge detection, the number of angular and radial partitions for ARP, the reduced size of image for IDM and the size of warp range and local context can have arbitrary values. For the latter, we initially had to pick reasonable ranges.

For β, we selected the values $\{2, 5, 8, 10, 12, 15, 20, 25\}$ for subsequent evaluations as values above 25 did not perform well on any of the sketches. The reason for this is that too many details from the real images would have been removed during edge detection. Smaller steps would be expected not to generate significantly different results as the results of neighboring β values were already very close. For ARP, we selected the combinations (8,4), (8,8), (16,8) for angular and radial partitions. For IDM, we decided to stick to a single resolution of at most 32 pixels on the longer side of the images, a maximum warp range of 5, and a maximum local context of 3. We do not allow the option to ignore empty areas for local contexts less than 2.

The objective of the second step was to assess the ability to retrieve a known item from a collection by the rank at which the sought image was found. In order to be a useful tool, the rank has to be significantly smaller than one would expect for browsing the collection in alphabetic or random order and also less than the number of items the user is willing to browse before giving up. The user would expect to find the known item on average after having seen half of the collection – so in this case for 25'000 objects in the collection, this would be after 12'500 if no text search or other filtering based on meta-data was used and this would certainly be more than any user would be willing to browse.

As we expect the QbS system to be used in an interactive way, we would usually expect the user to set the number of desired results (in other words, the number of items the user is willing to browse) when submitting a search and refining the search to her needs, e.g., enabling or disabling some invariances, and re-submitting the search rather than submitting a single search and then browsing until the item was found. However, for the purpose of evaluating the ranks, we always continued to compute results until the known item was found. To remain comparable with what the user would see after some refinements, we performed this search with a smaller number of combinations selected in the first step and selected the best-performing setting. Figure 10.25 shows the ranks split up by known item. The ends of the bar show the rank achieved by the best and the worst sketch for this image, the thick grey bar starts at the first quartile and ends at the third quartile and therefore indicates the range of ranks which half of all sketches achieved. Finally, the blue diamond indicates the mean or –in other words– the average rank of all sketches for this image. Notice that the rank axis is in logarithmic scale. Also take into account that for image im18797.jpg in both ARP and IDM, the first and third quartile are both positioned at rank 1, since QbS retrieved the top rank for all of the sketches with a couple of exceptions that gave relatively poor results, which in return increased the value of the mean.

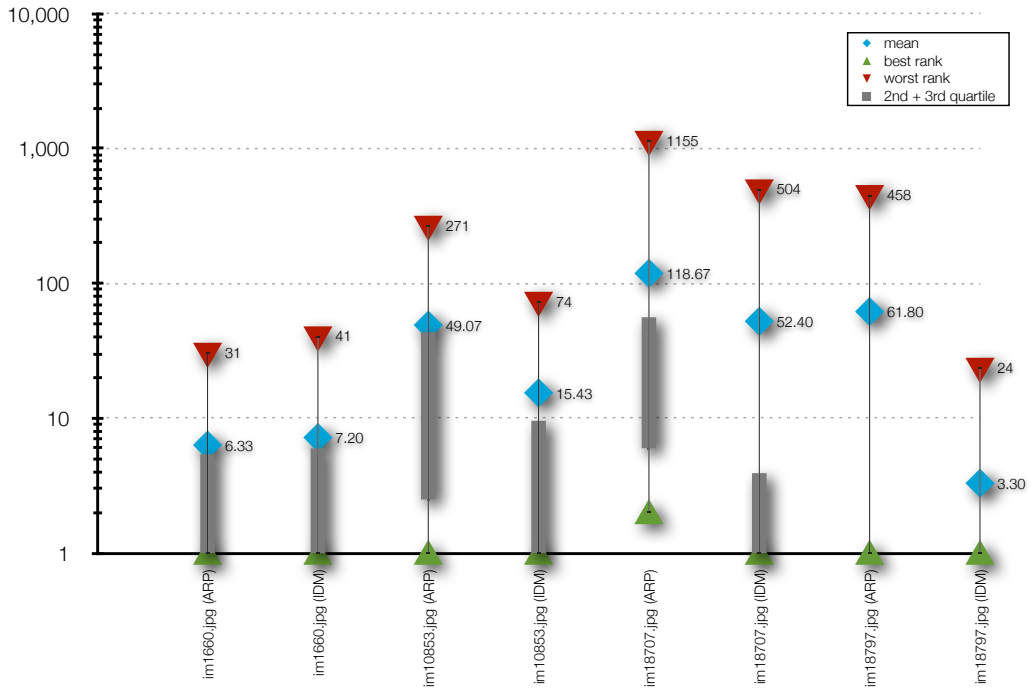


Figure 10.25: Ranks of Known Items (Text Filter off)

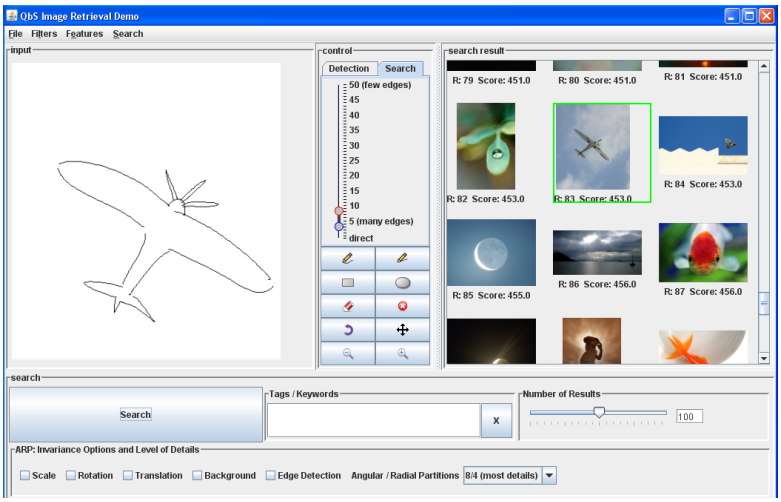
The results show that only a single sketch had a rank slightly worse than 1'000 while the vast majority of sketches achieved a rank below 100, which could already be a number of results a user might be willing to browse. For searching with IDM, even the majority is below 10, a number of results that can easily fit on a single result screen and out of which the user would recognize the image almost instantaneously.

With respect to the parameters, for all searches the choice of the appropriate β value depends on both the known item and the user sketch, with the latter being the one with higher impact. Therefore, it is not possible to come up with a single value of β for a diverse collection like the MIRFLICKR benchmark that would suit all users. However, the QbS system defines a set of values with reasonable limits and lets the user not only pick a single value, but also a range from this set that she can refine and shift during the search process. General observations show that the more detailed the sketch is, the lower the values of the threshold β are that give the best rankings since they preserve more edge information. For example, Figures 10.24(g) and 10.24(h) are sketches that pertain a high degree of detail, including background information in the form of clouds and therefore β values of 2 and 5 gave the best results for these two sketches. On the other hand, very rough sketches that contain little details tend to give best results with higher values of β. An example of this is evident in Figures 10.24(c), 10.24(d), and 10.24(k) that gave their best results with a higher β value of 15. It is also evident from

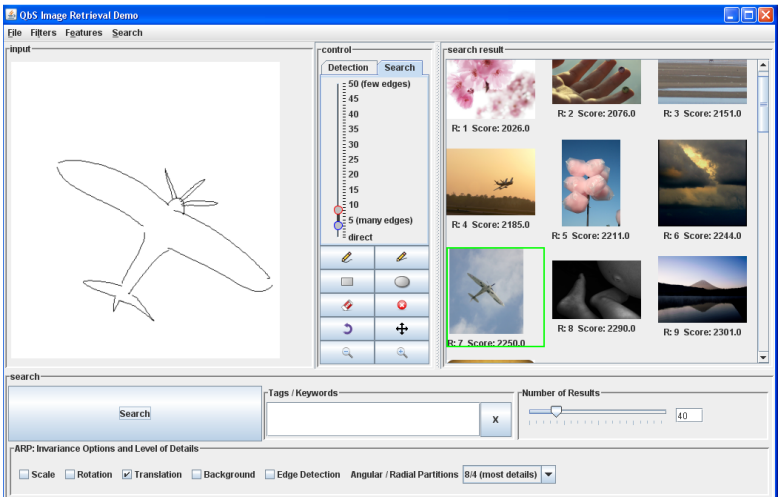
the classification of sketches according to best performing β in both ARP and IDM that IDM is less sensitive to the β values as it resulted in a smaller number of clusters than in ARP. This can be attributed to the fact that in IDM, local context tends to ignore much of the "clutter" of details in the image.

For ARP, rotation invariance was most helpful for finding im1660.jpg (cf. Figure 10.21(a); in 4 out of 15 sketches, this improved the ranking). Half of the sketches for im10853.jpg (cf. Figure 10.22(a); 7 out of 14) improved by ignoring background / completely empty areas. Retrieval results for a majority of sketches (8 out of 15) for im18707.jpg (cf. Figure 10.23(a)) did improve by enabling scale invariance. Scale invariance was helpful for 3 sketches of im1660.jpg (cf. Figures 10.21(a)) and im18797.jpg (cf. 10.24(a)). This shows that all invariances were needed in some cases — however, as one can expect, this heavily depends on the user's sketches. As an example, Figures 10.26(a), 10.26(b), and 10.26(c) show the effect of using an appropriate invariance and a keyword filter for a given sketch. Figure 10.26(a) shows the result obtained by QbS when using the given sketch without neither turning on any invariances nor providing any keyword filtering, which results in finding the known item only at position 83. When turning on translation invariance, the result significantly improves and a rank of 7 is achieved as shown in Figure 10.26(b). Furthermore, filtering the results with a proper keyword (in this case the keyword "flying"), assists in retrieving the known item at the first position as shown in Figure 10.26(c).

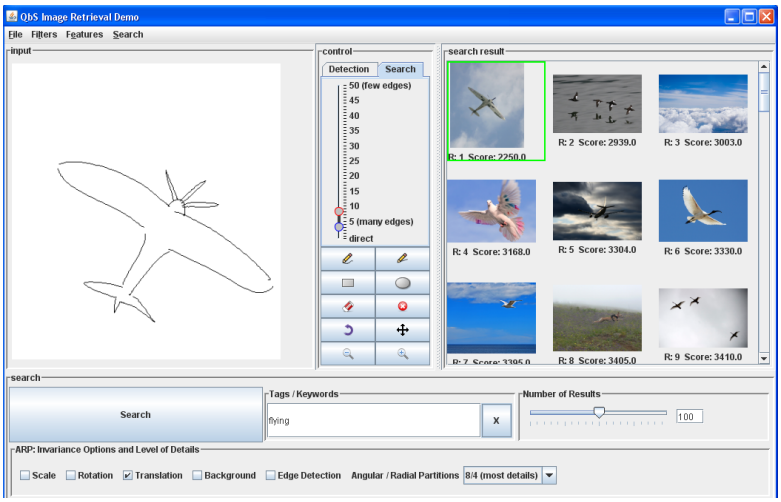
For IDM, it is even harder to identify trends as the results are very close. However, there is a tendency that for im1660.jpg (cf. Figures 10.15(a)) and im18707.jpg (10.15(c)) a warp range of 5 and a local context of 3 return the best results. This can be explained by the fact that this is the biggest setting for deformation that we allow while still constraining the results through the patch size and therefore being the option in which IDM can cope best with issues in translation and scaling.



(a) No invariances turned on.



(b) Translation invariance turned on.



(c) Translation invariance turned on & text filtering enabled.

Figure 10.26: Searches for im1660.jpg: No invariances turned on in (a) achieved rank 83, translation invariance turned on in (b) achieved rank 7, translation invariance turned on with text filtering in (c) with the keyword "flying" achieved rank 1.

		Text Filter (OFF)		Text Filter (ON)			Search Space	
img	Keywords	ARP	IDM	Lucene	ARP	IDM	Text	CBIR
im1660.jpg	flying	6.33	7.2	56	1	1	57	25,000
	plane			33	1	1	35	
	flying & plane			4	1	1	4	
	flying | plane			6	1	1	88	
im10853.jpg	sea	49.07	15.43	157	1.64	1.43	301	25,000
	mountain			45	1.07	1.07	88	
	sea & mountain			4	1	1	7	
	sea | mountain			4	1.86	1.57	382	
im18707.jpg	bike	118.67	52.4	30	1.13	1.53	111	25,000
	bicycle			21	1.13	1.2	87	
	bike & bicycle			8	1	1.07	37	
	bike | bicycle			8	1.2	1.73	161	
im18797.jpg	paris	61.8	3.3	24	1.2	1	224	25,000
	eiffeltower			1	1	1	13	
	paris & eiffeltower			1	1	1	11	
	paris | eiffeltower			1	1.2	1	226	

Table 10.4: Average Rank of Known Item

If search via keywords, tags, or other meta-data is performed, the number of images that have to be browsed in worst case shrinks already dramatically – as long as the keywords match the known item. In Table 10.4, we compare the average ranks achieved by pure content-based retrieval using ARP and IDM with the ranks when combined with keyword search and also with the rank achieved when only the keyword search is used.[32] The column 'Search Space' contains the number of objects that pass the filter, so in case of plain CBIR, all images of the collection and in case of a text filter, the number of hits. Notice that even for words with low selectivity like "sea" the combination with either ARP or IDM results in an average rank below two and in every combination, using the sketch plus a single keyword achieves a better rank than performing a boolean search where two keywords are combined with AND (with a single exception being Paris and using ARP).

This does not mean that keyword-based search does not perform well for this collection. We have selected only terms from a list of tags that people knew and therefore would use for search, even if they did not tag the images themselves. Drawing a sketch and adjusting the search parameters will certainly be a bigger effort than adding another keyword to the search. However, there are many situations in which keyword search

[32]For keyword searches, we use Lucene (available at `http://lucene.apache.org/`) to build a full-text index. The rank corresponds to the order in which Lucene returned the results.

(a) im972.jpg (b) im17919.jpg (c) im5820.jpg (d) im2267.jpg (e) im10829.jpg

Figure 10.27: Some color images from the MIRFLICKR-25000 dataset with different characteristics: im972.jpg in 10.27(a) contains only few different colors and few prominent edges; im17919.jpg in 10.27(b) few colors, but many edges; im5820.jpg in 10.27(c) has prominent colors, but less visible edges; im2267.jpg in 10.27(d) has also prominent colors, combined with strong and many edges; im10829.jpg in 10.27(e) shows a colored object in front of an almost white background.

alone cannot deliver satisfactory results – for instance for the 2'128 images (8.51%) of the collection that do not have any tags.

10.3.6 Extensions using Color and Texture Features

The QbS system in its initial configuration supported the features ARP and IDM, which both capture mainly the perceptual feature of shape. The system was extended in the context of [Giangreco, 2010, Giangreco et al., 2012] to also support color and in the context of [Kreuzer, 2010, Kreuzer et al., 2012] to support texture.

Color

Most images in the MIRFLICKR-25000 dataset contain colors. Some examples (that have also been used in the multi-user evaluation in [Giangreco, 2010]) are shown in Figure 10.27.

The colors a user chooses in a sketch will differ from real images due to several reasons:

1. The user will not be able to accurately select the color unless the devices used for this task are calibrated using standardized sets of colors or the user works on the same device the image is visible. In Known Image Search, it is very common that the user does only have a memory of the image or the image in a form that is not so easy to process, e.g., as a printout. Therefore any attempt to select a particular color usually results in some deviation.

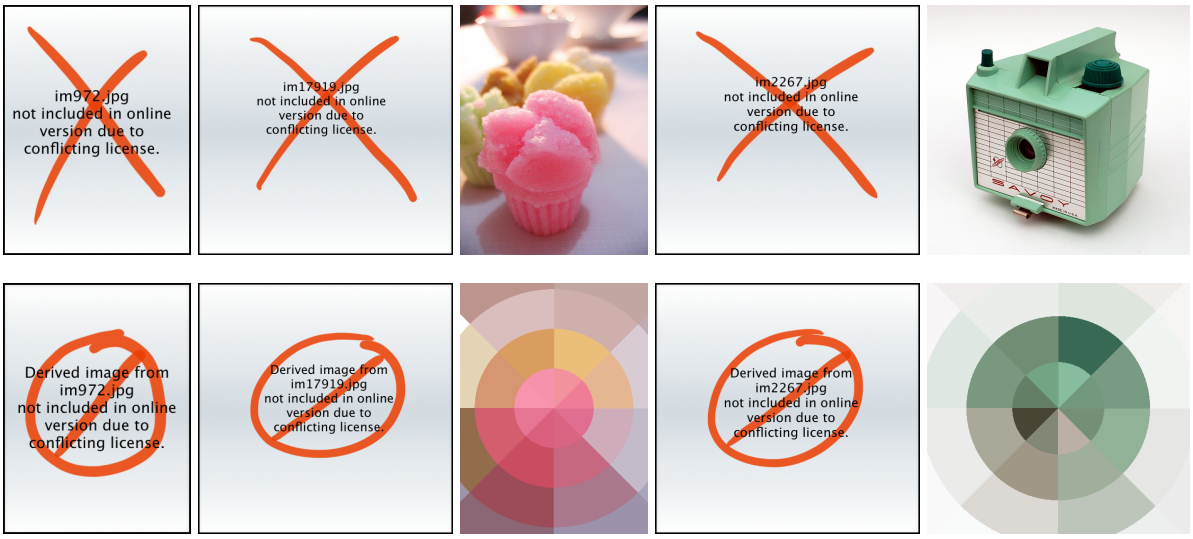
2. Real world objects usually do not have entirely homogenous color when photographed due to uneven light distribution, absorption, reflections, and shadows. Drawing an image that shows the same effects is a very hard task and time consuming; in particular for a sketch which is only used for searches. For such a task, users prefer to color larger areas only with few different colors.

As mentioned already in Chapter 2.3.2, a survey in [McDonald and Tait, 2003, p. 86] revealed that users frequently focus very much on color rather than the overall layout of the scene depicted in the image although this does reduce the result quality. Similar finding was also reported in the oral presentation of [Westman et al., 2008]. We therefore went for an approach that allows the user to provide colored sketches in which the spatial layout is of great importance in the derived features. From the positive experience with angular radial partitioning (ARP [Chalechale et al., 2004]), we re-used the same partitioning scheme with a modified descriptor to capture colors. Compared to descriptors like the Color Layout Descriptor (CLD) [Manjunath et al., 2001, pp. 710f], Spatial Color Distribution Descriptor (SpCD) [Chatzichristofis et al., 2010], or Multiresolution wavelet decompositions [Jacobs et al., 1995] that have previously been used for retrieval based on hand-drawn color sketches, this allows us to reuse our existing implementation and jointly optimize the search results for invariances to scale, translation, and rotation.

Color moments [Stricker and Orengo, 1995] have shown good results for content-based image retrieval, in particular compared to color histograms as color moments capture the statistical distribution of colors including the mean value of a color channel, the variance per channel and the covariance to other channels. Therefore they are not affected by operations that shift the entire color spectrum as this affects only the mean color directly. Furthermore, by computing the color moments in a color space that is close to the human perception and isolates different aspects like the *CIE 1976 (L*, a*, b*) color space* (a.k.a. *CIELAB*), *HSL*, or *HSV*, modifications of an image may change only few values. For instance, if the overall brightness of an image is changed, this is captured in CIELAB in the L^* channel, which stands for *Lightness*, without affecting the chromatic channels that encode the color with respect to the colors green and magenta (a^* channel), and blue and yellow (b^* channel).

For each of the partitions we extract 9 values: the three moments mean, variance and covariance for each of the three channels in CIELAB [Wigger, 2007]. We will refer to these as Angular Radial Color Moments (ARCM) in order to differentiate easily between them and the edge map-based ARP. We extracted ARCM with settings of 4 angular $\times$ 4 radial, 8 $\times$ 4, 8 $\times$ 8, and 16 $\times$ 8 and experiments showed that 8 $\times$ 4 performed best [Giangreco, 2010, pp. 26f].[33] ARCM can get compared with a L_1 Minkowski Norm (a.k.a. *Manhattan* or *City Block Distance*), either in a weighted or unweighted version. Experiments in [Giangreco, 2010, pp. 27f] showed that the configuration using the color moments directly (unweighted) performed best, followed by using only the mean values with weights on the chromatic channels a^* and b^* being one, the weight for the lightness channel L^* being set to 0.5. To support more control over the matching tolerance, we use a ϵ-insensitive version similar to Equation (5.9) in Chapter 5.3.1; only with the added property that different values of ϵ can be set for each moment, therefore allowing for instance to ignore small deviations on the L^* channel. Such a possibility to adjust the matching tolerance to the needs in color sketch retrieval was not present in [Manjunath et al., 2001, Chatzichristofis et al., 2010, Jacobs et al., 1995].

[33] This is the same partitioning scheme that also showed overall the best results for the search using only edge information in Section 10.3.5 and [Springmann et al., 2010a].



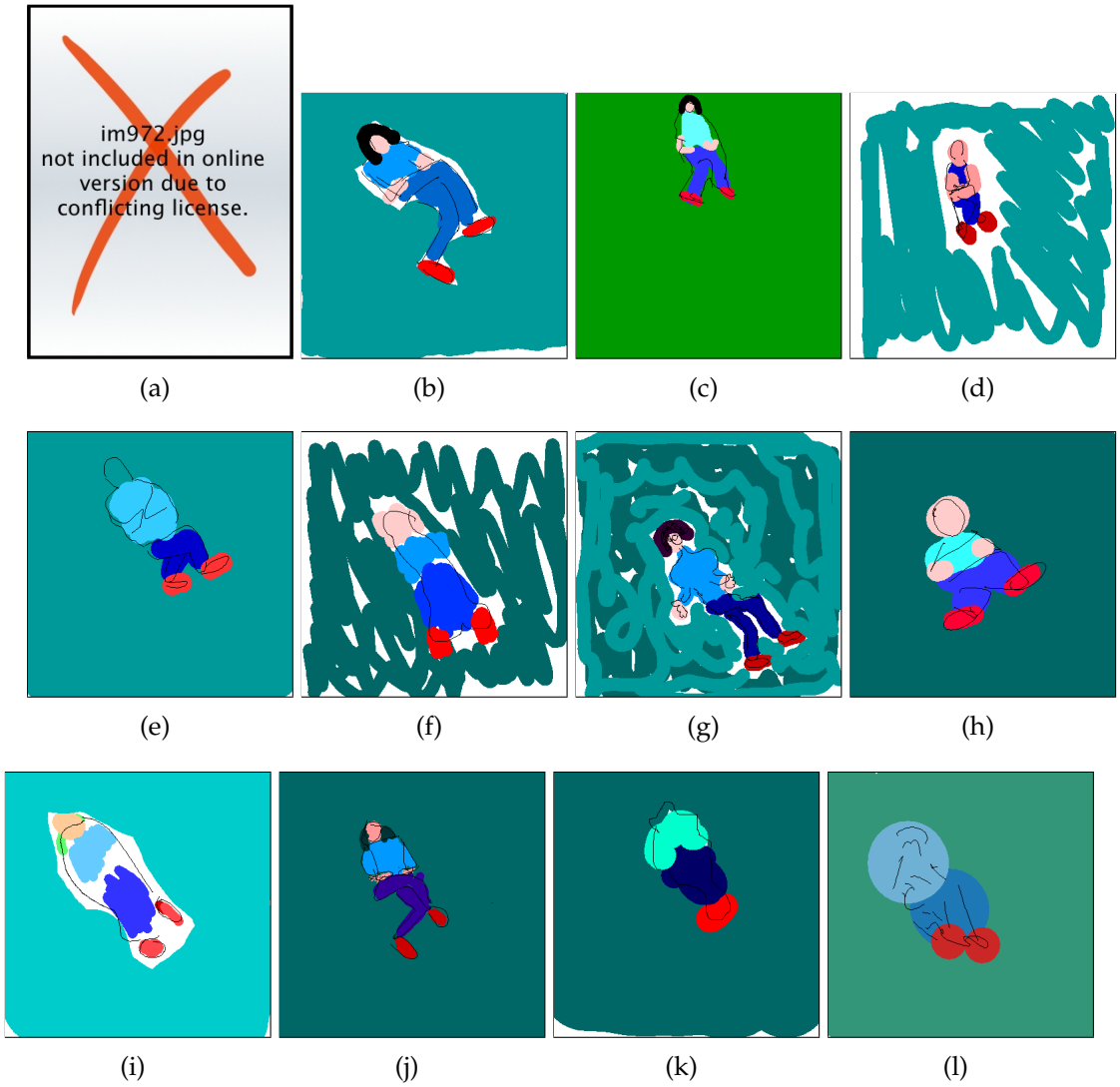
(f) im972.jpg (g) im17919.jpg (h) im5820.jpg (i) im2267.jpg (j) im10829.jpg

Figure 10.28: Original images in top row and below the mean colors with 8 angular, 4 radial partitions.

Figure 10.28 shows the mean color of each of the 8×4 partition; the other moments, variance and covariance are not shown. The figure shows the descriptor is much more about spatial orientation of colors than precise drawing in the sketch. This color descriptor therefore will only be able to separate the images in the collection if there are not too many images with similar placement of colors. However, the descriptor does not have to be used in isolation: Due to the same partitioning in ARCM, combination with edges captured with ARP is easier as also the same strategies to add support for invariances can be used. In order to compute an aggregated distance using both features, edges in ARP and color moments in ARCM, the individual distances have to be normalized as already mentioned in Chapter 5.3.4 as the features have different value ranges and distribution of values.[34] [Giangreco, 2010, pp. 13–16, 34f] investigated four different normalization approaches which all achieved very similar results: Min-Max, Gaussian, Equi-Distant-Bin, Equi-Frequent-Bin.

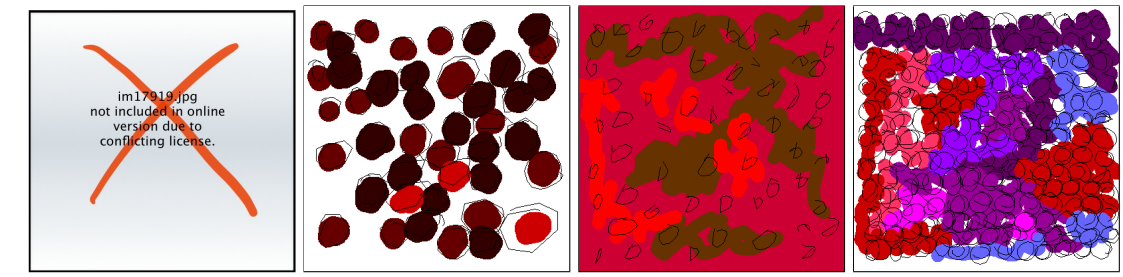
For evaluating the performance of ARCM in isolation against ARP and also the combination of ARCM with ARP, one sketch per image was collected from 11 participants, using again a Tablet PC as shown in Figure 10.20.

[34]Cf. also [Schmidt, 2006, pp. 238–254] and [Ruske, 2008].

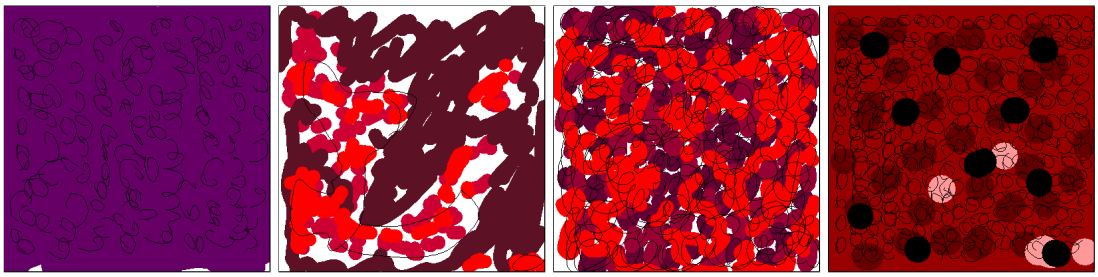


im972.jpg: 'The Head On The Floor' by ZORRO, License: CC BY NC SA

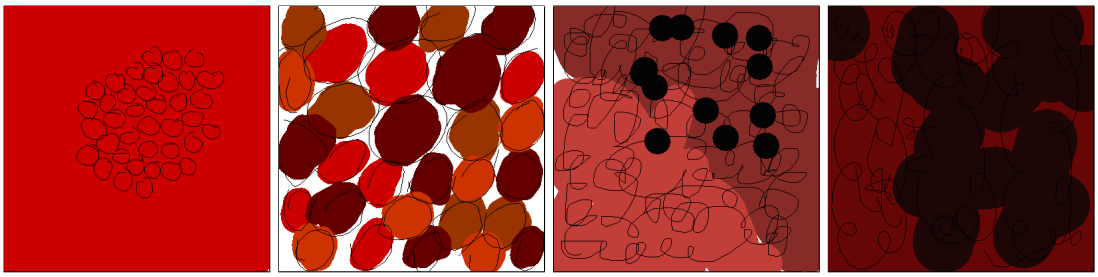
Figure 10.29: Sketches for image im972.jpg used for evaluation.



(a) (b) (c) (d)



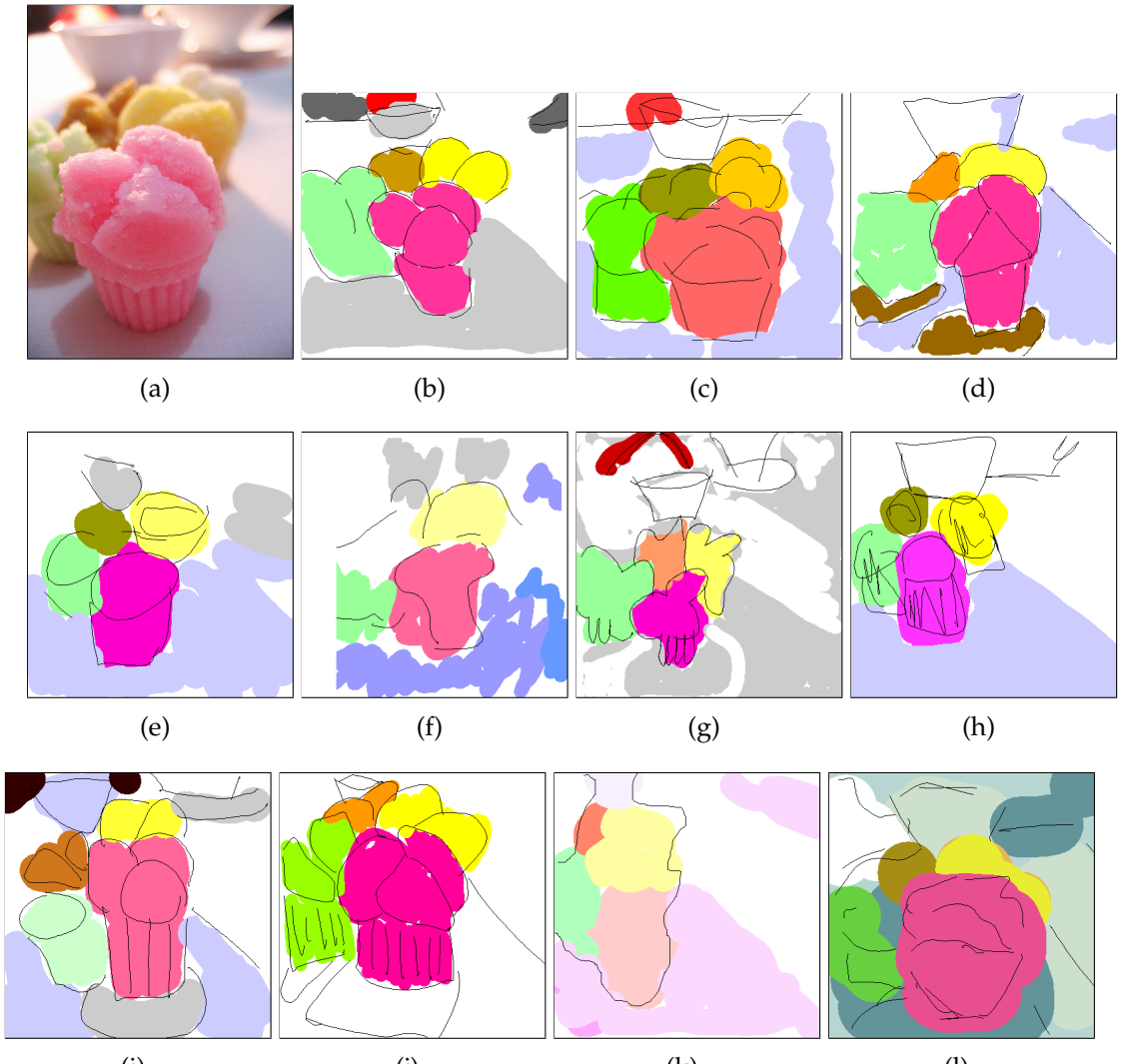
(e) (f) (g) (h)



(i) (j) (k) (l)

im17919.jpg: 'huckleberries' by julie (hello-julie), License: CC BY NC SA

Figure 10.30: Sketches for image im17919.jpg used for evaluation.



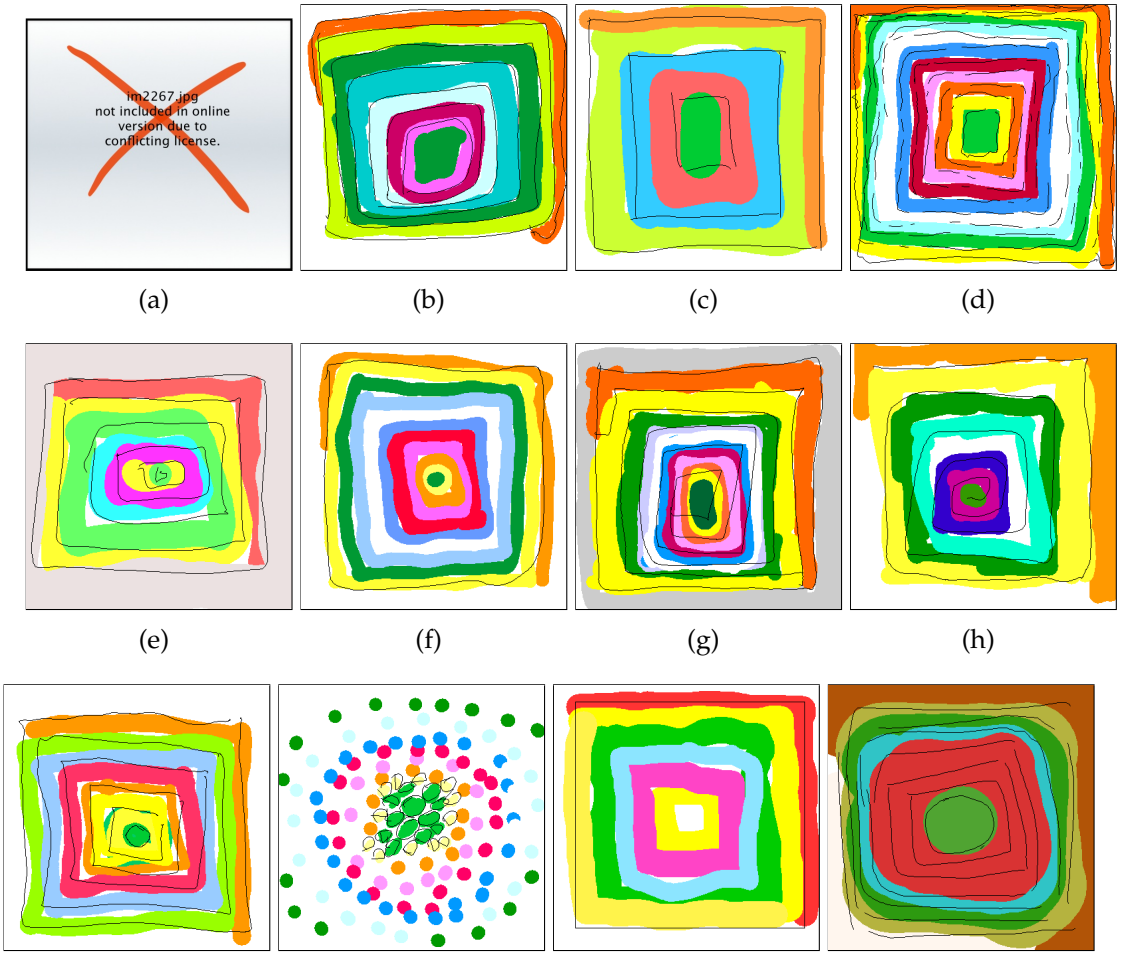
(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

im5820.jpg: 'Kue Mangkok' by Riana Ambarsari (p3nnylan3), License: CC BY NC ND

Figure 10.31: Sketches for image im5820.jpg used for evaluation.



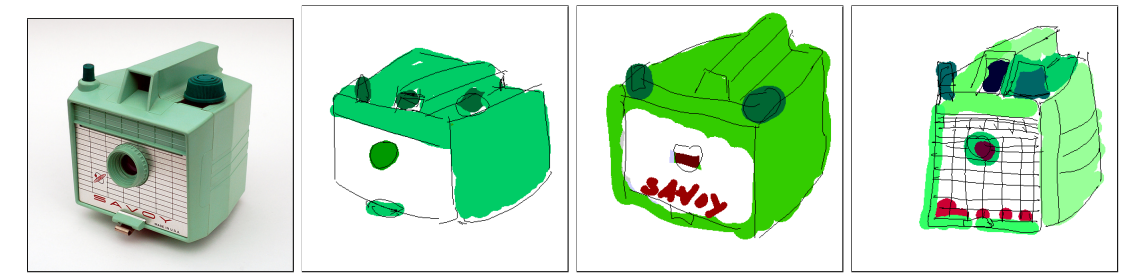
(a) (b) (c) (d)

(e) (f) (g) (h)

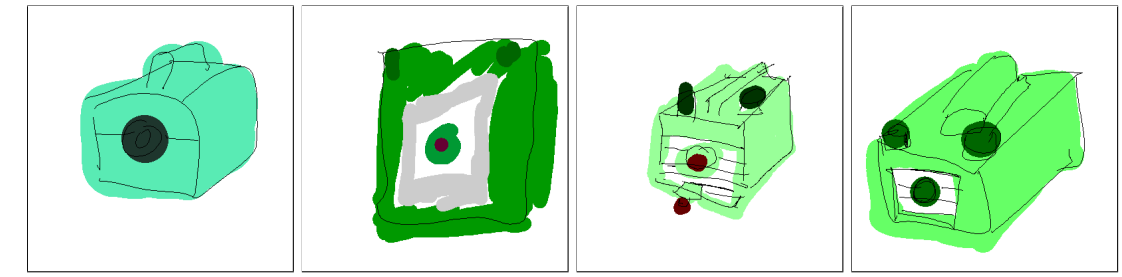
(i) (j) (k) (l)

im2267.jpg: 'Happier than happy square' by Carina Envoldsen-Harris, License: CC BY NC ND

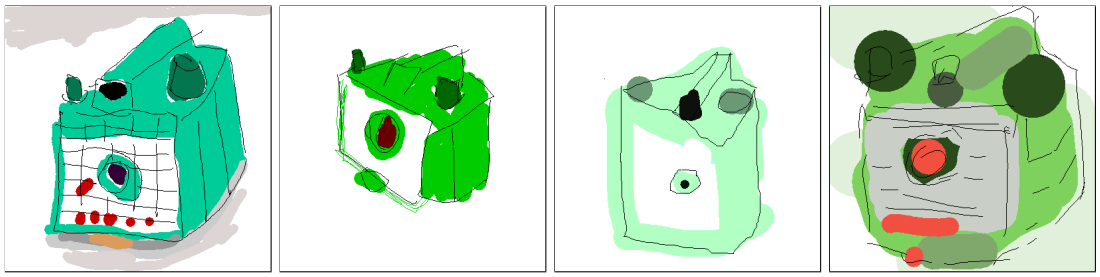
Figure 10.32: Sketches for image im2267.jpg used for evaluation.



(a) (b) (c) (d)



(e) (f) (g) (h)



(i) (j) (k) (l)

im10829.jpg: 'Imperial Savoy' by John Kratz, License: CC BY

Figure 10.33: Sketches for image im10829.jpg used for evaluation.

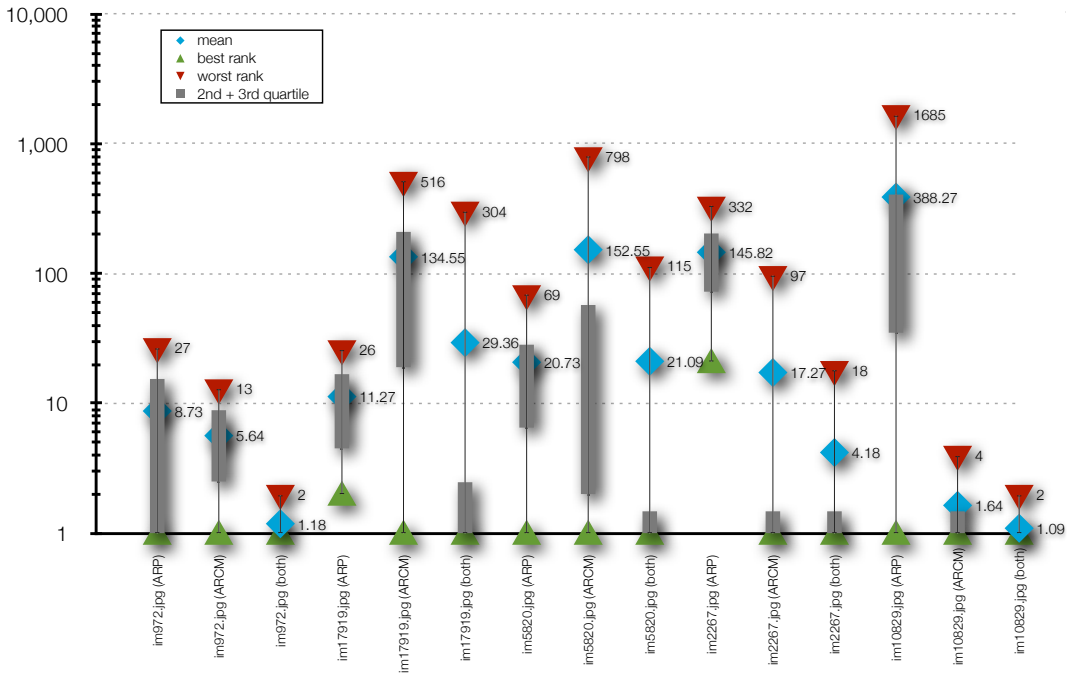


Figure 10.34: Ranks of Known Items (Text Filter off), Edges and Color

Figure 10.34 shows the results for the search for the known items in Figure 10.27. As in previous evaluations, a number of parameters can be set by the user to influence the search, e.g., the number of partitions, enable/disable invariances, the ϵ value for adjusting the invariance for deviations in color, the edge threshold β (cf. [Giangreco, 2010, pp. 32f] for details). The numbers represent the best search results achieved with these parameters.[35]

The combination of ARP and ARCM performs very well: For most sketches, the purely-visual search with best parameters achieves a ranking where the sought item is found at rank one. Furthermore, the sought item was not returned within the first

[35] [Giangreco, 2010, pp. 37] shows also the results averaging over the 10 best configurations of parameters rather than just the best search, which can be expected to be more robust against "over-training" as the combination of all parameters for searches based on edges and all parameters for searches based on color increases the effective number of configurations. In this average, the ranks of the worst searches doubled; with the worst value being 1230.8. However, the average ranks increased only by a factor of approximately 1.3. From the user's perspective, considering only the best search configuration represents the result that the user is most interested in; in some cases the user will only achieve these settings by trying out many searches with the same sketch. Considering the 10 best searches gives a higher probability that the user will find *any* of these; but in there entirety, the user will never really use them: If one search shows the sought item, the user can end the task successfully and does not need to search any further. As both ways of evaluating show similar overall trends, we present in this thesis only the approach using the best overall configuration.

10 results only for the one sketch of im17919.jpg shown in Figure 10.30(i)[36] and two sketches of im5820.jpg (Figure 10.31(g) and 10.31(k); in both cases color was too far off).

As these search results do not use any metadata like tags for search, it shows that the combination of different, complimentary features can deliver very good retrieval quality to the user. Still, even if the presented results are promising: It depends on the individual search task whether the same search strategies with the presented features can be used. This depends mainly on:

- Availability: Not all image collections provide sufficiently rich metadata for faceted searches or text-predicates that are meaningful to the user.
- Convenience: Not all images are easy to draw, in particular when the user has to draw from memory.
- Matching Tolerance: The users input for individual features will not always be close enough to the sought image to separate it nicely from all the other images. For instance, in a big collection of medical images, a generic sketch of a bone will most likely match an unmanageable number of images.

Texture

In [Kreuzer, 2010] the Edge Histogram Descriptor (EHD) [Park et al., 2000, Manjunath et al., 2001, pp. 713f] has been implemented with support for optional global and semi-global edge histograms as proposed in [Yamada et al., 2000]. If neither global nor semi-global histograms are used, background invariance is supported in the same way as for ARP: Empty areas of the 4×4 regions as shown in Figure 5.5(a) – 5.5(b) are ignored in the distance computation.

Figure 10.35 shows the results for the same images as in Figure 10.35, now including EHD. For im1660.jpg (cf. Figure 10.21(a)), EHD performs slightly worse than both, ARP and IDM. For all other images, the results are very poor.

It is important to note that this is *not* just due to the different feature descriptor, but also relates to how the sketches where acquired: The users had time to familiarize with the system at a time when only ARP and IDM were supported. Furthermore, the users were drawing the sketches until they found the sought image using ARP or gave up. As ARP does not take into account the edge direction, but only the spatial placement of edges. Thus, the histogram of edge directions used in EHD will not be optimal as users did not get any feedback on their sketches w.r.t. edge direction and therefore are likely not to have paid strong attention on that property.

In [Kreuzer, 2010], a different input device (digital pen and interactive paper) was used with new sets of target images. One of the three datasets used in the evaluations in [Kreuzer, 2010] is the MIRFLICKR-25000 collection. Figure 10.36(a) shows an example image, for which EHD performs significantly better with the sketch presented in Figure 10.36(b) than both features used in previous evaluations, ARP and IDM. One

[36] For the sketch in Figure 10.30(i), edges alone performed very well when the threshold β is set high such that only very prominent edges are preserved; therefore the user may not have paid much attention to the colors as the sought item was already found.

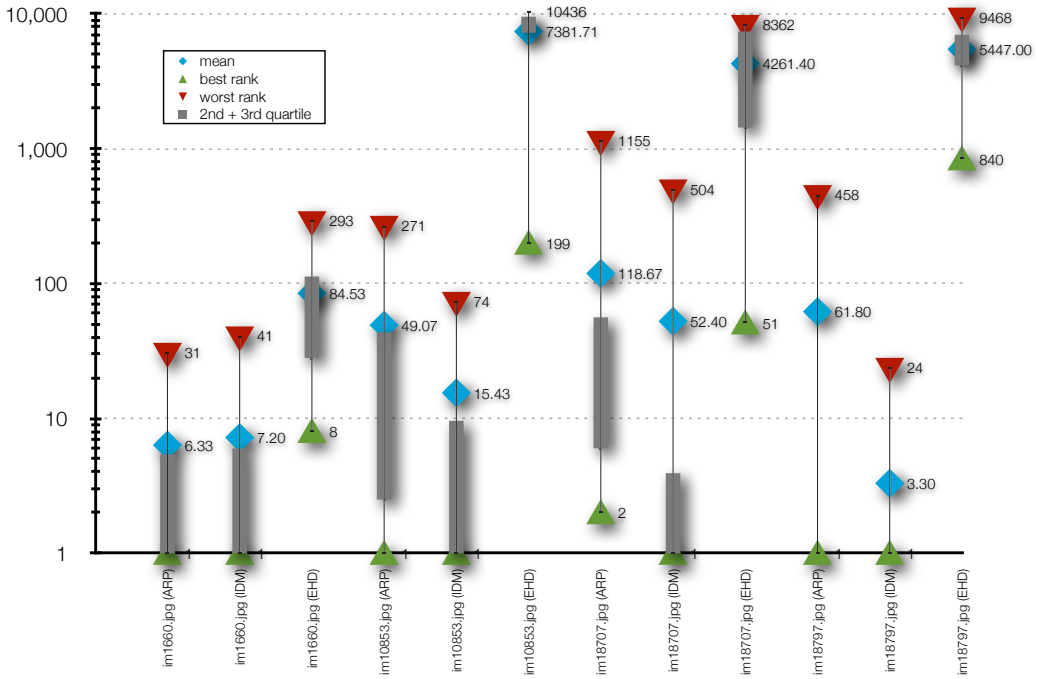
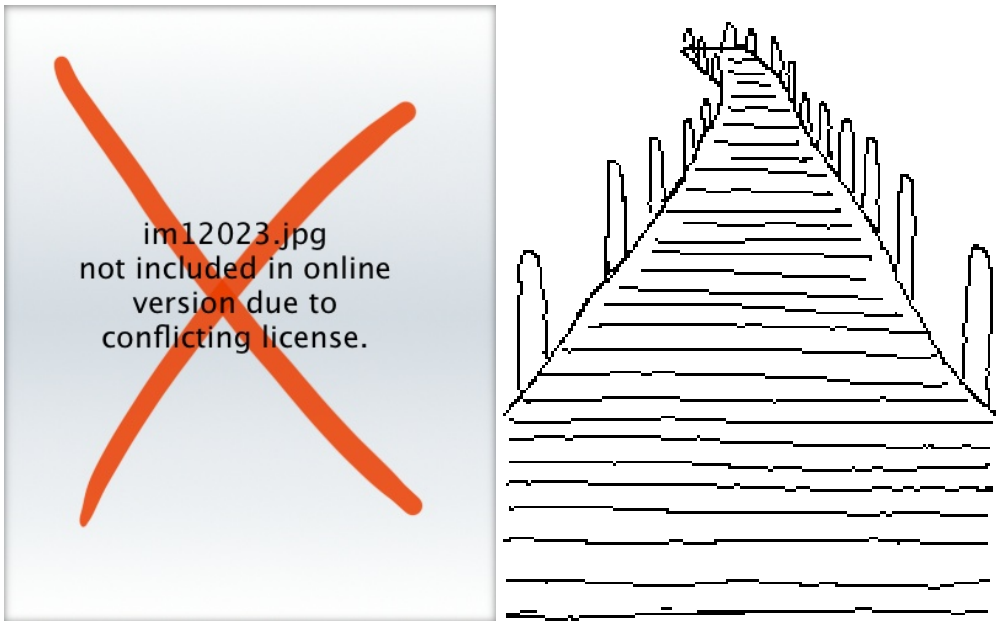


Figure 10.35: Ranks of Known Items (Text Filter off), including EHD

major reason for this is the limited depth of field and perspective in the picture that results in blurred lines in the background for the edge detection used in ARP and IDM: In the upper part of the image, even at very low values of β like 2 and 5, no individual horizontal lines for timber were found as they were drawn in the sketch. When looking closely at the picture, even in the highest resolution available on the web[37] of 400×500 pixels, no such horizontal lines can be seen. Removing these details –or marking these areas as *unknown*– improves the ranks in both, ARP and IDM. Nevertheless, it shows that in some cases, EHD can perform well and that users may sometimes draw a sketch that includes information that is not present in the sought image, but exists in the users experience of the real world. The latter will have greater impact for images drawn from memory than for images which the user could see and visually analyze or even trace while sketching.

In total, the evaluation of 45 purely visual searches for images by a single user in [Kreuzer, 2010, pp. 56–63] found 87% images at rank 1 using IDM, 71% using ARP, and 4% using EHD. With the two other, less challenging datasets that contained 791 cartoon characters [Kreuzer, 2010, pp. 64–66] and 812 paper watermarks [Kreuzer, 2010, pp. 67–70], the performance of EHD was better, although IDM remained the best descriptor with retrieving none of the images at a rank worse than 20. For ARP, this was

[37] Original image: `http://www.flickr.com/photos/monster/1658643576/`



(a) Original Image im12023.jpg (b) Sketch

Figure 10.36: Search for im12023.jpg: Best rank using ARP is 39, ARP+IDM is 95, best rank using EHD is 7.

also the case for none of the cartoon characters and in 16% of the cases for the watermarks; for EHD, in 1.5% of the cases for the cartoon characters and 19.38% of the watermarks.

Also in the context of [Kreuzer, 2010, pp. 72–79], a multi-user study with eight participants on the MIRFLICKR-25000 dataset was performed. Six images have been chosen from the library. We chose a simple image for the start and then increased the challenge by selecting more demanding target images.

Figure 10.37 - 10.42 on the following pages, show the target images and all the sketches created for this evaluation. Under each sketch the best retrieval results are noted, along with the search settings used for the query. Whenever abbreviations are used, they are explained in the legend on the top right of the figures.

The sketches in the figures 10.37 - 10.42 have been ordered left to right, top to bottom according to the quality of the retrieval rank of the target image. In addition, the sketches have been framed with a colored border.

Sketches with a blue border have led to the target images with only little effort with a retrieval rank of the target image smaller or equal to 10. When using ARP in combination with IDM, only searches done on a selection of images found with the ARP default search settings count toward achieving a blue border. If several searches using IDM had to be started to get a top 10 rank, or when the target image has been found among the first 20 results but not among the top 10, the sketch has be marked with a yellow border. All sketches that could not be found within the first 20 results have a red border. Whenever other means were available to get acceptable search results for these red framed sketches, it has been noted in their description.

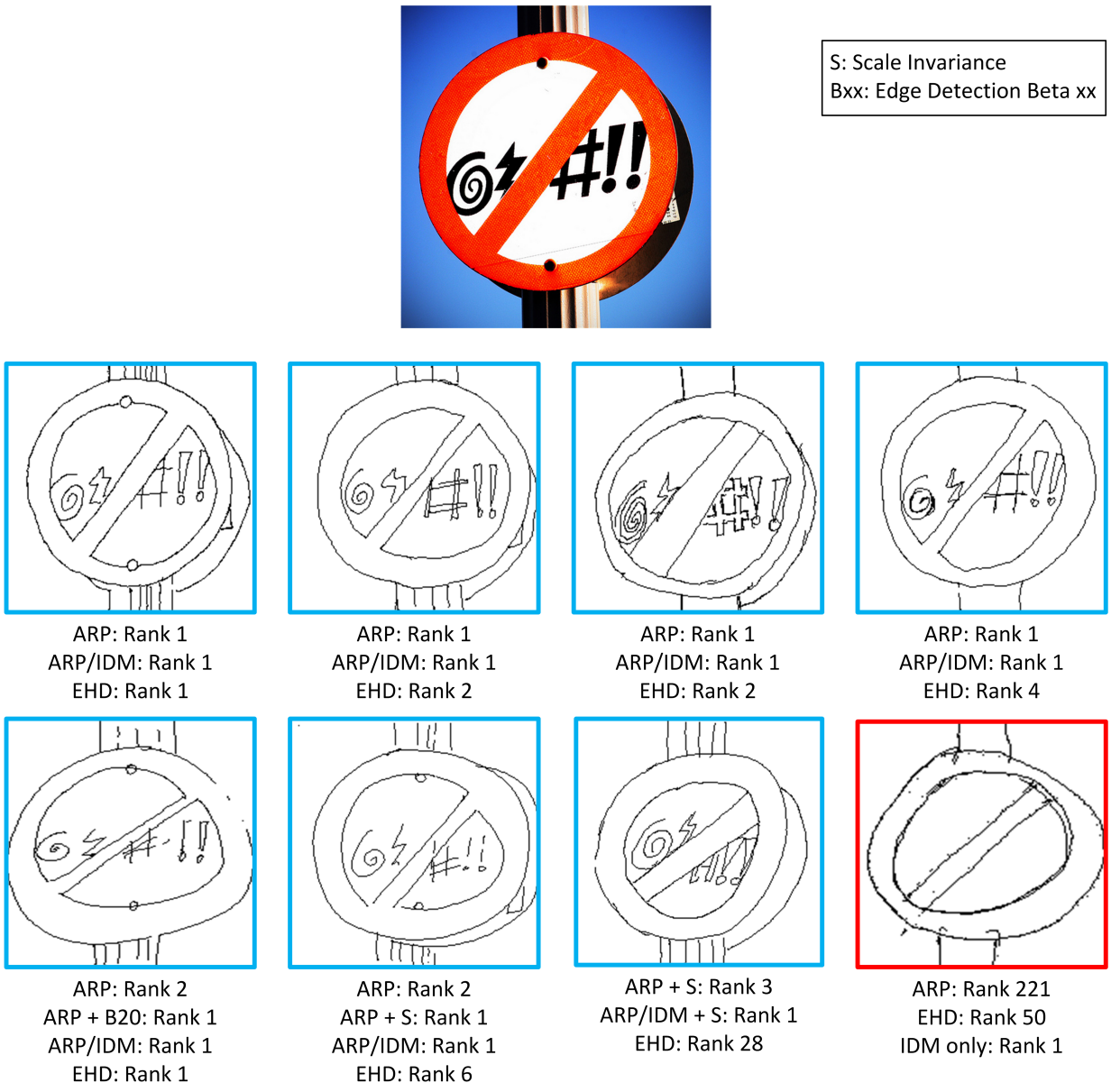


Figure 10.37: Search for im1561.jpg [Kreuzer, 2010, p: 73]: A simple target image has been chosen to get the users acquainted to the application and hardware. All feature descriptors perform very well with their default settings, therefore most searches showed the target image in top ranks.

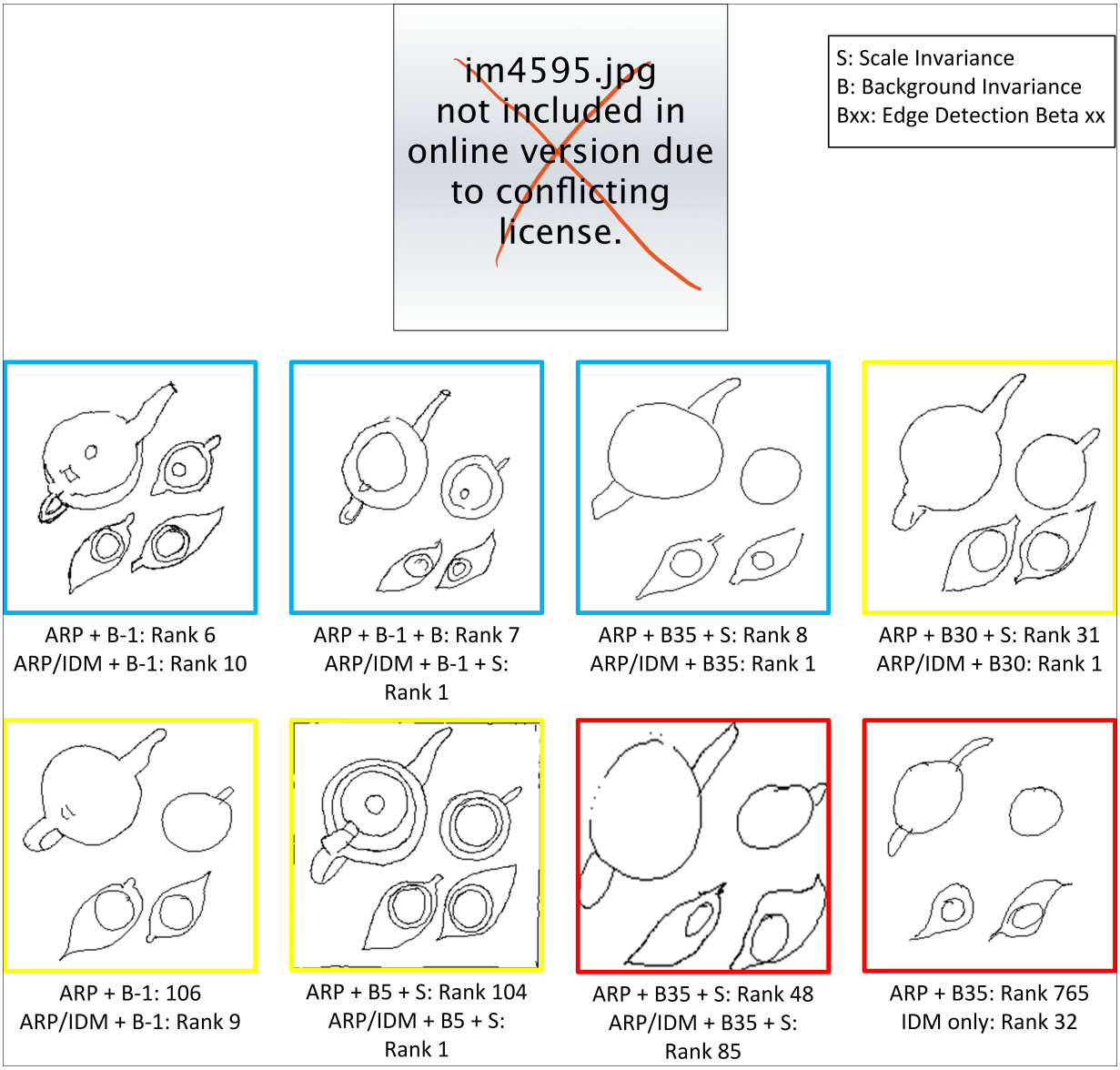


Figure 10.38: Search for im4595.jpg [Kreuzer, 2010, p: 74]: The second image chosen for the evaluation turned out to be more challenging than expected. The target image may be rather simple, but turned out to be difficult to sketch in terms of scale and relative placement of the objects in the picture. Background and scale invariance have helped to improve the results. With the use of IDM, 8 out of 10 sketches led to the target image among the best 10 search results, but additional parameters often had to be set for the search. ARP performed better when varying the beta for the edge detection.

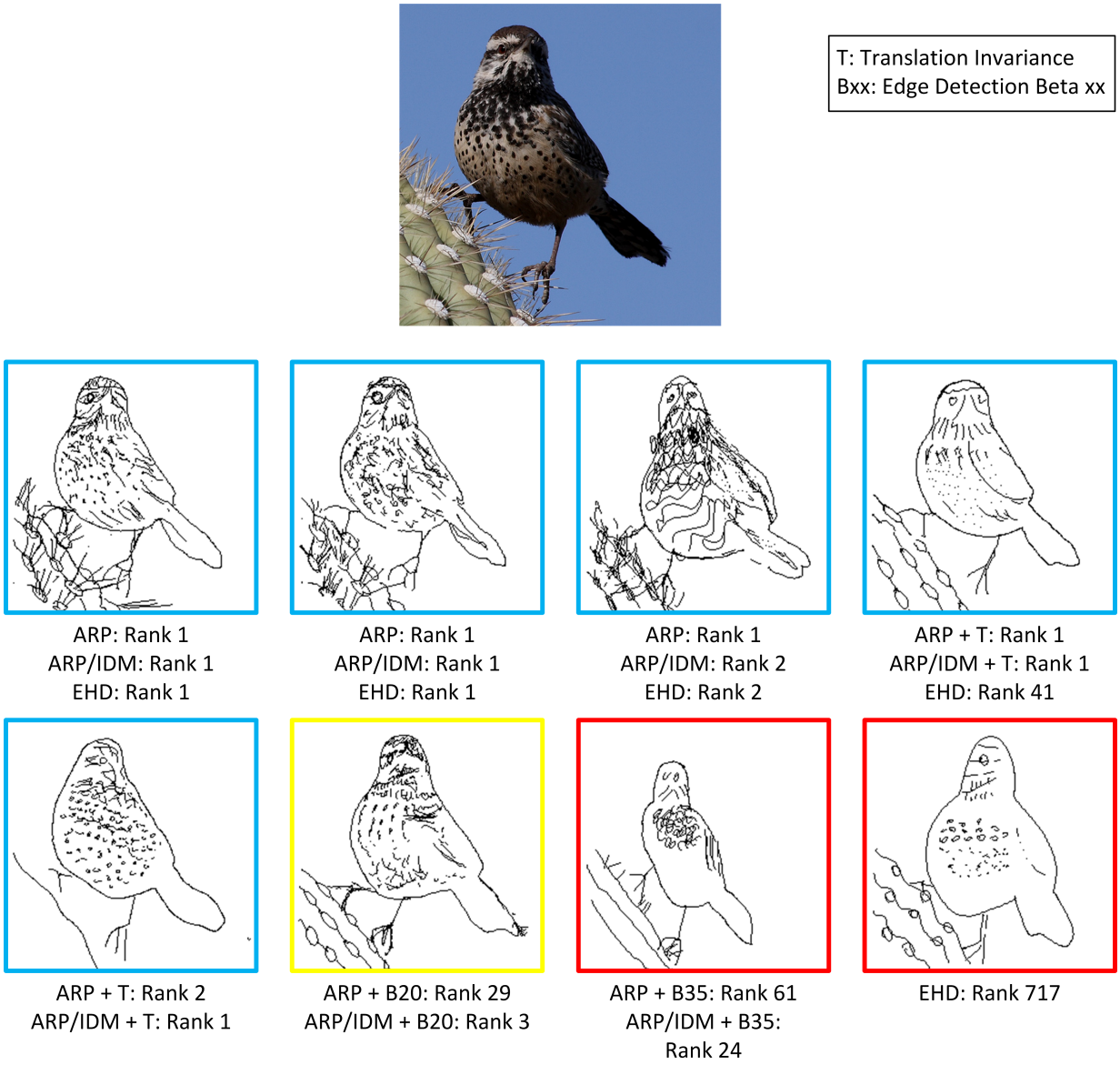


Figure 10.39: Search for im7799.jpg [Kreuzer, 2010, p: 75]: The challenge in sketching the image lies in sketching lines on the body of the bird as these patterns are essential for a good search result. The sketches show that it's not important to get the pattern right or exactly match the form of the bird and most of the sketches have led to a perfect match. ARP and EHD perform well with their standard configurations and translation invariance seems to be helpful in some cases.

S: Scale Invariance
R: Rotation Invariance
Bxx: Edge Detection Beta xx

ARP: Rank 4
ARP + B-1: Rank 1
ARP/IDM: Rank 1

ARP: Rank 6
ARP/IDM: Rank 1

ARP + B-1 + S: Rank 12
ARP/IDM: Rank 1

ARP + B5 + S: Rank 5
ARP/IDM: Rank 1

ARP + S: Rank 22
ARP/IDM: Rank 1

ARP + R: Rank 134
ARP/IDM + R: Rank 2

ARP + B5 + S: Rank 396
IDM only: Rank 19

ARP + B15 + R:
Rank 478

Figure 10.40: Search for im11541.jpg [Kreuzer, 2010, p: 76]: Most sketches performed very well for the chosen image, especially the combination of ARP and IDM has led to the target image quite often. The yellow framed sketch performed well, only when using rotation invariance. This might be because the pumpkin's mouth: the sketch is rotated to the right a little. This part is essential for the retrieval, since it contains some very strong edges. Scale invariance has helped in improving results from sketches where the pumpkin has been drawn too small.

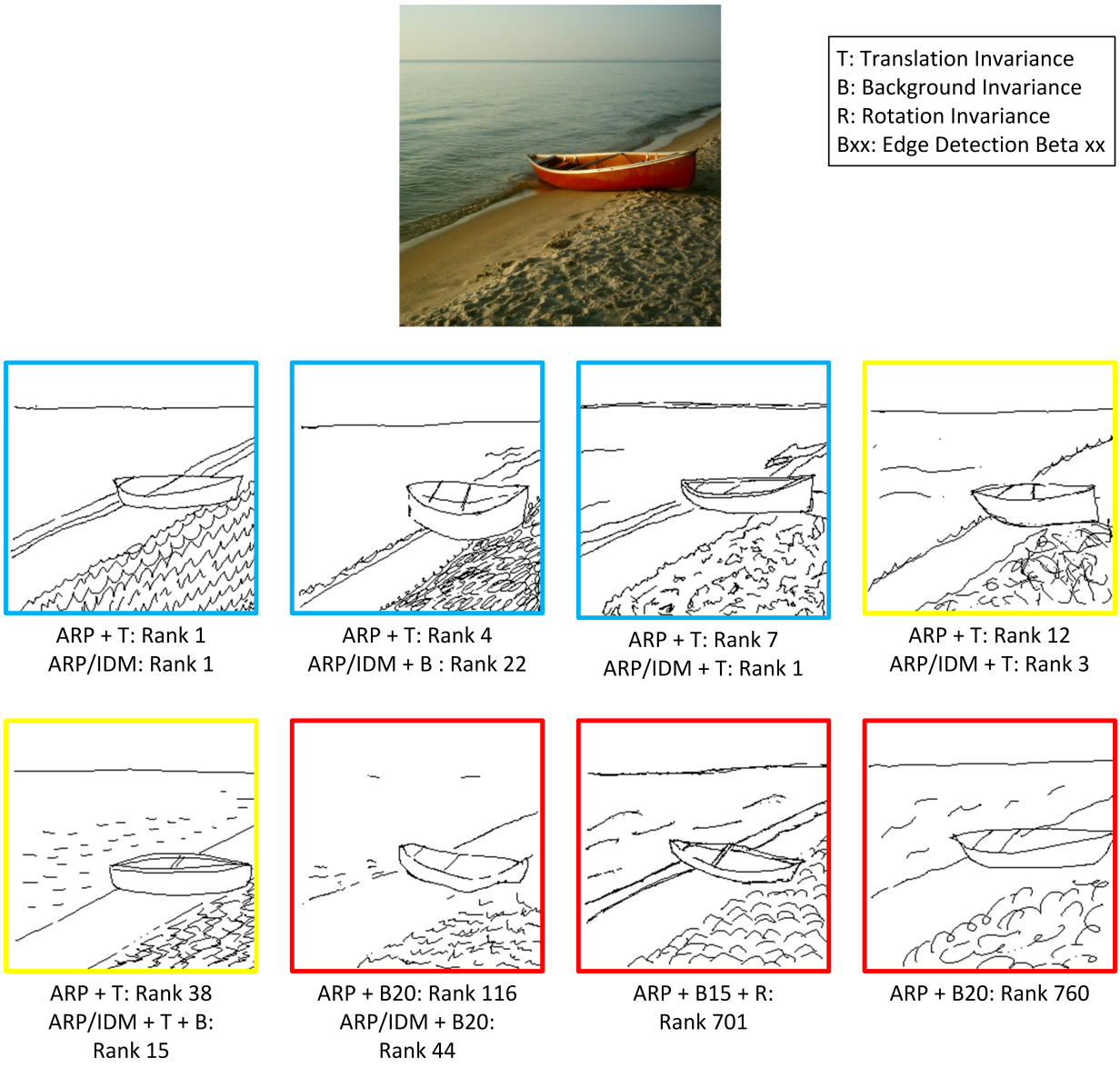


Figure 10.41: Search for im18791.jpg [Kreuzer, 2010, p: 77]: The sandy beach in the lower right corner of the image contains some strong edges. The sand pattern will disappear in the edge representation of the image, when increasing the beta value for the edge detection – but only after many other edges (e.g., horizon, boat) are no longer found in the edge detection. Therefore, for acceptable search results, some pattern has to be drawn in the area with the wavy sand. This has shown to bee more essential for acceptable search results than exactly matching the position or shape of the boat.

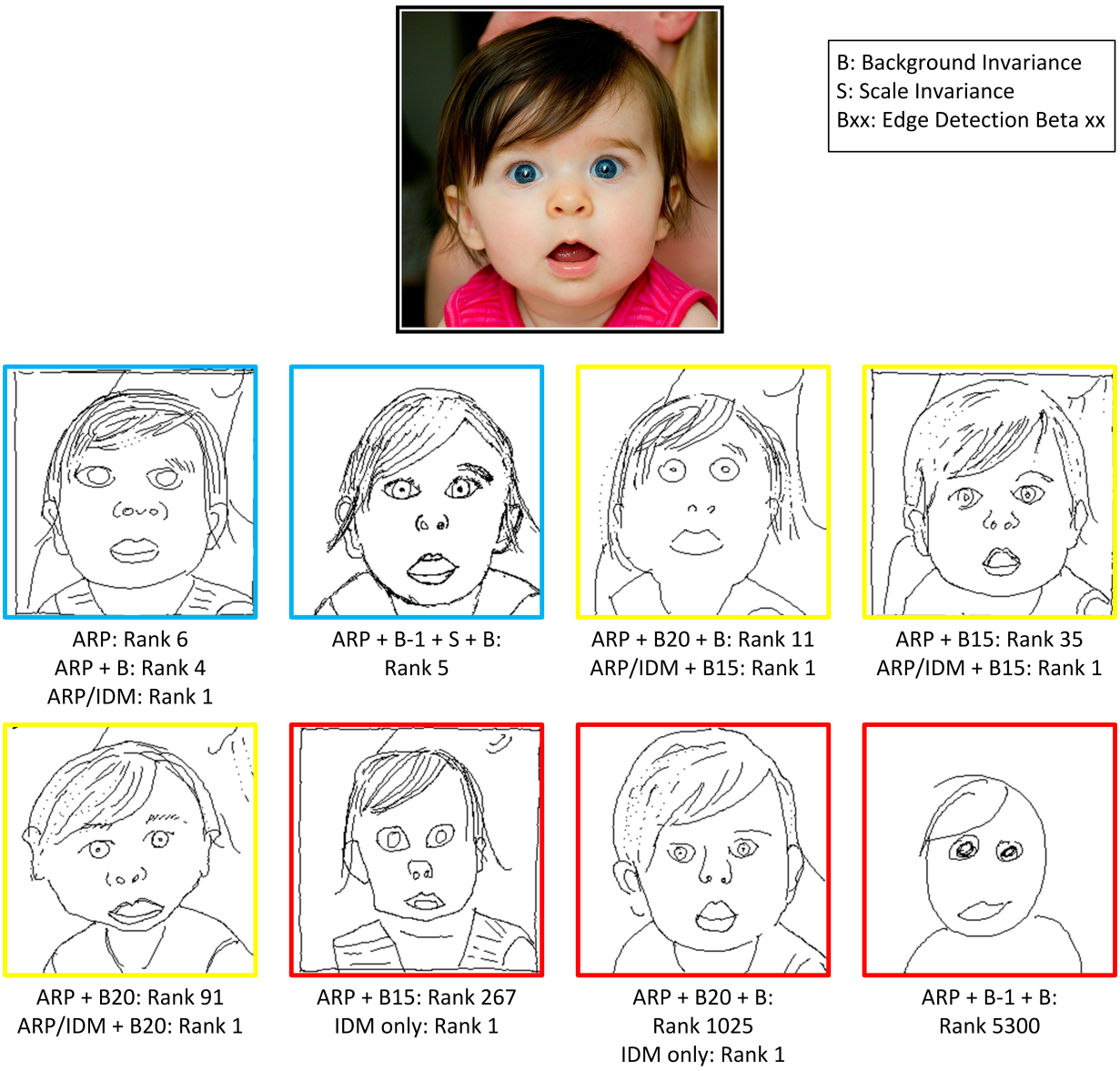
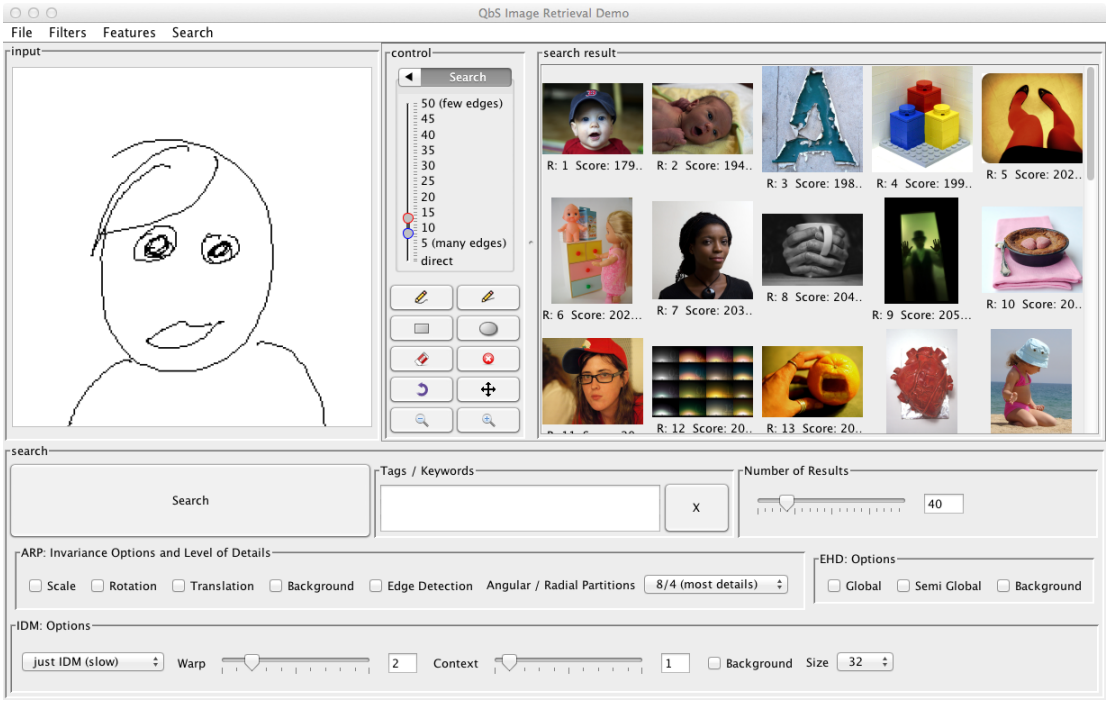


Figure 10.42: Search for im15816.jpg [Kreuzer, 2010, p: 78]: For this image, background invariance was helpful when the user didn't draw the frame around the image. As the MIRFLICKR-25000 collection contains many images of faces and similarity was evaluated purely based on visual appearance, it is not sufficient to sketch something that is recognizable as a face to find the sought image. This can be observed especially with the last sketch on the bottom right of the figure, for which Figure 10.43 shows search results.



(a) Screenshot of Search

(b) Sketch

(c) Top Rank Result

Figure 10.43: Search results for the worst sketch from Figure 10.42: A screenshot in 10.43(a) shows the results using IDM, for which it can easily be explain why based on the spatial distribution of edges these images are considered much more similar to the sketch than the sought image. The sketch is shown enlarged in 10.43(b), the image im6140.jpg which was ranked first (not the sought item) is shown in 10.43(c). In this search, the sketch is simply not sufficiently close to the sought image im15816.jpg from Figure 10.42.

10.4 Fine-Grained Matching Tolerance for Retrospective Geotagging

10.4.1 Motivation: Space and Time in Photography

For organizing human life, space and time have always had great importance. Organizing and retrieving images is no exception to that and space and time can provide very helpful dimensions in faceted searches, for instance, for known image searches as described in Chapter 2.2.1, but also themed searches and retrieval by class.

Recording the time when an image was captured is in most cases no longer an issue: Even in the time on analog photography, some (compact) cameras could embed the current time on the pictures; digital cameras have better means to track the time and store it at least as the file date, but usually record it also in the Exif metadata [JETIA, 2010] where it will remain unaffected of image manipulation that may be performed at later stages of the processing workflow. Errors in the range of hours may occur when the local timezone is not set properly or the (local) daylight saving time is not used properly, but these errors can usually be corrected easily in post-processing and even if not corrected, do usually not affect retrieval performance as long as the time inaccuracy does not exceed days and the relative order of shots at a single event does not get mixed up. As part of processing workflows, the information may also be converted to other metadata formats like Information Interchange Model (IIM) of the International Press Telecommunications Council (IPTC, therefore IIM is frequently referred to as "IPTC fields") and Extensible Metadata Platform (XMP) [Tesic, 2005].

Geotagging

All these formats not only allow to store the time at which the picture was captured and the parameters of the camera like exposure settings, focal distance and aperture, usage of flash, white balance, attached lens, etc. but also the location at which this happened based on the geographic coordinates [Tesic, 2005]. With the last attribute mentioned in the list it is possible to assign geographic coordinates consisting of latitude and longitude to images, a process which is known as *geotagging*.

Let $(lat, long)$ be a geospatial coordinate with lat being the latitude ranging from -90° (south pole) to +90° (north pole) and $long$ being the longitude ranging from -180° to +180° with 0° being the Prime Meridian that passes through the Royal Observatory, Greenwich, UK. Furthermore, let $\mathfrak{G}$ be the space defined by all valid geospatial coordinates, which is illustrated in Figure 10.44. Let $\mathcal{I}_{\mathcal{G}}$ be subset of images for which the geolocation is known. Then it is straight forward to derive a function that given such an image $\mathfrak{I} \in \mathcal{I}_{\mathcal{G}}$, returns its location:

$$location : \mathcal{I}_{\mathcal{G}} \rightarrow \mathfrak{G} \tag{10.10}$$

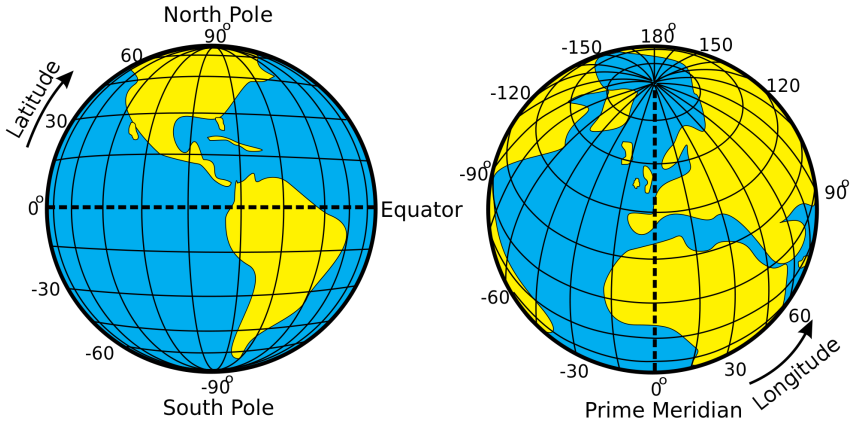


Figure 10.44: Illustration of Latitude and Longitude (Illustration by Djexplo)

The critical part therefore remains to acquire knowledge of the location at which an image has been taken as this imposes some challenges on either the used technology or the user and is therefore not as ubiquitous as storing the time.[38]

While recording the time does not increase the cost for camera manufacturers significantly, to automatically determine the geographic coordinates it requires additional hardware to receive the signals from a global navigation satellite system (GNSS) and compute the position of the device from these signals. The most-commonly used GNSS is the U.S.-american NAVSTAR Global Positioning System (GPS)[39] and allows position measurements with an accuracy that should not differ from the real position by more than 100 meters.[40]

[38] Another aspect that reduces the use of geotags is an non-technical issue: Privacy. While the date at which a picture was taken cannot be linked directly to a single person or small group of people, this is frequently possible for location at which a picture was taken, e.g., if the location belongs to an non-public place like a private house – and in combination with other information which might be captured inside the image or its metadata, even public locations can rapidly reduce the circle of people involved in the production of a picture (either by taking the picture or being displayed in it). These privacy concerns are of great importance when images are shared and may spread over the entire internet. It is therefore reasonable that some tools for sharing images provide options to remove the geotags when uploading images – or even before taking a picture, sometimes users don't want the geolocations to be stored at all. In the context of retrospective geotagging, we are not interested in assigning geotags where users did not want to include such tags, but focus on situations in which the users involuntarily did not geotag images –in other words: did not decide to avoid geotags due to privacy concerns– and would like to annotate images with geolocation and use them in retrieval.

[39] Other GNSS are the Russian *GLONASS* (already in operation) and systems still being deployed like the Chinese *Beidou/COMPASS*, European *Galileo*. There exists also some regional navigation systems that do not provide coverage for precise location measurements all around the globe like the earlier Chinese system *Beidou 1*, the French *Doppler Orbitography and Radio-positioning Integrated by Satellite (DORIS)*, the *Indian Regional Navigational Satellite System (IRNSS)*, and the Japanese *Quasi-Zenith Satellite System (QZSS)*.

[40] The quality available for military use is usually much higher than for civilian use due to the intentional degradation of quality known as *Selective Availability (SA)*. Since May 2000, the Selective Availability should no longer introduce errors greater than 20 meters; however, in situations of poor reception or if not all available satellites have been discovered by the GPS device yet, the overall error may still exceed 20 meters.

(a) Phone: Map (b) Phone: GPS (c) GPS Tracker outside (d) GPS Tracker inside

Figure 10.45: Devices for tracking the position while tacking pictures: (a) shows a smartphone with built-in GPS. The application shows the position on the map; pictures taken with the smartphone camera are automatically tagged with the position. Running a different application as displayed in (b), the same smartphone can log the position to create a track that can later be used to tag images taken with a different device via timestamps. The same functionality can be provided with a dedicated device like the one shown in (c). However, this only is possible as long as the GPS signal is received. In (d), this was not possible while being inside a building.

Such GPS devices have recently become more common as being built inside smartphones, in first place to provide navigational assistance as shown in Figure 10.45(a). However, the operation consumes battery and this, in addition to the added cost has so far restricted the number of digital cameras that provide built-in GPS coordinate recording. There are external devices available, so-called "GPS tracker" like the one presented in Figure 10.45(c) that provide the functionality in case the camera does not have GPS built in – but even then there remain several occasions in which the coordinates cannot be recorded accurately:

1. As GPS is based on signals received from satellites, objects obstructing the signals can deteriorate the position measurement – or even make it impossible, in particular when picturesare taken inside buildings as shown in Figure 10.45(d).

2. When using external GPS trackers, it happens sometimes that these devices run out of battery during a photo session / journey without the user recognizing this situation until the end of the shooting. This is due to the fact that most of these external devices record an entire track without any need for user interaction. The track is matched afterwards to the timestamps of the pictures from the camera[41], thus restricting the interaction with the user during operation to turning the device on before shooting and off after shooting – so the user may recognize the problem of low battery too late.

[41] Exception being mostly (more costly) devices which are connected via Bluetooth or the flash hot shoe of the camera to tag the images at the time of recording.

3. Images that have been taken before GPS devices were easily available and are probably digitized just now cannot be assigned automatically with a correct GPS signal.

If no automatic location tracking due to any of these reasons and no content-based methods are used, manually assigning geotags remains a fallback solution. This can either be done by setting the location on a map or by looking up the geolocation of a place, e.g., of a well-known landmark. The latter is known as "geocoding" and depending on the information that the user can provide, is unfortunately in many cases ambiguous. Let $\mathfrak{T}_{\mathcal{G}}$ be the space of textual information that the user will provide to describe a geolocation with words, then *geocode* is a function that returns a set of possible geographic coordinates together with an additional textual description to disambiguate the results:

$$geocode : \mathfrak{T}_{\mathcal{G}} \rightarrow \mathfrak{G} \times text \tag{10.11}$$

As presenting a map is usually much more helpful for the user than presenting just the coordinates as numeric values, the disambiguation is frequently assisted by such a map. In addition, this also may allow fine-adjusting the location on the map. The latter is commonly performed if high accuracy of geotags is desired; this becomes necessary due to the fact that images of object are shot from a position slightly away from the object, but the geocoding is more likely available for the object itself, not the place of taking the image. Such manual work to disambiguate the coordinates returned by *geocode* and performing the fine-adjustments can become very time and labor intensive, in particular when done accurately and not just roughly assigning a place, e.g., the city in which a picture was taken.[42] The amount of manual work and whether it can ultimately be successful at all depends directly on the quality of the textual information that the user can provide: If the user does not remember the name of the place where an image was shot, no matter how well the implementation of *geocode* works, it will not be able to return any helpful results.

Examples of dedicated software for geotagging are GeoTag[43], GeoSetter[44], or HoudahGeo[45]. All of them provide the functionality to geotag images either based on GPS tracks or manually by setting marks on a map and may also provide assistance through geocoding. But many applications or websites for managing images have such functionally now also built-in, therefore no longer requiring additional software for this task. Examples are Flickr[46], Google Picasa and Panoramio[47], "Places" in Apple iPhoto

[42] The opposite process of turning geocoordinates back into names of places or a street address is called *reverse geocoding* and allows for instance also to search with keywords even if an image has not been assigned any keywords, just geotags.

[43] `http://geotag.sourceforge.net/`

[44] `http://www.geosetter.de`

[45] `http://www.houdah.com/houdahGeo/`

[46] See FAQ on Map `http://www.flickr.com/help/map/` and Organizr `http://www.flickr.com/help/organizr/`

[47] See `http://picasa.google.com/support/bin/answer.py?answer=161869` and `http://www.panoramio.com/`

and Aperture[48], Microsoft Windows Live Photo Galery[49], DigiKam[50], and Sony Picture Motion Browser[51].

10.4.2 Content-based Approach to Retrospective Geotagging

When an image has been taken but the location was not stored, the image content can still be used to determine –or at least estimate– the geocoordinates. Such an attempt can therefore be seen as a generalization of Equation (10.10), in which the image does no longer need to be from the subset of images with known geotags, but just an arbitrary image:

$$location_{CBIR} : \mathcal{I} \rightarrow \mathfrak{G} \tag{10.12}$$

Of course, not all image content can serve this purpose equally well: Moving objects and people in the picture or fairly generic objects that are found in many places provide less useful clues about the place than stationary objects. Furthermore, to be able to determine the geocoordinates through content-based methods, it is necessary to also have a ground-truth of geotagged images of the same objects.

We name the approach to geotag images *after* the creation of the image "retrospective geotagging" to clearly distinct this from geotagging which is performed by a device at the time of the creation of the image. If we consider the image-task model proposed in Chapter 2.5, the content-based approach to retrospective geotagging can be characterized as:

- The Task Input and Aim consists in a first stage of an *Object Classification task*: Identify location through prominent objects like buildings or other landmark. This is followed in a second stage of *Retrieval by Class*: Find images of the same objects, ideally shot from the same viewpoint.
- The Matching Tolerance has great emphasize on *Local Matching Tolerance*: Areas of the image not belonging to the object itself can/should be ignored, in particular moving objects and people. Some parts of the background can provide helpful insights, in particular if they define a landscape, while other parts like a blue or cloudy sky is not. The context of the images is also of importance: The user is only interested in images that are placed somewhere near the places where the user has actually been; if there are almost identical objects, only those are relevant which the user passed by – which limits the matching tolerance not based on the image content, but on the image context.
- The result usage is extremely *representation-oriented*: Only the geocoordinates of the found images will be used.

[48] See `http://www.apple.com/ilife/iphoto/what-is.html` and `http://www.apple.com/aperture/what-is.html`

[49] See http://blogs.msdn.com/b/pix/archive/2008/04/07/geotagging-photos-with-windows-live-without-using-a-gps-device.aspx

[50] See http://scribblesandsnaps.wordpress.com/2009/11/03/geotagging-photos-with-digikam/

[51] See `http://vaio-online.sony.com/prod_info/software/picture_motion_browser/index.html`

Existing Approaches to Determine Location using Image Content

There have been some attempts to solve this problem by content-based methods to implement Equation (10.12) directly. Just considering the aspect of determining the position of a camera given some known environment is essential for vision–based positioning in robot vision and techniques for this have been described for instance in [Borenstein et al., 1996, pp. 207–217] and more recent results mentioned in [Caputo et al., 2009, Pronobis et al., 2010]. There has also been a task "Where am I?" at the 10th IEEE International Conference on Computer Vision (ICCV 2005)[52] for which the participant had to estimate the GPS coordinates from a single image given the training data. An early system for performing the task of location recognition is described in [Yeh et al., 2004] in which the location of a person with a mobile phone is determined by taking an image with the built-in camera and sending this image to server where it will be matched with images of landmarks. Therefore the focus in that approach was on operating a single picture taken with a phone. In DAVID [Del Bimbo et al., 2009], the focus was slightly different on detecting and verifying monument appearance in images. Particular interest was given to learning to discriminate the monuments even when images of each were taken from different viewpoints using only the most salient features. It is therefore not the precise location –which depends on the viewpoint– but the object inside the image that is determined. The approach uses SURF keypoint descriptors and was evaluated with 159 images taken with the camera of a cell phone of 12 different monuments in Florence.

With enough pictures from a single location, even the reconstruction of a scene using only the images themselves is possible [Snavely et al., 2006, Snavely et al., 2008b, Snavely et al., 2010], showing that very precise annotation of a location is possible.[53]

The problem of identifying the location of an image given just its content and a big set of labeled training images can also be seen as a machine learning task. [Hays and Efros, 2008] performed experiments with a training set of over 6 million geotagged images to identify the location of test set of images, where the location was hidden. A similar setup has also been used as a showcase for Google Search by Image [Google Inc., 2011c, Singhal, 2011] which is technologically backed by [Jing and Baluja, 2008]. Incorporating prior knowledge like temporal relation to other shots can significantly improve the location estimation [Kalogerakis et al., 2009, Crandall et al., 2009, Li et al., 2009b].

[52] Contest website and dataset available at: `http://research.microsoft.com/en-us/um/people/szeliski/visioncontest05/default.htm`, unfortunately the data seems to be not up-to-date and the GPS locations for the images is not available; list of participating teams and used techniques can be found at `http://www.cs.ubc.ca/~deaton/iccv2005/approaches.html` and `http://cmp.felk.cvut.cz/publicity/iccv-contest05/` and the winning entry later published their experiences in [Zhang and Kosecka, 2006].

[53] [Tuite et al., 2011] shows how a "game with a purpose" similar to the idea behind [von Ahn and Dabbish, 2004, Russell et al., 2008] can be used to motivate people to capture images of real-world objects for which not enough information for a precise reconstruction was found online and therefore "crowd source" the most labor-intensive aspect of the task.

These approaches using big datasets are already close to solving the retrospective geotagging task, however, they are not very user-focussed to enforce the matching tolerance.[54] [Hays and Efros, 2008] states w.r.t. the used dataset and evaluation:

> "But as figure 2 shows the data is very non-uniformly distributed towards places where people live or travel which is fortunate since geolocation query images are likely to come from the same places."

This implies that most importance should be given to places that people in general travel to frequently. A similar bias was introduced in [Crandall et al., 2009] with focusing on the most photographed landmarks:

> "Table 3 presents classification results for the ten most photographed landmark-scale locations in each of ten most photographed metropolitan-scale regions. In each case the task is to classify photos according to which of ten possible landmark-scale locations they were taken in."

Even when adding the "human travel prior" [Kalogerakis et al., 2009] and taking into account image sequences respecting the recorded date and time [Crandall et al., 2009], this bias towards frequently photographed objects (and frequently shared images on Flickr) is not removed; it just introduced a regularization term that improves the results for test images sampled from the same distribution.

Though this assumption certainly helps in creating datasets, it does not fully correspond to what the user might be interested: The user's preference on what images should be geotagged and which don't have to be "geotagable retrospectively" does not need to correspond to what most people take pictures of. This conflicts can be observed in [Crandall et al., 2009]:

> "We do not report results for all 100 cities because most of the lower-ranked cities do not have a sufficient number of Flickr photos at their less salient landmark locations."

These approaches were not designed to solve the geotagging problems of individual users – the statement essentially means: "If you try to geotag your images with our

[54] In fact, it seems like these systems are best suited to help in automatically annotating images with very sparse metadata at image sharing websites: As bulk-uploading is nowadays very convenient, but annotating images is time consuming, at all major image sharing sites there is fairly high number of images without good annotations. This does not mean that in such cases the user uploading the image would not benefit from sharing the images – as sites like Flickr or Imageshack/yFrog are frequently used to interact with certain people, e.g., a social network of contacts and the images are uploaded to create a link which can be forwarded or will automatically get sent to contacts. However, these images cannot be found easily by people not directly connected to the uploader, but also interested in the image, e.g., if the uploader A has a very nice picture of the Eiffel tower and the user B performing a themed search for illustrating an article on Paris. By automatically geotagging the images in an approach like in [Hays and Efros, 2008, Kalogerakis et al., 2009, Crandall et al., 2009] such tasks can be eased for some popular image subjects without any additional manual work while they will not be applicable to most less popular subjects. In contrast, for retrospective geotagging, any approach should work similarly well for *all* content and some interaction with the user is acceptable, as the user otherwise will fall back to manually tagging the images anyway.

current approach, we can't help you unless the subject of your images are very, very common." And this is certainly something, what we didn't want to accept, because from a user perspective, it can even be quite the opposite: For well-known places, the user will probably remember their name and can easily use geocoding tools to get their geolocations. It's the not-so-known places for which it is hard to find out their location afterwards and where the user benefits most from support that a content-based system provides for retrospective geotagging.

Starting Point for Our Approach

The prior to be added to support the user best should be: *Where has the user been?* And given the user's task, we consider asking the user for the approximate region and considering images the user previously tagged a valid option. Therefore we can present the user only results that are close where the user was – and if we cannot deliver helpful images as there might be no geotagged images from that particular area, the time the user invested was not completely in vain as placing an image manually on a map can start now with the preselected area.[55]

Therefore instead of trying to solve Equation (10.12) directly to assign image coordinates automatically, our approach takes the user inside the loop by providing more information –in particular the area in which the user has been– and present a list of results together with the visual distance between the image that the user provided and images with known geotags:

$$location : \mathcal{I} \times \mathcal{P}(\mathfrak{G}) \rightarrow \mathbb{R} \times \mathcal{P}(\mathcal{I}_{\mathcal{G}}) \tag{10.13}$$

As all returned images have known locations, Equation (10.10) can be used again to return these locations and continue with these results in a similar way as in geocoding in Equation (10.11), that is, let the user solve the remaining ambiguity by selecting the most appropriate image or setting a mark on a map. What is different from geocoding is, that for this approach the user can provide only an image – limiting the area may help, but in worst case the entire space of valid coordinates $\mathfrak{G}$ can be used. With geocoding, the user would need to provide at least some textual information in addition to the image.[56]

[55]Considering retrospective geotagging as a machine learning problem and sampling the training images from Flickr as it was done in [Kalogerakis et al., 2009, Crandall et al., 2009] imposes that the images for which it will be used follow the same distribution – so as both references state, have peak location distribution in the North America and Europa. From a user perspective for a particular task, the distribution will frequently be skewed: Any person living in Asia or Africa might have a peak on the home continent and even people who travel both, North America and Europe may not travel both in a single event / session of tagging images. If considering family pictures, those will be located mainly where family members live – and not where most people using Flickr take pictures of landmarks. This is not just a prior on human travel behavior, but a clear violation of the underlying assumption for machine learning, that the training dataset and the usage of the system will sample from the same distribution and that individual samples are independently and identically distributed (IID, cf. also similar problem in relevance feedback techniques on page 177): Individual samples from a single user are not independent.

[56]Just for clarity: In geocoding, the image itself is not used. But without the existence of an image, there would be no target to geotag and there would be no need or use for geocoding for retrospective geotagging.

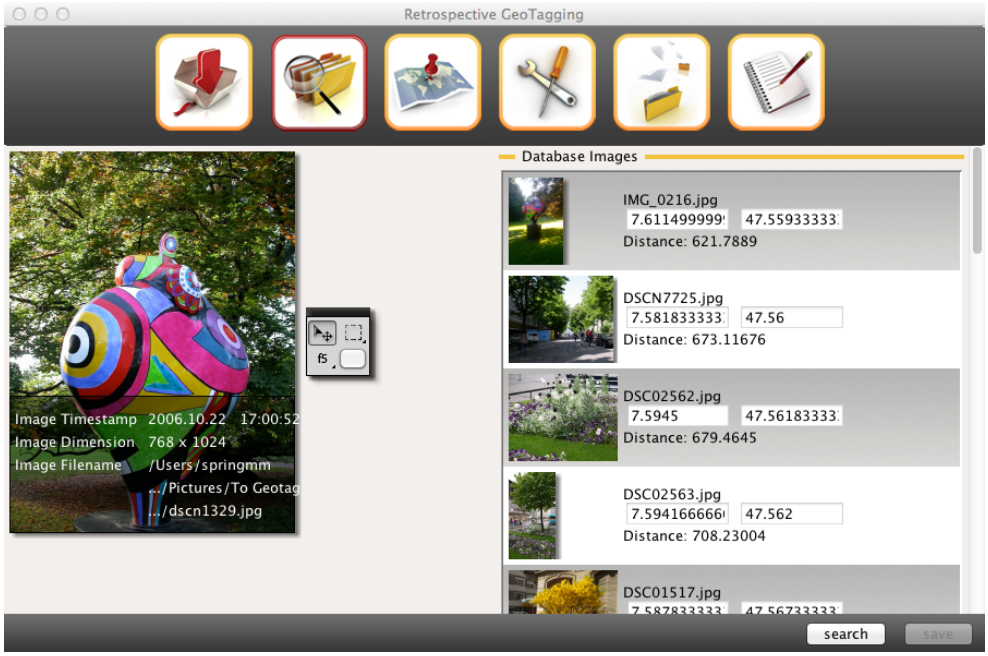


Figure 10.46: Enlarged view of query image and list of search results using global color moments as the feature.

10.4.3 Implementation of Query Formulation and Execution for Retrospective Geotagging

The results delivered by Equation (10.13) form a list ranked on visual distance based on perceptual features and a distance measure – to implement it we can reuse the search primitives from Section 10.1 and therefore also building blocks that we already introduced. Novel parts have been added where needed to support user in the task. In the context of this chapter, we will focus the discussion on the parts of Query Formulation and Execution.

Tagging of Series

One general observation in management of personal image collections is, that frequently people collect many images at an event.[57] As retrospective geotagging is a task that is performed *after* the event happened, it is very common that the user will post-process not a just a single image, but all (or at least: all good) images of the event in a single session.

As a consequence, images that have been geotagged by the user have to be added to the set of images with known geocoordinates $\mathcal{I}_G$, such that similar images from the same series can already use them. The user interface of the system that will be mentioned in more detail in Chapter 14.2.3 provides some support to either use geocoding as an alternative or also copy or interpolate tags directly from previously tagged images.

[57] Cf. [Mulhem and Lim, 2003] and it is very visible in how common applications structure the content as mentioned in Chapter 4.1.2.

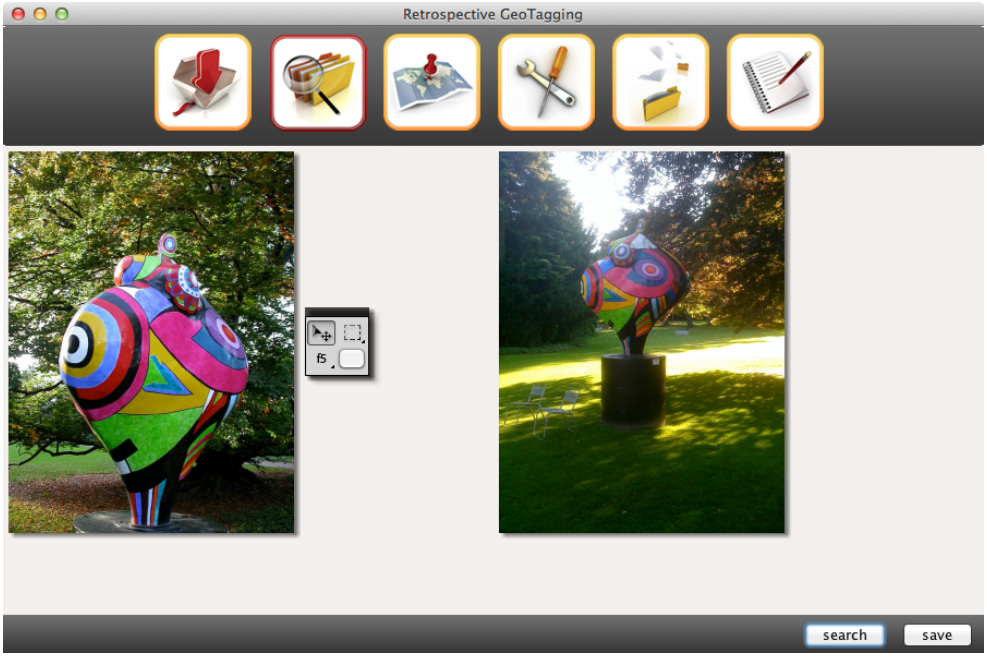


Figure 10.47: Enlarged view of both, the query image and and the best match from the search results using global color moments as the feature.

Simple Content-based Search

When the user has selected an image of the series, a search with this image can be performed. Based on the implementation of [Wigger, 2007], color and Gabor texture moments [Stricker and Orengo, 1995] are available for the following static regions:

1. *Global*: A single feature vector corresponds to the information of the entire image (cf. Chapter 5.1.4).

2. *Fuzzy5*: A central image region gradually converges into 4 regions towards the image corners as proposed in [Stricker and Dimai, 1996, Stricker and Dimai, 1997] (cf. Chapter 5.1.3 and Figure 5.5(e)–(f) for illustrations).

3. *3 × 3 Non-Overlapping Rectangles*: A regular split of the image into nine regions (similar to the 16 / 64 regions in Figure 5.5(a)–(d)).

4. *3 × 3 Overlapping Rectangles*: A regular split of the image into nine regions were each region covers part of its neighboring regions (similar to the four overlapping regions in Figure 5.6(a)–(d)).

These features are mainly intended for simple searches in which the image as a whole is closely matched: A similar image in the database was shot from a similar position. Figure 10.46 shows the query image on the left and the results of a nearest neighbor search on the right using global color moments and the L_1 distance (a.k.a. Manhattan distance). Figure 10.47 shows the closest matched image enlarged. This image shows the same real-world object as the query image. The two images are taken from a slightly

Figure 10.48: Map of Basel showing the locations of the search results using global color moments as the feature.

different viewpoint, but close enough that the difference in geolocation caused by the shift of viewpoint is less than the inaccuracy that common GPS measurements generate. As the user is in control of the process, she could also correct the position manually if she wants locations that are even more accurate.

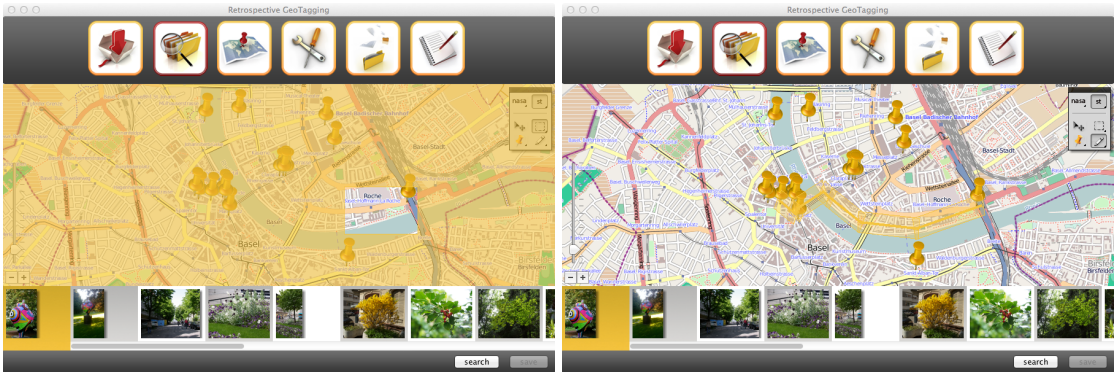
User-Selectable Geographic Areas

Geocoordinates presented as numeric values are usually not very helpful to user. Therefore the search results can also be presented on a map. Figure 10.48 shows a map zoomed to the inner area of Basel with the same results as in Figure 10.46. Yellow pins are used to indicate the position of an image, the images of all results are shown at the bottom of the screen.

In the map view, the user may select certain areas to restrict search results to images with a known location within the area bounds. For instance, Figure 10.49(a) shows an selected area covering the Tinguely Museum and Solitude park. Using Equation (10.10) to determine the location of an image, Equation (10.14) can be used to filter out images that are outside a selected area. Figure 10.50 shows the new search results when this filter predicate has been applied.

$$P_{inArea}(img) = \begin{cases} \textbf{true} & \text{if selected area contains } location(img) \\ \textbf{false} & \text{otherwise} \end{cases} \tag{10.14}$$

With this feature, the user can adjust the matching tolerance by specifying which images are relevant because they are in the area that the user has passed while taking the pictures. For even more fine-grained control, the user can also mark a path instead of a



(a) Bounding Box Selection (b) Path Selection

Figure 10.49: Selecting areas on the map can be performed in two ways: (a) shows the simple selection of a single bounding box; (b) shows the selection of a path.

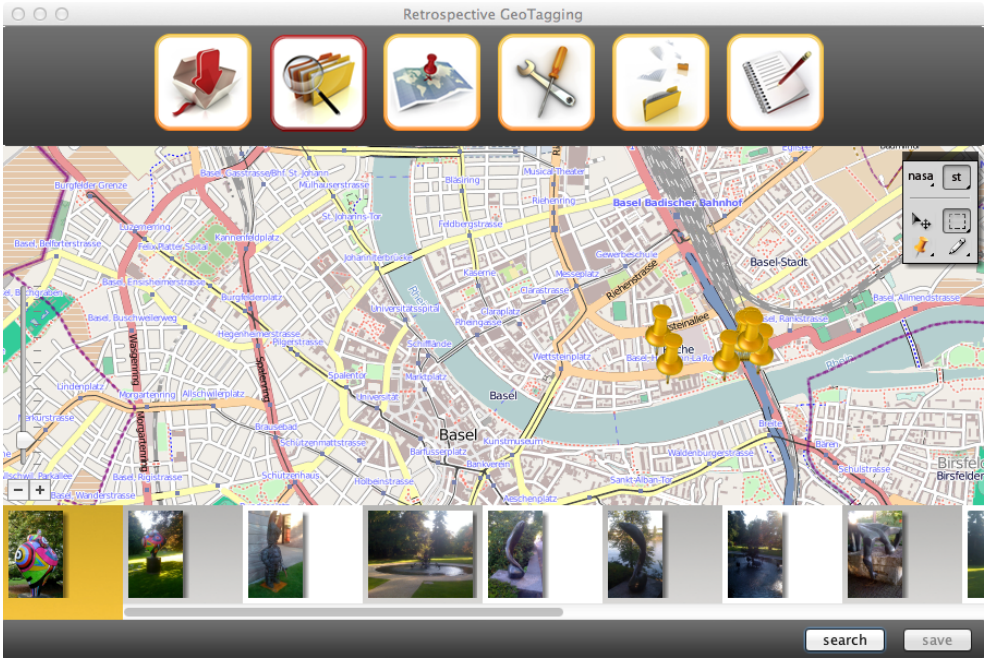


Figure 10.50: Search Results within the selected area on the map from Figure 10.49(a).

single bounding box as shown in Figure 10.49(b). But it does not only add more control, it also enables to track on the map where the user traveled and use this information for all images of the series. Such a usage is similar to how geotagging is performed using a log captured by a GPS tracking device as shown in Figure 10.45, with the major difference that here the image content will be used to link location to the path rather than timestamps – the latter are not available when the user just draws a path, while the image content can be compared to the content of images within a (configurable) area along the drawn path.

By restricting the search the subset of the dataset that is located in the user-selected area on the map, two highly desirable properties are achieved:

1. *Less "false positives":* If no constraint on the location was imposed and the image contained an object that exists in many places, e.g., a telephone booth that might look similar anywhere in the same country, the most visually similar pictures may have been taken at very different place – therefore making the geotag of the visually similar match a very inaccurate help in determining the geolocation of the query image. By constraining the reference images to the subset with a geotag close to where the user has actually been, it is much more likely that visually similar images were taken of the same scene containing the same physical objects.

   This also reduces the time that the user may spent in situations, where content-based search cannot help the user since the dataset simply does not contain any similar image taken at the same location. In such a case when the user wouldn't restrict the area on the map, the system will present the most similar images even if they can never help the user and the user has to browse through the result list. The user will probably reformulate the query several times until realizing that the only successful strategy to geotag the image is, to tag the image manually. And this manual tagging will frequently be performed by interacting with the map through placing a virtual pin at the correct location – an interaction step which doesn't take that much additional time if the user already identified the approximate location. Of course, the latter is only possible, if the user remembers the location and is able to identify it on the map; without enough knowledge on either the side of the user or on the side of the system, retrospective geotagging tasks remain unsolvable.

2. *Faster search results:* When using adequate index support, restricting the area on the map will result in less candidates for which the distance between their high-dimensional features and the features of the query image have to be computed. Therefore the more time the user invests in formulating the query to express as precisely as possible what are desired results (in other words: fine-tunes the matching tolerance), the faster the system can return results.[58]

In a similar, although less graphical way we allow the user to define a time range for the matching by specifying either the earliest time, latest time, or both for images to be used as references to derive the geolocation of query images. This is most useful in cases where: (a) the objects in the query image do exist only since a fairly recent point in time (or do no longer exist at the location), or (b) the user knowns the dataset very well as it is constructed from previous events and trips that the user did and therefore knows for instance, that she has visited the same location before; therefore it makes sense to restrict the search only to images of the time period of the previous visit.

A particular instance of the second case occurs when a group of people take pictures at the same event, e.g., go together on vacation and later want to merge the collections

[58]Notice, that this desirable property is not always given in other applications for retrieving information. For instance, for text-retrieval systems based on boolean or vector-space retrieval models, the more keywords are added to the query, the longer the execution of the query takes. However, the execution of a single textual query in such a model can usually still be processed much faster than common image similarity search in high-dimensional feature spaces since all documents that do not contain any of the requested keywords can be safely ignored; for image similarity searches, unless the features are designed to mimic that aspect of text retrieval, the individual values in the feature vector frequently do not possess a semantics to exclude any image in ranking that easily.

of pictures that people inside this group have taken. If some members of the group recorded the geocoordinates for the images but other didn't, those that have geotags can be used for retrospective geotagging of the ones without tags. A reasonable time range is defined by the event's duration – which can also be derived for convenience from the date of creation of the first and the last non-geotagged image of the event. In this case, the time range helps to separate the images of the event from other unrelated images in the dataset; in particular when combined with selecting areas on the map. This allows the user to keep a single personal dataset with all images without negative impact on retrieval quality or retrieval time as the unrelated images are filtered out efficiently and effectively, thus do not appear as false positives and do not consume additional time for similarity search.

Regions of Interest in Query Images

Similar desirable results as with letting the user define an area of interest on the map can be achieved by letting the user define regions of interest (ROI) inside the query image. Such ROI allow to separate the areas that should be matched from irrelevant or even unwanted areas (cf. Chapter 2.3.3 on *empty, unknown, irrelevant, and unwanted areas*). Such areas that shouldn't be matched for the purpose of retrospective geotagging are all areas that reduce the reliability of finding other images of the same location based on similarity; therefore may include moving objects or objects that change their visual appearance easily over time. Examples of moving object can be vehicles, people or animals passing by and an example of something that changes the appearance over time is the sky.

However, this rule is not without exceptions: depending on the particular image and location, even such objects might be indicators for a location. For instance, most animals in a zoo are not allowed to move over the entire area, but have to stay in their compound – therefore providing in combination with the image context a reasonable indicator of where inside one particular zoo the image was taken. The same can be the case for vehicles which or people who frequently appear in at a certain place.[59] And last, but not least, for identifying that an image belongs to an event that took place at a particular location and out of which some images have already be geotagged, any image content that occurs several times within the event can help in identifying the geolocation – in particular when images of a series are incrementally processed. Therefore we believe that the user is best served by being allowed to select regions of interest based on her own knowledge and experience.

[59] An extreme example of moving objects can even be stars detected at the clear sky during night – in combination with the date and time of image creation, this might actually be usable to identify the location where the image was taken. However, taking good pictures of the stars during night is a non-trivial task. Therefore a user will likely remember where she spent the effort to take the picture, which may have required to mount a tripod, tweak camera parameters for low noise, and capture a long-time exposure. But if the user remembers the location well, the need for a system supporting retrospective geotagging is reduced. Furthermore, in this extreme setting, it is fairly unlike that some other image outside the series will show the same content including a very similar star constellation. For using stars to determine a geolocation, it is very likely that specialized systems will be needed.

Figure 10.51: Bounding box to select a region of interest and search results using SIFT.

A simple geometric structure to mark regions of interest are bounding boxes (BBox), which can easily be selected with a common mouse as an input device. Figure 10.51 shows the interface to formulate a query that uses a user-selectable bounding box to define a region of interest.

In theory, any feature can be used with a region of interest as the contents of the bounding box is not different from any other image. However, in practice, the quality of retrieval results for features that are used with a single global region or –even worse– static subregions can degrade significantly when the selected region of interest no longer correspond visually to the style of the images in the database. For instance, if the images in the database mainly consist of images that capture large sceneries or landscapes but the bounding box corresponds to the close-up of some detail, the extracted features will no longer correlate nicely. As an extreme example: If color moments are used as a feature and extracted from five fuzzy regions, in many images of landscapes, the two regions in the upper half of the image will account to a great extent for the color of the sky – simply because that is the content that will be found in the top 20-60% of many landscape images. If the user selects now a bounding box that excludes the sky as it is rarely a good indicator of the location the color moments of two of the five regions will differ so much, that they could hide any good result of the remaining three regions.

Such effects can be compensated by explicitly enforcing a “contains” semantic of matching: The information found in the query image must be present in the reference image – but not the other way round. This could still be fairly easy to achieve if the camera was forced to take pictures only in standardized positions such that the regions could simply be ignored – for instance, by adjusting setting the weights of static regions outside the region of interest to zero similar to the treatment of unknown areas in sketches in Chapter 10.3.4. As there are no standardized positions in general for images

being geotagged, a remaining heuristic could compare the region of interest with subimages of the reference images in the dataset similar to the strategy of extracting several image regions to compensate for translation in user-drawn sketches on page 243. However, in comparison to the overall setup of the task in Chapter 10.3 which focused on Known Image Search, the user in retrospective geotagging is usually not aware of the reference images in the dataset and would usually not optimize the bounding box to correspond to these images. Therefore difference in scale and translation are frequently much more severe. The logical consequence would be to use more and smaller subregions of an image and consider also very strong misplacements – which essentially describes the approach that uses a different, extreme kind of regions: Keypoints (or also referred to as Salient Points or Salient Regions, cf. Chapter 5.1.2). Potential alternative would have been to use segmented regions as described in Chapter 5.1.1 – however, dealing with keypoints is usually less prone to errors when individual points are occluded or missing due to a changed viewpoint.

Usage of Keypoints in Retrospective Geotagging

Keypoints are points in the image that can be detected rather reliably even if the image has been transformed and for which a descriptor is derived from a small region around the point. Figure 10.52(a) and (b) show the keypoints that have been detected on two different images taken at two different dates with two different cameras from a slightly different viewpoint covering a different area of the same object. We use a Differences of Gaussian (DoG) keypoint detector that searches for extrema in scale-space [Lowe, 2004] and the Scale-invariant Feature Transform (SIFT) as keypoint descriptor proposed in the same reference. The descriptors in SIFT analyze the homogeneity of the gradients of pixels around the detected salient point to derive a 128-dimensional histogram that provides a local view on the image's texture. The direction of the arrows in Figure 10.52(a) and (b) indicates the main gradient orientation found in the 128-dimensional histogram while the length of the arrow indicates at which scale this keypoint was found and therefore also corresponds to the area of pixels around the keypoint that was used to generate the gradient orientation histogram[60]. The same information is displayed with the circles for the matches in Figure 10.52(c): The size of the circle around the keypoint corresponds to the scale at which the keypoint was localized, the radial line from the circle center corresponds to the main gradient orientation. SIFT was designed in order to cope well with changes in scale and the gradient orientation histogram is normalized in a way, that when comparing two descriptors, the distance is independent of the main orientation – therefore also being invariant to rotations of the object in the picture as long as this rotation is mainly occurring in the plane orthogonal to the camera direction taking the picture. Tilting of that plane can and will affect the retrieval performance, usually starting to slowly deteriorate at angles above 15 degrees and remain rather reliable up to changes in viewpoint of 40 to 60 degrees[Lowe, 2004, Mikolajczyk and Schmid, 2005].

Images in the dataset that we will use in Section 10.4.4 were scaled to at most 1280 pixels on the longer side while preserving aspect ratio. The aspect ratio of the used cameras were either 3:2 or 4:3, which results in either 851 or 960 pixels for the shorter side.

[60] Although the length of the arrow is not equivalent to the pixel area in absolute terms.

(a) Keypoints DSCN7226 (b) Keypoints DSC01842

(c) Matching

Figure 10.52: Extracted and matched keypoints in two images of the clock on the wall of the Rathaus in Basel: 1175 keypoints were detected in the query image (a), 2308 were detected in the reference image of the geotagged dataset (b). The seven matches with minimal distance on the L_1 norm applied to the SIFT key point descriptors are shown in (c).

As the number of detected keypoints when using DoG does not only depend on the number of pixels in the image, but also on the actual content of the image, between 151 and 5'010 keypoints have been detected per image. The average number of keypoints was 1'482 with only some very dark images taken during night having less than 400 keypoints. For each detected keypoint the SIFT descriptor was extracted; in total, a little more than 2 million keypoint descriptors have been extracted from the dataset of 1'408 images. These are similar values to the reported 40'000 keypoints detected in 32 images in [Lowe, 2004] (average: 1'250 keypoints per image).

Figure 10.52(c) shows the seven best matches based on the L_1 distance between the SIFT descriptors. When looking carefully at the matches, one can see that two of the keypoints are located very close to each other just below the clock in the left image DSCN7226, which both get matched to the same keypoint in the right image DSC01842. This is possible if no constraint is applied and for any keypoint in the left image, the most similar keypoint in the right image is identified. Furthermore one can see that one matching is incorrect: One keypoint located on the curtain in a window in the left of DSCN7226 is matched to a keypoint on the flag to the right in DSC01842. Both keypoints are located on objects, that cannot be matched correctly due to the different region of the building contained in the images (the window from DSCN7226 is outside the region shown in DSC01842) and the flags shown in DSC01842 were not mounted to the building when DSCN7226 was shot. This example shows that not only the choice of particular keypoint detector and descriptor is of great importance when using keypoints[61], but also the strategy to match keypoints and generate rankings of images.

Matching of Keypoints for Retrospective Geotagging

SIFT has originally been proposed for object recognition [Lowe, 1999], in particular to detect previously seen objects when presented a new image as it is a common task in robot vision. For the task of object detection, it is most important to identify some highly reliable matches which allow to determine the placement and orientation of the object inside the picture. [Lowe, 2004, Mikolajczyk and Schmid, 2005] have found that as a criterion for such matches, a threshold on the ratio between the closest and the second closest match performs better as an indicator than a threshold on the absolute distance to the closest match.

For both matching criteria, nearest neighbor ratio and distance threshold, Pseudo Code Listing 10.3 presented on page 205 can be re-used when instead of whole images $\mathfrak{Q}$ and *Docs* just individual keypoints are fed to the function. Let κ_{DoG} be the function to detect the keypoints in an image using the Differences of Gaussian (DoG) detector and Φ_{SIFT} be the function to extract the 128-dimensional SIFT descriptor for any

[61] A review of keypoint descriptors can be found in [Mikolajczyk and Schmid, 2005]. SIFT is probably the most frequently used descriptor as it has shown overall solid performance, for instance also in the related work [Zhang and Kosecka, 2006, Crandall et al., 2009, Snavely et al., 2006] and the VisualRank [Jing and Baluja, 2008] used for Google's Search by Image feature, while [Hays and Efros, 2008, Kalogerakis et al., 2009] relies on a mix of various features which use mostly global or static regions and a 512-dimensional texton histogram. A much more exhaustive list of features that can be used as descriptors for detected keypoints has been mentioned towards the end of each section corresponding to the feature type *color*, *texture*, and *shape* of Chapter 5.2.

DISTANCEMATCHCOUNTINGRATIO($\mathfrak{Q}, \mathfrak{R}, ratioThreshold$)

```
1  count ← 0
2  For each q ∈ κ_DoG(𝔔)
3       do
4          Res ← NEARESTNEIGHBORSEARCH(q, κ_DoG(𝔑), Φ_SIFT, Δ_L1, 2)
5          (d_best, r_best) ← Res[1]
6          (d_sec, r_sec) ← Res[2]
7          if d_best / d_sec < ratioThreshold
8            then count ← count + 1
9
10 return |κ_DoG(𝔔)| / count
```

Pseudo Code Listing 10.11: Computation of the distance between two images using the match counting strategy and a threshold on the ratio between the best and the second best match.

DISTANCEMATCHCOUNTINGDISTTHRESHOLD($\mathfrak{Q}, \mathfrak{R}, distanceThreshold$)

```
1  count ← 0
2  For each q ∈ κ_DoG(𝔔)
3       do
4          Res ← RANGESEARCH(q, κ_DoG(𝔑), Φ_SIFT, Δ_L1, distanceThreshold)
5          if |Res| > 0
6            then count ← count + 1
7
8  return |κ_DoG(𝔔)| / count
```

Pseudo Code Listing 10.12: Computation of the distance between two images using the match counting strategy and a simple threshold on the distance of the best match.

passed keypoint or set of keypoints. Then NEARESTNEIGHBORSEARCH($\kappa(\mathfrak{Q})[1], \kappa(\mathfrak{R} \in Docs), \Phi_{SIFT}, \Delta_{L_1}, 1$) will return the closest matching reference keypoint to the first keypoint in the query image and its distance. With $k = 2$ the distance of the closest and second closest keypoint is returned, making it straight forward to compute the ratio.

Although being able to compute some reliable keypoints is already a good starting point, it is not yet sufficient to provide a ranking for the reference image as a whole. Such a ranking of the image as a whole it will be needed for using keypoint matching in retrospective geotagging. Two very simple strategies, *Match Counting* and variants of *Sum of Keypoint Distance*, can be used to aggregate an overall distance from the individual keypoint matches.

Ranking Result Images using Match Counting

For ranking the the result images based on a match count, a criterion has to be defined to identify which keypoints of the query image should be considered matched to a keypoint in the reference image. Pseudo Code Listing 10.11 and 10.12 show the pseudo code for the two different matching criteria, nearest neighbor ratio and distance threshold. The value that is returned as the distance value in both cases takes the count of keypoints detected in the query image and devides it by the count of found matches. This normalizes the distance score to enable interpretations of the value as "every i-th keypoint in the query image is matched" with i being the value returned by DISTANCEMATCHCOUNTINGRATIO and DISTANCEMATCHCOUNTINGDISTTHRESHOLD.

By considering all the keypoints in the query image in the outer loop, but ignoring all the keypoints inside the reference image, a semantic is expressed that corresponds to "the query image is contained in the reference image". The critical part in using this strategy successfully is finding a good threshold that separates the matches that correspond to the users' perception of which images are similar / show the same scene from a similar viewpoint. Since only the number of matches is counted and therefore the individual distances for the matches are not further evaluated, if the threshold is very relaxed, all reference images will achieve the same score: As all query keypoint will be considered matched, the number of keypoints in the query image divided by the number of matches that passed the threshold will approach 1. If on the other hand the threshold is too tight, hardly any query keypoint will get matched and the few that will be matched, will not necessarily be the ones that are best indicators of the precise geolocation. For instance, a logo or symbol that can be found throughout the entire area might be quite well detectable by the DoG keypoint detector and as those are standardized and usually placed on flat surface, the matching may turn out to be achieve a good distance.[62]

AVERAGEKEYPOINTDISTANCE$(\mathfrak{Q}, \mathfrak{R})$

1 $sum \leftarrow 0$
2 For each $q \in \kappa_{DoG}(\mathfrak{Q})$
3 **do**
4 $(d_{best}, r_{best}) \leftarrow$ NEARESTNEIGHBORSEARCH$(q, \kappa_{DoG}(\mathfrak{R}), \Phi_{SIFT}, \Delta_{L_1}, 1)$
5 $sum \leftarrow sum + d_{best}$
6 **return** $\frac{sum}{|\kappa_{DoG}(\mathfrak{Q})|}$

Pseudo Code Listing 10.13: Computation of the distance between two images as the average distance of the matching of keypoints.

[62]The problem is less prominent when using the nearest neighbor ratio; however, as we will see later, the nearest neighbor ratio leads to other problems.

AVERAGEBESTMATCHEDKEYPOINTDISTANCE($\mathfrak{Q}, \mathfrak{R}, Docs, minK, bestPercentage$)

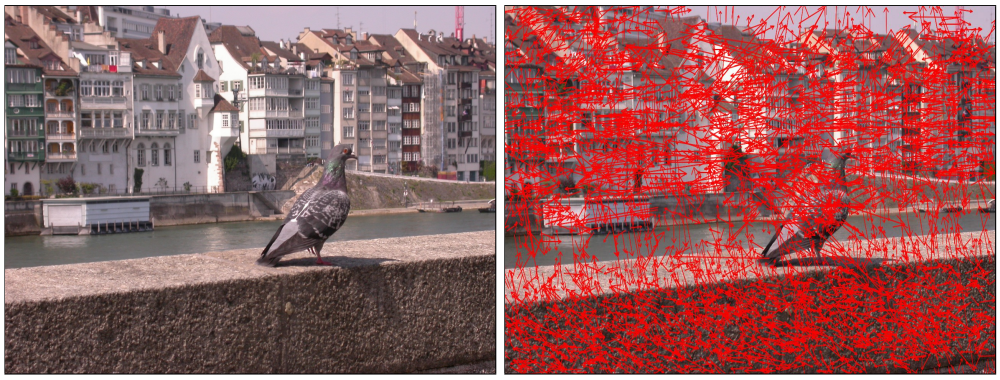
```
1  bestK ← max(minK, |κ_DoG(𝔔)| × bestPercentage)
2  bestMatched ← LIST of size bestK
3  allKeypoints ← ∪_{𝔗∈Docs}(κ_DoG(𝔗))
4  For each q ∈ κ_DoG(𝔔)
5       do
6          (dist, q) ← NEARESTNEIGHBORSEARCH(q, allKeypoints, Φ_SIFT, Δ_L1, 1)
7          if length(bestMatched) < bestK
8            then
9                 INSERT-SORTED(bestMatched, length(bestMatched) + 1, dist, q)
10           else
11                (dist_k, keypoint_k) ← bestMatched[bestK]
12                if dist < dist_k
13                  then
14                       INSERT-SORTED(bestMatched, bestK, dist, q)
15
16 sum ← 0
17 For each q ∈ bestMatched
18      do
19         (d_best, r_best) ← NEARESTNEIGHBORSEARCH(q, κ_DoG(𝔑), Φ_SIFT, Δ_L1, 1)
20         sum ← sum + d_best
21 return sum / bestK
```

Pseudo Code Listing 10.14: Computation of the distance between two images as the average distance of the matching of the best-matched keypoints, only.

Ranking Result Images based on the Sum of (Best Matched) Keypoint Matches

In theory, one could also simply sum up the distance for all the keypoints found in the query image. Averaging this through dividing the sum by the number of keypoints in the query image as in Pseudo Code Listing 10.13 would normalize w.r.t. the query image, thus making the distance more comparable between several searches. In practice, any query image usually contains a significant number of keypoints for which there will simply be no good match in any other image in the dataset even if they show the same objects as keypoint detectors return different keypoints depending on the image resolution and content; minor changes in the image might result in one keypoint no longer being detected while the majority of keypoints get.

There is always a nearest neighbor in matching a single keypoint with images with at least one keypoint. However, the distance to the nearest neighbor of a well matched keypoint is low and the distance of a single keypoint for which there is no good match is fairly large – which leads to the situation that an image would need many matches with very low score to compensate for a single keypoint with no good match.



(a) DSCN7671 (b) Keypoints DSCN7671

Figure 10.53: Original image and extracted keypoints of the image DSCN7671: In this image from the reference dataset, 3816 keypoints were detected and the descriptors show great variability – thus providing a low overall sum of distances for any query image when a simple approach like Pseudo Code Listing 10.13 is used.

To better illustrate the problem with the Pseudo Code Listing 10.13 that computes the simple average over the distances for all query keypoints, we will discuss the impact for some particular images: If we consider for instance the images in Figure 10.52(a) and (b), the latter actually corresponds to the smaller region of the real world object but the DoG detector found almost twice as many keypoints in it. The main reason for this is that fairly homogenous areas like the clear blue sky contain hardly any keypoints – and (b) contains a much smaller portion of sky, but an additional row of windows in the bottom of the picture where many keypoints get detected. Image DSCN7671 shown in Figure 10.53 does not contain any part of the Rathaus of Basel shown in the images of Figure 10.52. The DoG keypoint detector identifies 3'816 keypoints –more than three times the number of keypoints found in DSCN7226 that was taken with the same camera and processed at the same image resolution– and the SIFT descriptors for these keypoints show great variability. The latter means that for any query keypoint in any image, there is not necessarily a perfect match, but there will be with high probability some keypoint in DSCN7671 that is not as extremely far off compared to other images that do not contain the objects in the query image and are more homogenous. And this leads to the undesirable situation that query images for which the DoG detector is *less* able to detect keypoints deliver frequently better results:

- When DSCN7226 (1'175 keypoints) is matched against DSC01842, the average distance per query keypoint is 2.187. The average distance when matched against the unrelated image DSCN7671 is 2.273. Although the difference between the average distances is fairly small, at least the ranking is what the user would expect: Images that show the same object achieve the smaller average distance per keypoint.

- When the query image is exchanged against DSC01842 (2'308 keypoints) and is matched against DSCN7226, the average distance per query keypoint is 2.307. When DSC01842 gets matched against the unrelated image DSCN7671, the av-

erage distance is 2.235. As this distance is lower than the distance between the related images, the unrelated image would get ranked higher.[63]

Therefore it is advisable to focus just on those keypoints for which there is at least one good matching keypoint in at least one image in the dataset – thus rank the images mainly on the best matchable keypoints in one particular image instead on how bad the worst matches are. The latter gives preference to images that contain many and very diverse keypoints.

A strategy to determine the best matched keypoints can be added by first computing for every keypoint the best matching keypoints from the entire dataset, not just a single image. From this first matching phase, we can select those k keypoints in the query image that achieved the lowest distances. Alternatively, also a certain percentage of all keypoints in the query image can be selected or a combination of percentage and minimal number k as done in Line 1 of Pseudo Code Listing 10.14. Notice that in Pseudo Code Listing 10.14 the blue text color is used to highlight the changes in Line 1 – 17 compared to Pseudo Code Listing 10.13. Notice further that the added Line 4 – 14 are near identical to the corresponding code in Pseudo Code Listing 10.3 on page 205 and reuse the function INSERT-SORTED.

When using only the best matched 10% of the query keypoints, the problem matching Figure 10.53 does no longer occur. The same is also when applying the Match Counting strategy when a distance threshold of 1.5 is used; this threshold is significantly smaller than the average distance found for all keypoint matches in the problematic examples.

AVERAGEINVERTEDSQUAREDKEYPOINTDISTANCE($\mathfrak{Q}, \mathfrak{R}$)

1 $sum \leftarrow 0$
2 For each $q \in \kappa_{DoG}(\mathfrak{Q})$
3 **do**
4 $(d_{best}, r_{best}) \leftarrow$ NEARESTNEIGHBORSEARCH$(q, \kappa_{DoG}(\mathfrak{R}), \Phi_{SIFT}, \Delta_{L_1}, 1)$
5 $sum \leftarrow sum + \frac{1}{(d_{best})^2}$
6 **return** $\frac{1}{sum}$

Pseudo Code Listing 10.15: Computation of the distance between two images by internally taking the inverted squared distance of individual keypoint matches.

Another approach that uses the same numeric trick as in Pseudo Code Listing 10.8 on page 219 could also resolve the issue without the need of any tuning parameter: Computing the sum of inverted squared distance of the keypoint matches. This way, most emphasize is given to the keypoints in the query image that are best matched within the compared reference image. Pseudo Code Listing 10.15 shows the highlighted changes compared to the simple average computation. The downside of this approach

[63] Switching to a different commonly used distance function for comparing the SIFT descriptors, e.g., L_2 did not resolve the issue.

is, that the values used as distance for the ranking of images cannot get interpreted that easily anymore.

Restricting the Amount of Keypoint Rotations

The visualizations of keypoints and matching in Figure 10.52 as explained on page 293 show the main gradient orientation of descriptors and the scale at which a keypoint is detected. While the scale-invariance of SIFT is very important to achieve good results in retrospective geotagging as smaller regions of interest might get matched to images of significantly different size and resolution, the rotation orthogonal to the camera viewing position is only needed for objects that may turn (such as the hands of a clock) and to a lesser extend to compensate for holding the camera not completely in vertical (portrait) or horizontal (landscape) orientation.

AVERAGEINVERTEDSQUAREDKEYPOINTDISTANCE$(\mathfrak{Q}, \mathfrak{R}, rot)$

1 $sum \leftarrow 0$
2 For each $q \in \kappa_{DoG}(\mathfrak{Q})$
3 **do**
4 $o \leftarrow$ MAIN-ORIENTATION(q)
5 $rKeys \leftarrow \{r \in \kappa_{DoG}(\mathfrak{R}) | (o - rot) \leq$ MAIN-ORIENTATION$(q) \leq (o + rot)\}$
6 $(d_{best}, r_{best}) \leftarrow$ NEARESTNEIGHBORSEARCH$(q, rKeys, \Phi_{SIFT}, \Delta_{L_1}, 1)$
7 $sum \leftarrow sum + \frac{1}{d_{best}^2}$
8 **return** $\frac{1}{sum}$

Pseudo Code Listing 10.16: Computation of the distance between two images using the inverted squared distance while restricting matches only to those reference keypoints for which the change in orientation does not exceed the allowed rotation rot.

The main orientation of the gradient for the SIFT descriptor for a single keypoint may not always correspond ideally to the orientation of the overall image, but nevertheless it can be used heuristically to consider only keypoints as valid matches when the main orientation of the gradient does not differ by more than a certain amount. Assuming there is a function MAIN-ORIENTATION that returns the main orientation of a gradient and the operator $\leq$ handles arbitrary rotations of orientations correctly –in other words, it does not need special treatment when for instance, the orientation plus some rotation exceeds the range of 360°– Pseudo Code Listing 10.16 shows a variant of Pseudo Code Listing 10.13 that restricts keypoint matches to those that are within the range of the allowed rotation $\pm rot$.[64]

Much more precise control of matching and ranking is possible when the matching is only used to determine the area that potentially contains the query image and

[64] A different approach tailored to bag-of-feature models which uses the scale and orientation of the keypoints for weighting scores has been proposed as *weak geometrical consistency (WGC)* in [Jegou et al., 2008].

to approximately align the two images. The actual distance between the aligned images can then be determined using a distance measure that tolerate small deformations, for instance the Image Distortion Model (IDM, [Keysers et al., 2007]) that was used successfully in Chapter 10.2 and 10.3. We described such a processing pipeline in more detail in [Springmann and Schuldt, 2008]. However, for the purpose of retrospective geotagging, the experiments on our dataset so far did not reveal a need for such additional processing as that previously described strategies, Match Counting, Sum of Inverse Squared Distances, and computing the Average Distance only of Best-Matched Keypoints, already rank items that contain the same objects from a similar viewpoint on top – such that the user is able to quickly select the image from which the geolocations should get copied to the so-far not geotagged query image.

Problems with Nearest Neighbor Ratio

As mentioned on page 297, the nearest neighbor ratio instead of using the nearest neighbor distance directly proposed in [Lowe, 2004] for object recognition that also improved reliability of matching in [Mikolajczyk and Schmid, 2005] has its benefits, but is not free of problems when trying to apply it to retrospective geotagging. In object recognition task, in particular in the evaluation of [Lowe, 2004, pp. 103f] 32 training images, these training images were labeled and free of clutter – in other words, the training images contained nothing but the object of interest. This allowed [Lowe, 2004, p. 104] to define the ratio as being the distance to the closest keypoint divided not just by the distance to the second closest keypoint, but by the distance of the closest keypoint from a different object:

> If there are multiple training images of the same object, then we define the second-closest neighbor as being the closest neighbor that is known to come from a different object than the first, such as by only using images known to contain different objects.

When enforcing such precise annotation for retrospective geotagging, the number of available images that are geotagged and labeled for building a dataset would either shrink dramatically or require a lot of human interaction. In case the user wants to reuse her own images that she geotagged using the system, she would have to also label the objects found in the images – a task to a certain extend similar to geocoding using textual information and therefore would no longer benefit from the content-based approach that is designed for cases in which the user does not have easy access to the names and locations of objects inside the pictures.

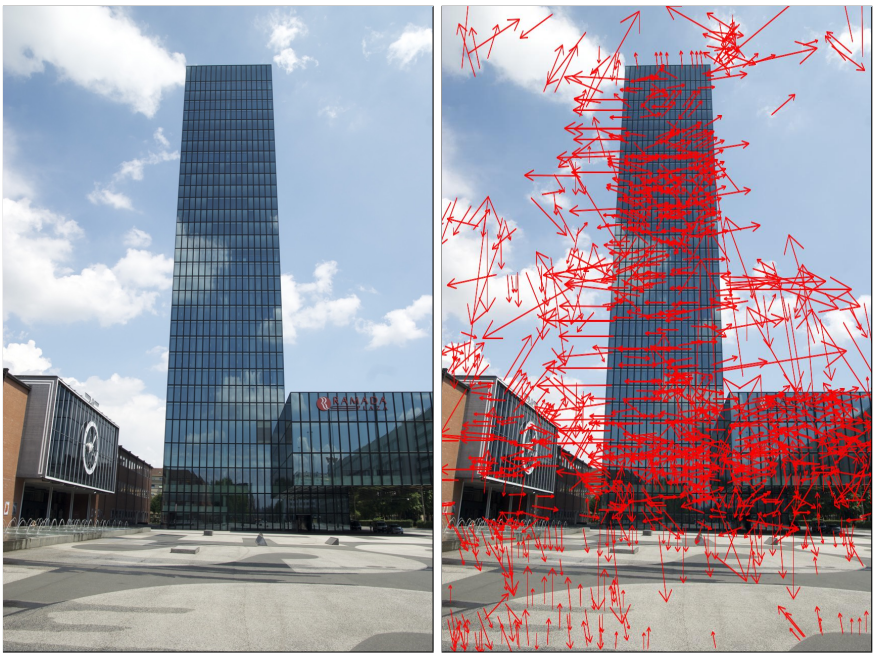
Figure 10.54 and 10.55 show what can happen, if the main object in the image is in itself very similar, therefore illustrating the problem even with just a single image: The main object in both images is the Messeturm in Basel[65]. When using the nearest neighbor ratio as a sorting criteria without ensuring that the second-best match is placed on a different object, only very few of the best matches are located on the building as shown in Figure 10.55(a): Even matches of clouds and trees that are obviously not correct are

[65]The tower at the exhibition square in Basel was until 2010 the tallest building in Switzerland. Due to its size and design, it is very easily distinguishable from other buildings.

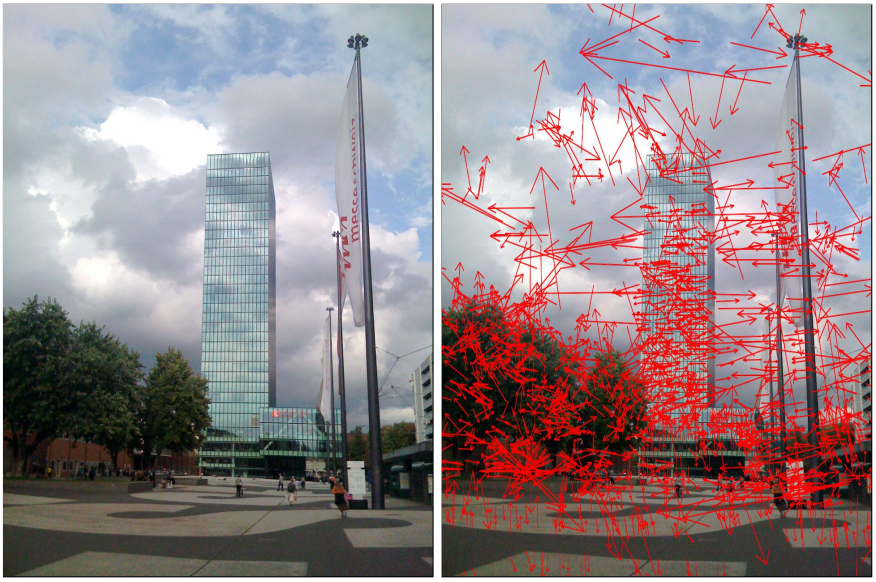
among the 10 best matches, while only one match correctly describes a correspondence between keypoints located on the buildings. In comparison, when using the distance directly as the criteria for the quality of a match as in Figure 10.55(b), 8 out of the 10 best matches identify at least correspondences on the same building.[66]

The same problem will also occur with any other object if several images of the same image are stored in the database and this is not handled as described in [Lowe, 2004, p. 104]. Without an additional object-based annotation of the images inside the dataset, this can only get approximated through the geolocation of the images, which may get combined with scanning for highly similar keypoints. The two most closely related works [Hays and Efros, 2008, Crandall et al., 2009] rely on keywords / tags associated to the images in addition to the presence of a geotag to build their datasets.

[66]Using a little less scientific language: Self-similarity in objects acts like a "stealth technique" for nearest neighbor ratio matching. Similar observation was independently reported in [Zhang and Kosecka, 2006].

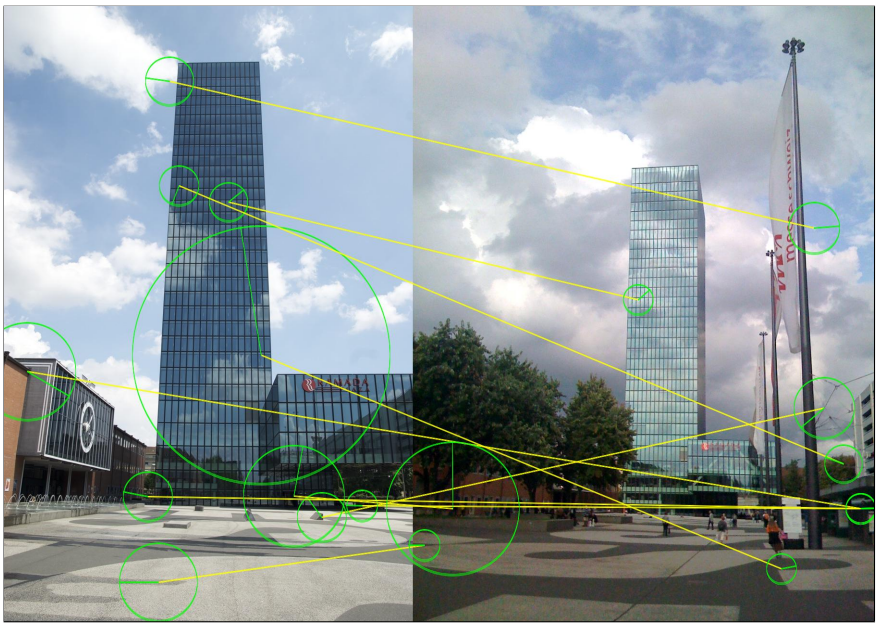


(a) DSC04948 (b) Keypoints DSC04948

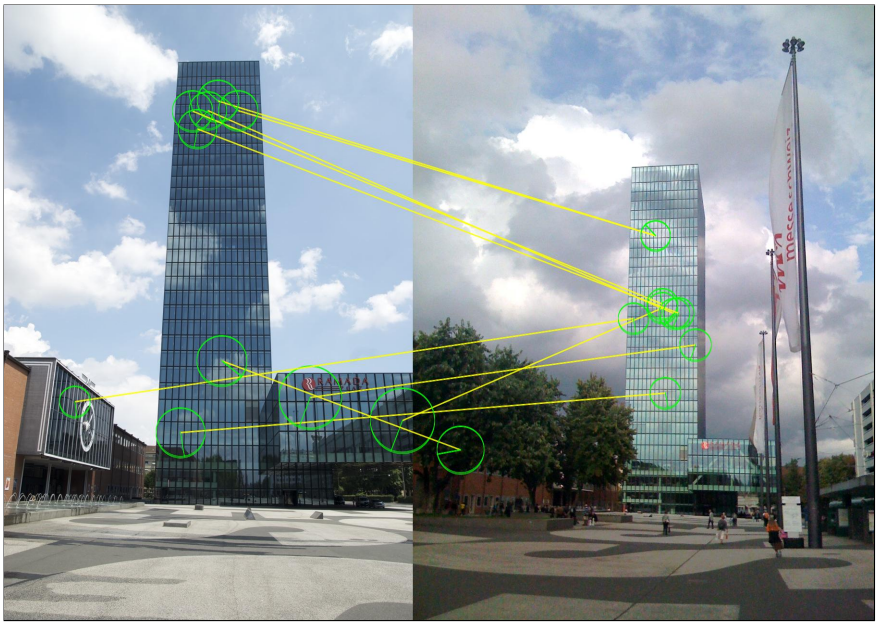


(c) IMG0020 (d) Keypoints IMG0020

Figure 10.54: Two images of the Messeturm in Basel and extracted keypoints: Many of the descriptors extracted from 1175 keypoints detected in DSC04948 (and the 1510 keypoints in IMG0020, respectively) are very similar as areas of the building's facade at the keypoint locations itself are very similar.



(a) Matching using ratio between best and second-best match



(b) Matching using the distance of best match directly

Figure 10.55: Matching of images of Messeturm using the nearest neighbor ratio and the distance directly: With best ratio in (a) only one match out of the ten matches has both participating keypoints on the building itself. Using the distance directly as in (b), eight out of ten matches with lowest distance have both keypoints located on the building.

10.4.4 Discussion of Our Approach to Retrospective Geotagging

In our approach, we enable the user to retrospectively geotag images by considering also the image content. The main contribution is not on which image features are used to perform this task, but how computing the ranked list of visually similar images can be combined and controlled with additional information the user provides, such as defining an area on a map out of which only reference images are selected and defining regions of interest to restrict which parts of the image are used to determine visual similarity.

As [Hays and Efros, 2008] observed:

> Users tend to geo-tag all of their pictures, whether they are pet dog pictures (less useful) or hiking photos (more useful). In fact, the vast majority of online images tagged with GPS coordinates and to a lesser extent those with geographic text labels are not useful for image-based geolocation.

Although we agree to the observation, we see the analysis of how useful such tagging is from a completely different perspective: [Hays and Efros, 2008] takes the perspective of a system that analyzes images from different users without any direct interaction with the user. In contrast, we consider the use case to assist a user in organizing her images better by making it easy to assign and propagate geotags. If the user wants to geotag images of her pet dog or any other image content, she shall be able to so. This cannot be done fully automatically as the dog can certainly appear at different locations – but we do not consider this inability to deliver a fully automated system for such a task as a limitation as the user cannot expect this. We consider our approach successful if it reduces the need to assign geotags to images in a labor intensive manual manner. Therefore we built our approach that it allows to incrementally geotag previously untagged images and if a similar content reoccurs, the user can reuse the tags found on these images.

To achieve this, simple color and texture features can help to quickly identify images that show a very similar scene, in particular images belonging the same event or taking place inside the same room. Integrated navigation through series of images and lookup of geographic names add complementary tools to reduce the time required for the part that can be performed mostly in a manual manner. Fine-grained control over the matching through selecting areas on the map, regions of interest in the image, the use of keypoint descriptors, and the definition of time ranges allow the user to adjust the search to correspond closely to the matching tolerance of the geotagging task.

This leads to an important difference when comparing the results of our approach to related work:

- For the "Where am I?" contest at the 10th IEEE International Conference on Computer Vision (ICCV 2005), the quality measure for automatically determining the position was the average distance in meters by which the found geocoordinates missed the ground-truth of the actual camera position recorded with a GPS tracker. Four of the five participating teams in the final round were never more than 32 meters away with their estimates, which is already within the range that can occur when using GPS for tracking a camera position. The contest was designed to be solved automatically and the results were achieved on a fairly dense

collection of images from a very restricted area, all taken with the same camera on a day with blue sky in Boston, Massachusetts. As mentioned on page 283, the ground-truth about the correct location of the images is no longer available; therefore a comparison with these approaches based on a common dataset is currently not possible.

- When geotags are collected at one point in time and compared with images taken at a time months and years later, deviations of at least 20 meters have to be expected due to the Selective Availability (SA) that is inherent to the civilian usage of GPS without correctional data (e.g., road map information as used in GPS-based road guidance or differential GPS (DGPS) used by coast guards and geodetic professionals). Therefore there is little hope to achieve results that are more accurate than 20 meters and we did not attempt to do so, e.g., by taking several images in which matches were found and estimate a position by computing the most likely camera position.

  Instead, we simply let the user choose from the most similar images in order to copy the coordinates or manually place the pin if those do not seem close enough to the user. The latter is an absolute necessity when the available reference dataset is not dense enough in covering the area in which the user has taken images. By letting the user interact, we avoid results that are very far away from the correct position; when the user selects an area on a map or even traces a path, it is very simple to achieve that the nearest neighbors are all within the range of 20 meters.

  The approach is designed to incrementally geotag the images of series. When the user invests time in selecting areas on maps, this effort is rewarded as this selection might be helpful for all images of a series. Furthermore, it provides the guarantee that for all images that will be processed, no "catastrophic" results will occur. "Catastrophic" in a sense that if one of the images would get tagged with a wrong location that is very far away, this error could propagate to images that will be tagged later and reuse this incorrect geotag.

- This aspect of propagating errors has to be considered when comparing to other approaches, that are designed to determine the location of a photo without significant assistance from the user. For instance, [Li et al., 2009b] reports an accuracy of less than 18.48% when images of 200 landmarks are classified just based on visual information (which corresponds to an error rate greater than 80%) and less than 9.55% for 500 landmarks. The accuracy can be improved significantly by exploiting textual and temporal information, but still the error rate for 500 landmarks would be above 50% (accuracy of 45.13% for single images with visual and textual information, 45.34% for photo streams with visual and textual information – which has to be considered a very small improvement compared to 40.58% and 41.02%, respectively, when only the textual information is used).

  With such a level of accuracy for fully automated, world-wide estimation of the image location even when restricting to popular landmarks, the derived geotags should never be added to the reference dataset without a human being reviewing the location – only through this the quality of an incrementally growing dataset

can be maintained and only by incrementally growing the dataset with images from the user the system will overcome the restriction to be only usable to tag popular landmarks. Even if the images are not added to the dataset, these geotags should be checked before being stored with the image: Geotags are used in many retrieval tasks, but in particular for personal image collections, for which known image search is of great importance. Being able to use geotags as one dimension of a faceted search is a very valuable option for the user – but only if the geotag is accurate, otherwise it will make the geolocation unusable for Known Image Search when the sought image is misplaced to a completely different location. Therefore some level of human curation is inevitable.

However, when a human user is involved in reviewing suggested geotags for images, there is little to no benefit to the end user in attempting to solve a greater extent automatically. On the contrary, by involving the user in a later stage just as a reviewer, the user will get bothered with more images for which the suggested locations need manual corrections as the user has not been able to give the system valuable input like defining a region of interest in the image or preselecting an area on a map. Unless the proposed geotags are very accurate, leaving the user out of the loop until the last stage will only add frustration.

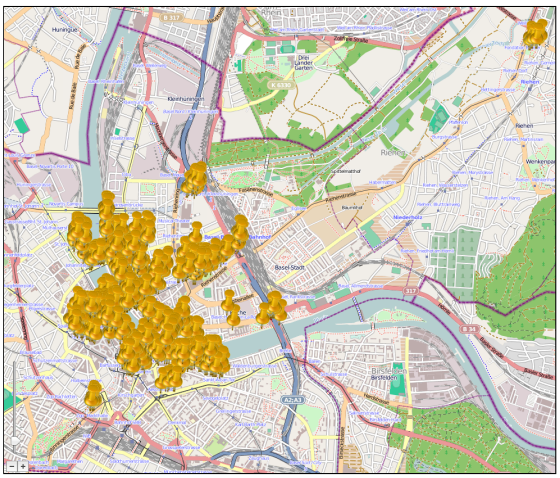
Instead of quantifying the error of proposed geotags like it was done for the fully automated systems, we will therefore show in this section some screenshots that a user will see when geotagging images. Through this, we hope to be able to provide some impressions of the retrieval quality of our approach.

10.4.5 Evaluation of Retrospective Geotagging

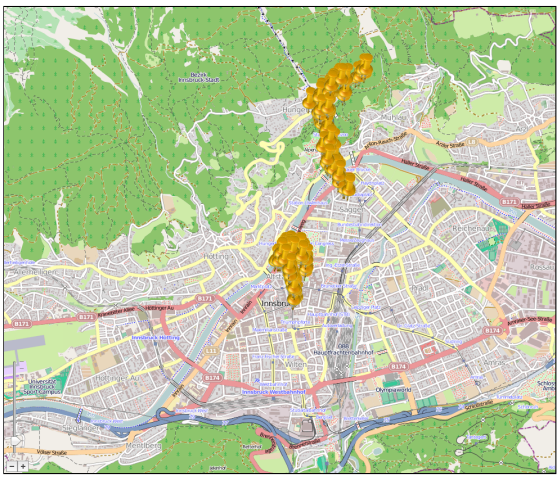
In our setup, we populated the system with 1'207 geotagged images taken at various public places in Basel. These images were collected between September 2008 and June 2009 with three different cameras, therefore providing some variability. We added also 201 geotagged images of Innsbruck in order to have also some references that are geographically disperse, but to a certain extend visually similar – therefore giving a total of 1'408 images from which "false matches" could be geographically up to approximately 300 kilometers misplaced as visible on the map in Figure 10.56. [67]

[67] The dataset was chosen mainly due to availability and knowledge of the quality of the geotags: All images were collected and geotagged by the author of this thesis. Previously used datasets, in particular for the "Where am I?" contest at the 10th IEEE International Conference on Computer Vision (ICCV 2005), could not be used as `http://research.microsoft.com/en-us/um/people/szeliski/visioncontest05/default.htm` does not include the geolocations for the images. Of course, we could have also gathered geotagged images from a picture sharing website like Flickr as done in related works [Hays and Efros, 2008, Kalogerakis et al., 2009, Crandall et al., 2009] – but in this case, one has to rely that the geotags are accurate or investigate them carefully; which is not easy for big datasets and images that were shot by somebody else, therefore the information found in the image, on map and known to the person evaluating the dataset must be sufficient to judge the accuracy of the geolocations unless the dataset is focussed just on pictures of very famous places / very popular landmarks. But exactly for these kinds of places, textual lookup performs very well, and therefore adding content-based techniques provide little benefit to the user over enabling text searches for geolocations.

(a) Overview



(b) Basel and Riehen



(c) Innsbruck

Figure 10.56: Map showing the locations of the geotagged reference images in the used dataset: The dataset contains 1,408 images mainly of urban area close to the centers of the city of Basel, Switzlerland (b) and Innsbruck, Austria (c).

Selecting areas on maps and defining time ranges are very effective filter predicates to reduce the number of false positives. In all of the searches that are presented in the following paragraphs we did not make use of these filter predicates *intentionally*. Using them would make the results even better as has been shown in Figure 10.50.

Results for Query Images from Basel

Figure 10.57 – 10.59 show some search results for query image from Basel. As with all previously used images in illustrations, the query images have been collected independently of the 1'408 geotagged images in the dataset. In particular, they have not been generated by cropping or applying affine translations on images of the dataset, but are new pictures taken at a different time.

Not all evaluations of CBIR methods that are found in literature follow such an approach: Some evaluation sets of keypoints have used generated images, in particular to check the robustness against rotations. The benefit of such an environment is, that

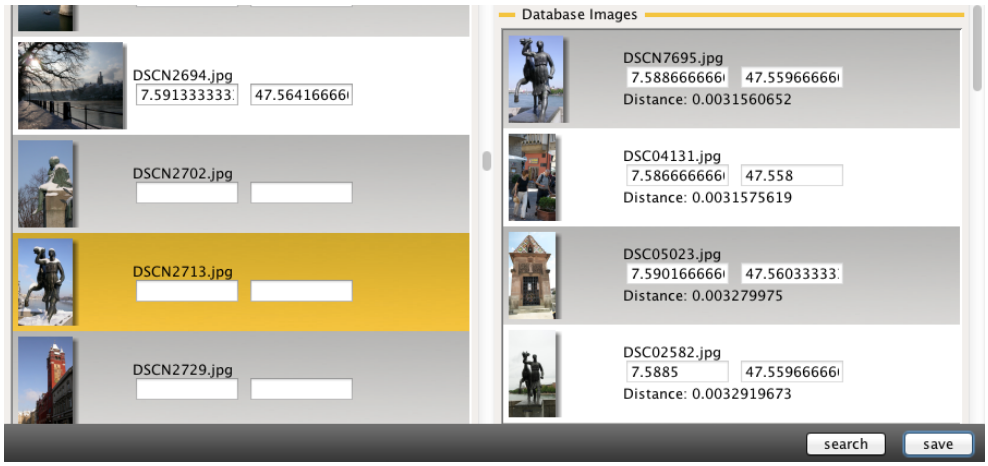


Figure 10.57: Results for DSCN2713: Rank 1 and 4 show same statue in frontal pose.

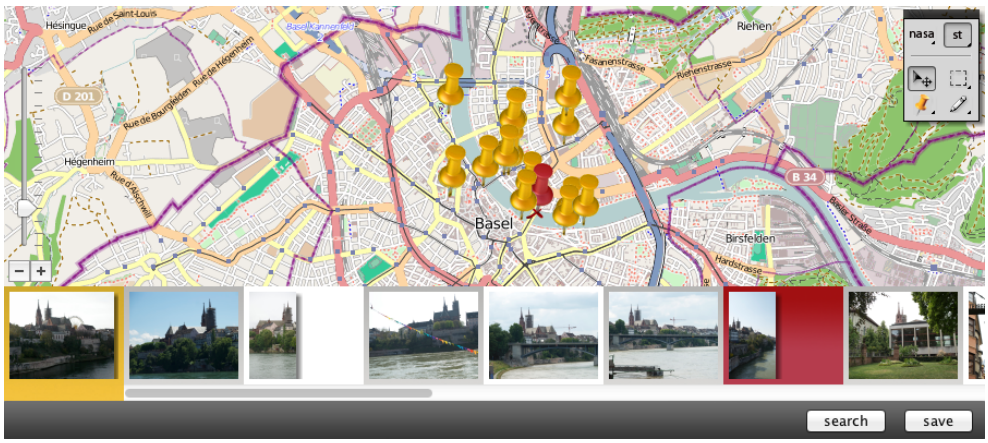


Figure 10.58: Map view on results for DSCN1699: All top ranks show the cathedral from the side of the river; sixth result (highlighted in red) has been shot from almost the same viewpoint as the query image.

arbitrary modifications of the image can get generated and evaluated automatically. The downside of such an approach is, that it can merely act as an upper bound on the reliability when realistic images are used that have not been taken in a controlled environment. Therefore we prefer an approach with a test image set that has been collected independently, therefore representing more realistically the intended task.

The timely independence of the datasets is easily visible by the snow on the statue in DSCN2713 (Figure 10.57) and the Ferris wheel behind the cathedral in DSCN1699 (Figure 10.58 and 10.59). The latter two show a number of images of the cathedral in Basel taken from the side of the river. Figure 10.59 shows the image that was taken from the most similar position as the query image DSCN1699, but in different orientation (portrait instead of landscape orientation), which also affects the image content and was therefore ranked behind other images that also show the cathedral, but taken from a different viewpoint.

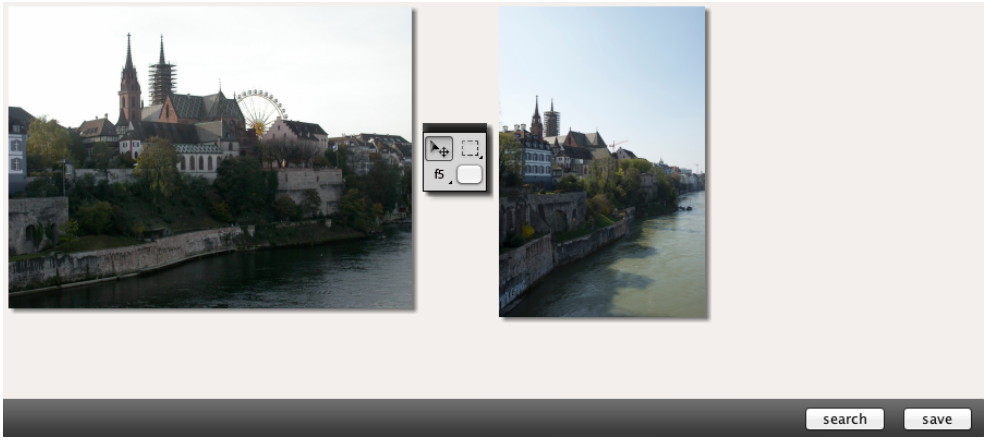


Figure 10.59: Enlarged View on Results for DSCN1699: Query image and image in database shot from similar viewpoint but with different camera orientation.

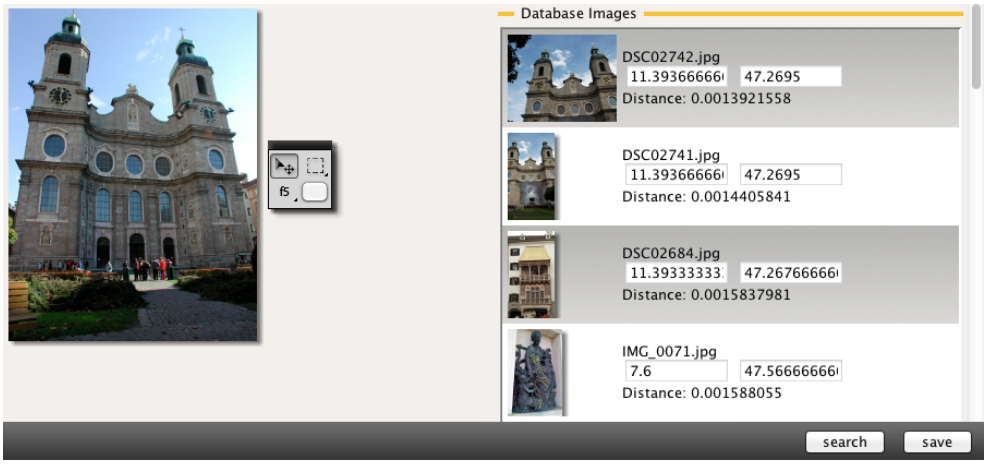


Figure 10.60: Results for DSCN2482: First two results show same building from similar viewpoint.

Better results could be achieved, if not only individual keypoints would be matched against each other, but the matching of all keypoints between two images is analyzed and optimized to determine a viewpoint that corresponds to a possible, consistent, and minimal overall image transformation. Such a transformation could also be used to estimate the viewpoint/location that is not identical to the one of the closest image, but derived from the location of that image (or several images) and the determined transformation. Such a technique has been described in [Zhang and Kosecka, 2006].

Results for Query Images from Innsbruck

The next set of results shown in Figure 10.60 – 10.63 use query images taken in Innsbruck to search for images in the same, full dataset of 1'408 images. This setup is more challenging than the previous using query images from Basel for two reasons:

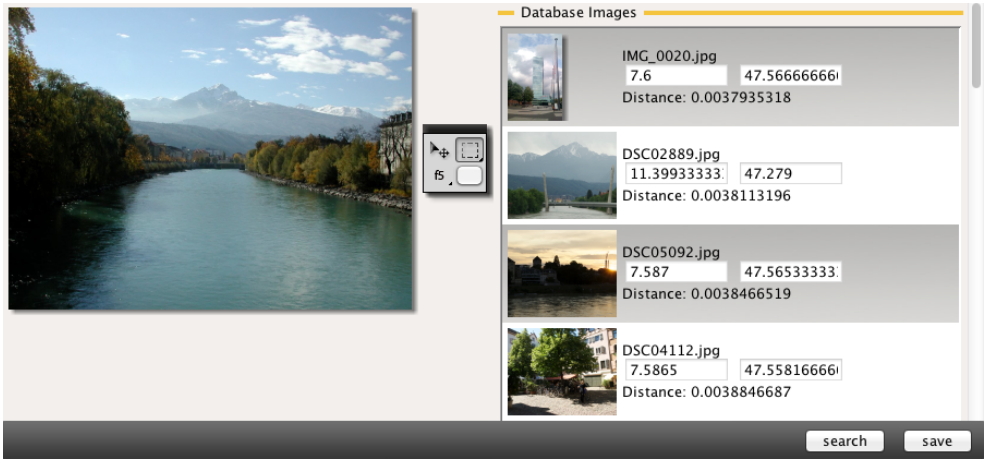


Figure 10.61: Results for 2523: Second result shows similar scene from similar position, but with different zoom level, different whether condition, and a bridge in foreground that has been built in the year time difference between the two shots.

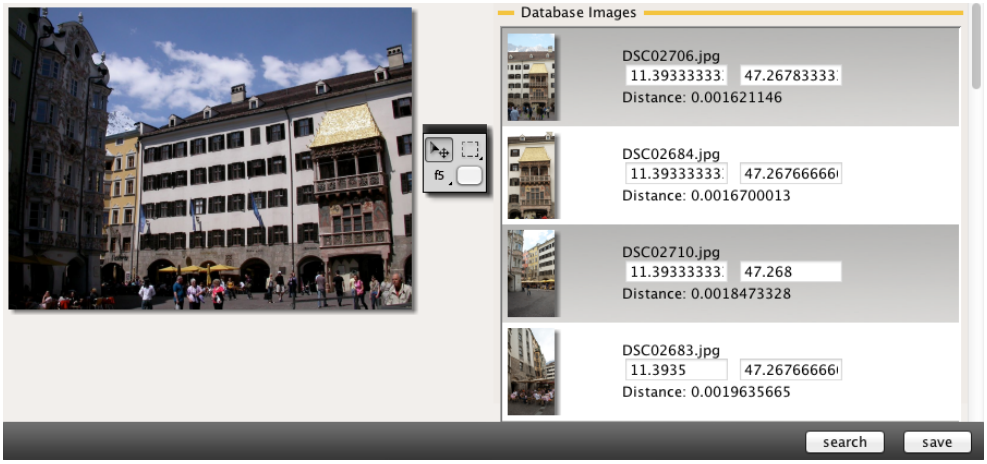


Figure 10.62: Results for DSCN2600: Ranks 1 and 2 show also shots of the "Goldenes Dachl" (Golden Roof), ranks 3 and 4 show detailed view of buildings to the left of the query image.

- Only 201 out of the 1'408 images in the dataset have been taken in Innsbruck – so the probability to pick an image that is close just by pure chance is much lower, compared to using images from Basel. As mentioned in the general setup, for all queries, no map area was selected therefore the probability of picking an image that belongs even just to the same city is about 14.27%. For picking even a picture of the same object, the chances are below 0.6%: The object of which most pictures in Innsbruck were taken is represented with 8 pictures in the dataset of 1'408 images.

- All query images have been taken with a digital compact camera in May 2008, all images in the dataset have been taken with a digital SLR in May 2009. The dataset contains images taken with both cameras for Basel.

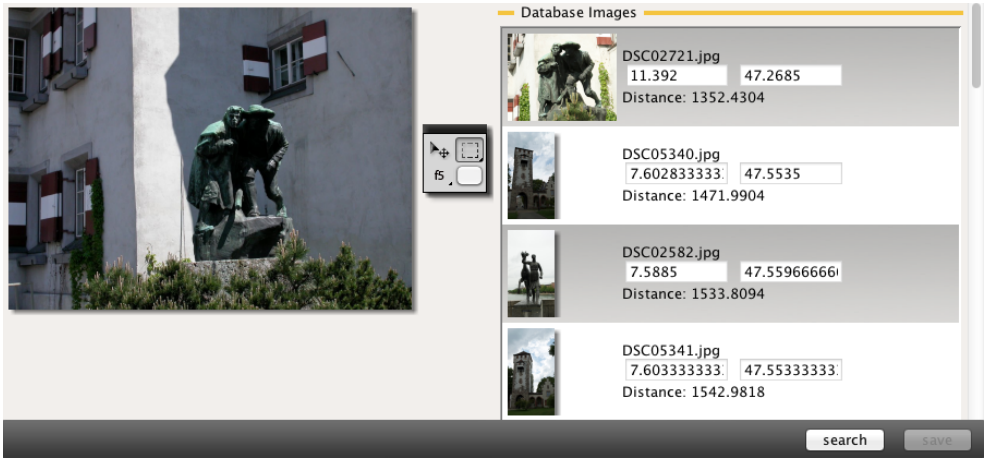


Figure 10.63: Results for DSCN2606: Rank 1 shows the same statue in front of Ottoburg. (Search performed based on Color Moments with 5 Fuzzy Regions.)

and no region of interest was used. Compared to the queries performed on images from Basel, this setup has to be considered more challenging. All queries except in Figure 10.63 were performed using SIFT descriptors as features and ranking the results using the inverse squared distance and matching keypoints only when their main orientation does not differ by more than $\pm 15°$. For Figure 10.63, color moments with five fuzzy regions were used. As one can see most easily from the longitudinal coordinates, not all results in the result lists are from Innsbruck. But all cases except for Figure 10.61 have quite accurate matches from Innsbruck in top positions. The exception Figure 10.61 shows a very good match DSC02889 as the second result. The picture was actually taken from a similar location near Innsbruck, however, the weather was quite different at that time compared to the picture taken one year before – at a time when the bridge in the foreground of DSC02889 did not yet exist.

Results for "Fountain Search"

The last setup we will discuss in here uses a slightly different which provides a solution for one of the example scenarios used in the introduction of this thesis: In Chapter 1.4.1 we introduced the *Scenario 1: Fountains in Basel*. The aim here is to determine not the geocoordinates of an image, but the name of a fountain depicted in an image.

This task has been evaluated over several years with multiple users. It should not be considered a scientific multi-user study, as the age group of the participants is certainly not fully representative: From 2008 to 2011 this scenario has been performed with school girls in the age between 11 and 14 every year as part of the *girls@science project week* in which in each year 3 groups with 3 or 4 participants per group took pictures of one of the two fountains that are located close to Petersplatz and the Department of Computer Science in Basel. To determine the name of the fountain the participants could either use the internet directly for any kind of websearch or our prototype to perform a visual search among the 423 images of fountains in Basel crawled from `http://www.brunnenfuehrer.ch`. For whatever strategy the participants applied,

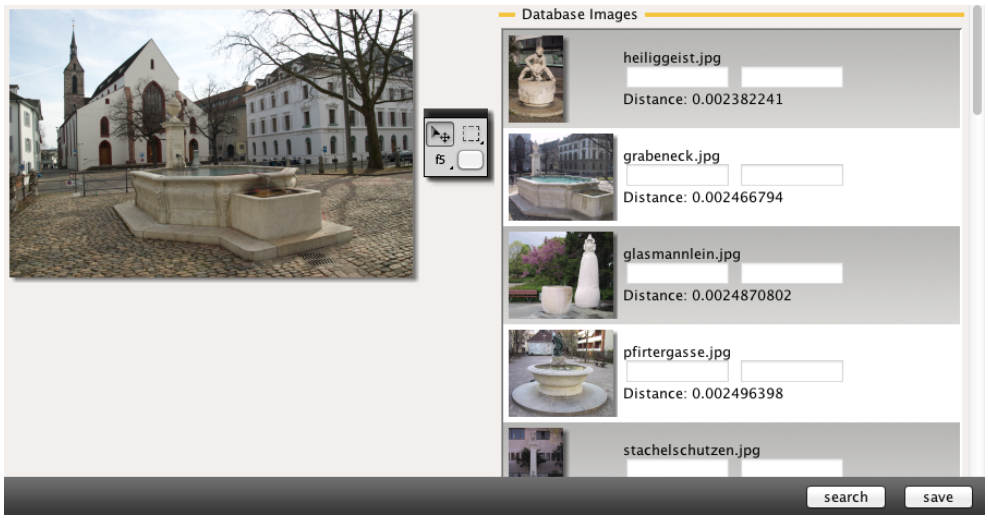


Figure 10.64: Results for DSC01269: near optimal query image. Image of sought fountain ranked second.

Figure 10.65: Results for DSC02641: query image taken from a different viewing position. Image of sought fountain was listed on rank 11.

technical assistance was provided, including introduction to any of the available search tools and search options. For instance, when regular websearch for images using a text-based search engine were used, the participants have been assisted in disambiguating search terms when only images of the fountains on Saint Peter's Square in Rome were returned. For content-based searches using our prototype, assistance involved transferring images from a camera or OLPC that was used by the participants to take pictures of the fountain.

While none of the groups was able to solve the task using only traditional keyword-based searches, every group was able to solve the task using our prototype and the images that they took. This strategy was also considered more appropriate than browsing all images (or all 170 entries for fountains, respectively) on the `http://www.`

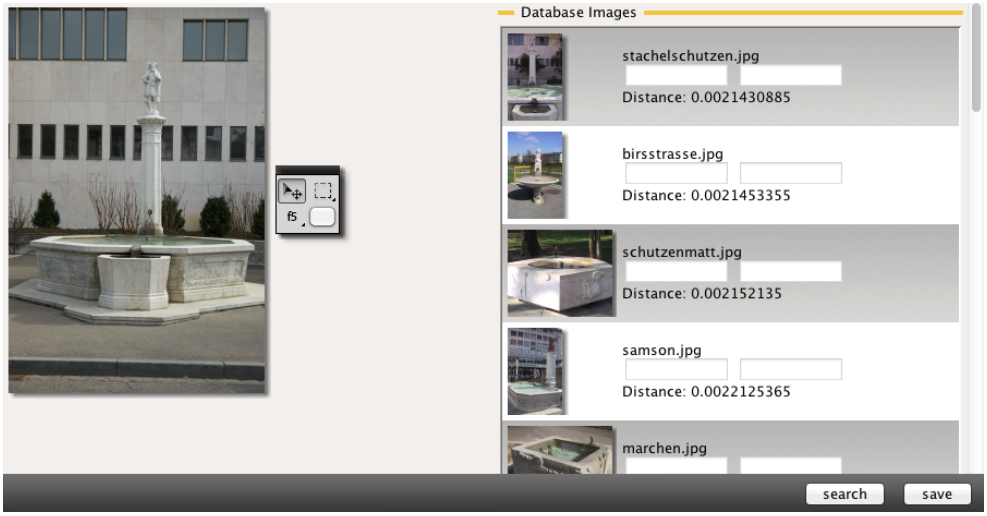


Figure 10.66: Results for DSC01275: near optimal query image. Image of sought fountain is listed in top position.

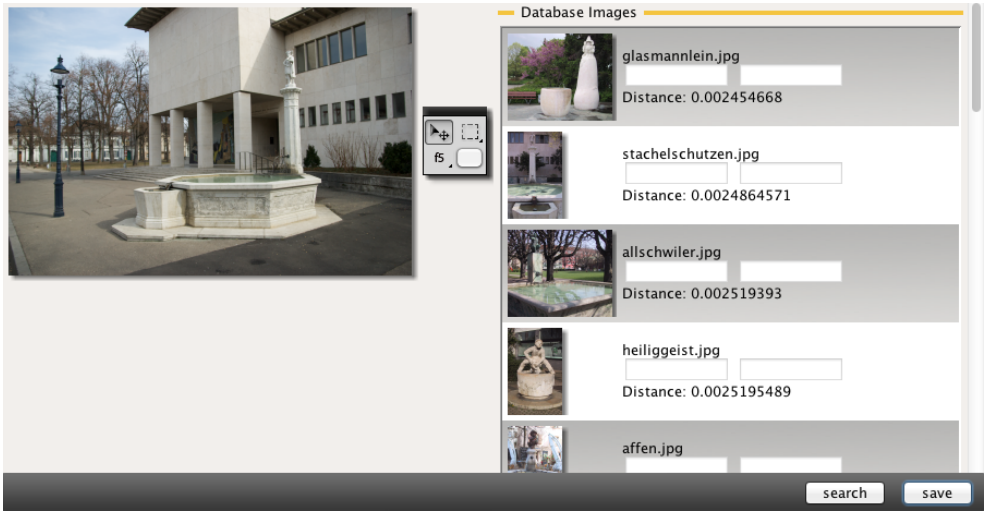


Figure 10.67: Results for DSC01276: query image taken from a different viewing position. Image of sought fountain ranked second.

`brunnenfuehrer.ch` website until the desired fountain appears. Most groups at some point started with this browsing strategy, but switched to a different strategy before it became successful.[68]

[68]Some of the results that usually appear when searching for these fountains were already shown in Chapter 1.4.1. For the particular two fountains, it is possible to solve the task in little time even without content-based image searches when at least (a) the name of the place is known (Petersplatz) and (b) the website `http://www.brunnenfuehrer.ch` is known / has been identified and (c) an external tool is available to provide full text search functionality for the website – as the website itself does not offer such a functionality by itself. For instance, `http://images.google.ch/search?q=peterplatz+site:http://www.brunnenfuehrer.ch` or `http://www.bing.com/search?q=petersplatz+site:www.brunnenfuehrer.ch` will both deliver a list with only 5 images / three hits in webpages, two of which belong to the sought fountains. However, none of the groups managed to make successful use of

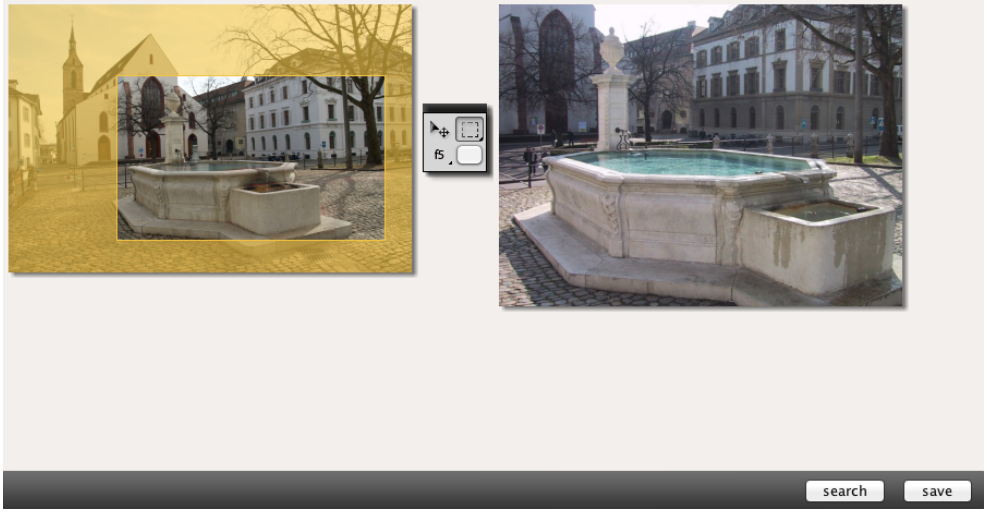


Figure 10.68: Enlarged view of DSC01269 and sought fountain.

Unfortunately, the images at this website are not geotagged, so we cannot use the map view and selection of areas that is usually quite helpful in retrospective geotagging. The first fountain shown in Figure 10.64 and 10.65 is named "Grabeneck-Brunnen"[69], the second in Figure 10.66 and 10.67 "Stachelschützen-Brunnen"[70].

The screenshots of searches show, that using a visual example and SIFT search with keypoint rotation of ± 15 and sum of inverted squared distances for ranking returns an image of the sought fountain within the first 20 results out of 423 images.[71]

Figure 10.69 and 10.69 show the enlarged view of the query images and the enlarged image of the sought fountain. The region of interest selection was used to highlight the approximate area that corresponds to the image in the database; this selection was not used in search and not necessary to achieve the ranks in Figure 10.64 – 10.67. Figure 10.70 shows results when Gabor Texture Moments instead of SIFT keypoints are used for search. As this feature does not have just the support of very small areas around keypoints and Gabor texture moments capture also the main orientation of edges, the results correspond very well to what a user may consider similar images.

---

such advanced search tools, which certainly was also due to the comparably low age of the participants. During the last months, there have appeared at least two new images on the web pages that show the Grabeneck-Brunnen and use the terms Petersplatz, Basel as well as Grabeneck-Brunnen prominently; they can easily be found using keyword based searches and solve the task: `http://www.bs.ch/bilder?act=detail&oid=30113` and `http://galerie.alfpa.ch/v/Brunnen+Basel+und+Umgebung/Grabeneckbrunnen+Petersplatz+2.JPG.html?g2_detail=1`. The task therefore is now much easier to solve even without content-based search tools than it has been for several years – at least, if one knows the name of the place.

[69] Brunnenführer-Referenz: 50, `http://www.brunnenfuehrer.ch/brunnen/grabeneck.htm`

[70] Brunnenführer-Referenz: 123, `http://www.brunnenfuehrer.ch/brunnen/stachelschutzen.htm`

[71] For some fountains of the 170 fountains there are more than one image in the database; in particular there are only 74 fountains for which an image in the resolution of 640×480 pixels exists. For the remaining remaining 96 fountains, only images of 225 pixels do exist; minimal resolution is 84×225, maximal resolution 390×225 – which is still fairly low for reliable retrieval.

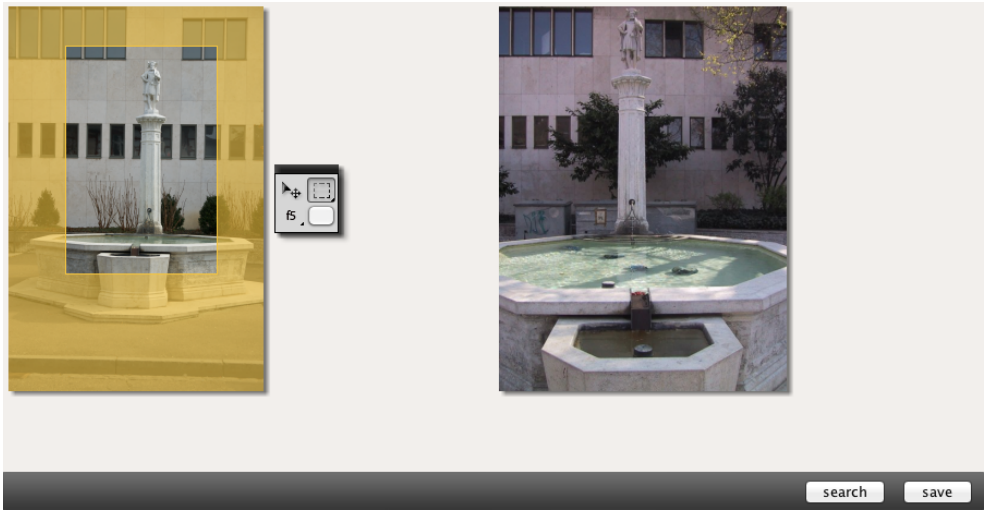


Figure 10.69: Enlarged view of DSC01275 and sought fountain.

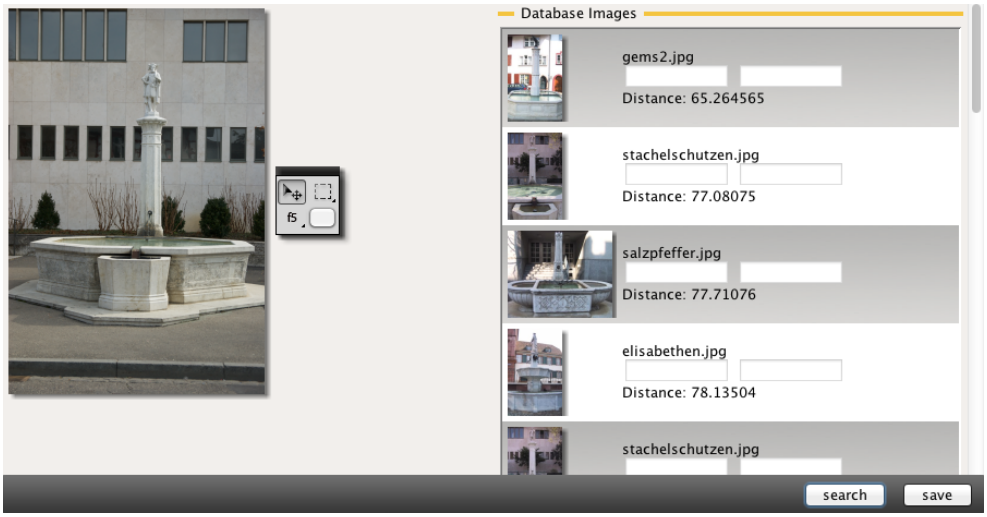


Figure 10.70: Results for DSC01275 when ranked by Gabor Texture Moments with 5 Fuzzy Regions; second and fifth result show the sought fountain.

Conclusion

The problem of identifying the place where an image has been shot has been addressed in a number of previous works. In our approach, we focussed on how to assist the user in a particular instance of this problem: The task of assigning geotags to the user's images. We added the possibility for the user to define areas on maps to consider for similar images, define a time range for known events, define a region of interest in the images that the user wants to tag, enforcing a heuristics to restrict the allowed rotation. When good results are found, the user can simply copy their geotags. In worst case, where no good match is found, e.g., when the user took really a unique shot of the area, the user can still easily assign the location by placing a "pin" on the map manually. As this newly tagged image is added to the user's database of geotagged images,

subsequent content-based image searches performed by the user will be able to reuse the manually assigned geolocation. Applying the approach to a slightly different scenario of identifying a particular fountain showed the added benefit that content-based searches can provide for such tasks.

What is important for this very user-centric approach is, that the user doesn't have to wait for a long time for search results; otherwise the user's flow of interaction will be interrupted, which would reduce the added value of the tools compared to manually tagging the images or a browsing strategy significantly. The next section will focus on how query execution can be optimized. As main building blocks for query execution are shared among all example tasks in Section 10.2, Section 10.3, and Section 10.4, any of the approaches will immediately benefit from these performance optimizations.

10.5 Reducing Execution Time to Present Search Results

For the tasks the user wants to perform, an important aspect is how long it takes to finish it successfully. This depends on several aspects, including the strategies that the user chooses, the content available in the system, the ability of the user to actually make use of the system, the quality of the results that the system delivers to the user for a given query, and last but not least: The time it takes to compute the results and present them to the user.

In the following we will describe which options to reduce the time exist in general, but also propose some particular techniques that reduce the time to execute similarity searches. For this it is important to keep in mind one general observation about similarity searches: Evaluating tolerant matches can take considerably more time than performing less tolerant or even exact matches. The reason for this is, essentially, that for less tolerant matches the system only has to evaluate the query until some document in the collection can safely be discarded – and in the extreme case of exact match this is already the case as soon as the slightest deviation is detected. For tolerant matching, any deviation has to be quantized, accumulated, and in many cases also ranked against the score of other documents before it is safe to discard it or keep it as a result. There are two different natures of limitations on the execution time for performing similarity searches:[72]

- *I/O bound*: The execution speed is limited by the capabilities of the system to transfer the data required to evaluate the search and deliver the results.

- *CPU bound*: The execution speed is limited by the computation capabilities of the system to process the data.

Which of the two will be the limiting major factor depends not only on the system, but also on the particular task that is being executed. Particular techniques may improve a single or both aspects, I/O and CPU to a varying extent.

This section is dedicated exclusively on optimizing the implementation of building blocks involved in Query Formulation and Execution. Some of the proposed strategies are fairly generic in nature and can also be implemented in a fairly generic and reusable way.

This section will be structured as follows: Section 10.5.1 gives an overview on what strategies do exist to optimize the execution of queries that involve similarity search in general. Section 10.5.2 – 10.5.4 presents selected techniques that we have developed and refined in much more detail. In Section 10.5.5 – 10.5.7 we show the impact of applying the selected techniques to our unoptimized approaches that have been used in Section 10.2 – 10.4 to perform user tasks. Finally, Section 10.5.8 summarizes quickly the results.

[72] Cf. also [Weber et al., 2000a, Weber, 2001]

10.5.1 Overview of Optimization Techniques

To reduce the amount of time it takes to execute a query, the system may preprocess some of the data before a query is issued by the user. Such optimizations can happen to the insertion or update of content and therefore *ahead of (query execution) time*. The remanding optimizations will be performed when the query is executed and therefore *just in time*.

Optimizations at Insertion Time

The time to compute the (dis-)similarity does not only depend on the measure used for it, but also the selected features. Many commonly used features are expressed as vectors of integers or floating point values – the higher the dimensionality of these vectors, the longer the computation of a (dis-)similarity takes. In addition to this computation, also the created I/O costs become important: Reading long feature vectors requires more time than reading short vectors or just a single value per document.

Of course, techniques exist to reduce the time needed for evaluation without any loss of expressivity or quality. However, one of the effective approaches is to compute as much as possible even before query execution starts. This is possible for all parts of the search process that are independent of the concrete query image:

- *Feature Extraction:* The features of the images in the collection can be extracted as soon as the image is stored in content management as already mentioned in the context of maintaining consistency in Chapter 4.3 on page 89. When an image can be member of more than one collection, keeping the feature referenced to the image can assure that the feature will get extracted only once – not for every collection it becomes a member. And for systems that support virtual collections where membership is dynamically evaluated, this is one of the few possibilities to perform similarity search without having to extract the features of the images in the collections at the time of query execution.

  As the content model may allow any image to be member of more than one collection and the content-based features are not affected by collection membership, storing the features in Content Management primarily and adding it to indexes per collection avoid the need to re-extract the same features over and over again.

  For supporting queries over virtual collections, storing the features directly associated to the image is the only reasonable possibility for ahead-of-time feature extraction.

  Furthermore, in order to make best use of the I/O capabilities, the features can also be stored in a way, that they can get read as fast as possible. For a system using traditional hard disk drives (HDD), sequential reading of files is much faster than random reads. If membership to collections is fixed (non-virtual collection), all features of same type for all images of the collection can be stored (or replicated) to a single file; thus allowing to access a single file per query feature type at query time. This also has the side-effect that it may reduce the number of blocks occupied in the file system compared to storing each feature in an individual file. Also

modern solid-state drives (SSD) will benefit from this property even if they do not have such a big difference in access times for sequential and random reads. To reduce the needed space on disk which may also reduce the time it takes to read an entire file, a compression algorithm may be applied to the features before storing them to files. This will reduce the time to read features if (and only if) the time needed to uncompress the features is not greater than the time-savings in reading the smaller files. These I/O based optimizations are always beneficial, as long as the documents inside the collections remain fairly static: It wouldn't really make sense to replicate and/or compress features for the purpose of improved reading performance when theses features get more frequently updated than read.

- *Precompute (Partial) Distances:* When indexes are built, these are based on the distances between the images in the collection. Therefore already at the time of index creation and index update, distances will be computed that can be exploited at query time.[73]

  In particular when the distance measure satisfies the triangle inequality of Equation (5.11) from page 127, this can be exploited by deriving bounds on the distance even if the query item is not yet known. Let $\mathfrak{Q}, \mathfrak{R}, \mathfrak{S}$ be images with $\mathfrak{Q}$ being the (so far unknown) query image and $\mathfrak{R}$ and $\mathfrak{S}$ two reference images inside a collection. The visual features $\Phi(\mathfrak{R})$ and $\Phi(\mathfrak{S})$ can therefore already get extracted and stored in content management. Some distance measures require additional parameters at execution time, e.g., weights, but many distance functions do not. For the latter the distance $\Delta(\Phi(\mathfrak{R}), \Phi(\mathfrak{S}))$ can get computed already which we will denote as $d_{\mathfrak{R},\mathfrak{S}}$ and also the opposite direction $d_{\mathfrak{S},\mathfrak{R}} = \Delta(\Phi(\mathfrak{S}), \Phi(\mathfrak{R}))$. In case of a symmetric distance measure that satisfies also Equation (5.10), $d_{\mathfrak{R},\mathfrak{S}} = d_{\mathfrak{S},\mathfrak{R}}$.

  The triangle inequality now states that for any query image $\mathfrak{Q}$:

  $$\Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{S})) \leq \Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{R})) + d_{\mathfrak{R},\mathfrak{S}}$$

  but also:

  $$\Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{R})) \leq \Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{S})) + d_{\mathfrak{S},\mathfrak{R}}$$

  which can be reformulated as:

  $$\Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{R})) - d_{\mathfrak{S},\mathfrak{R}} \leq \Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{S}))$$

  Therefore, at query time, when the distance $d_{\mathfrak{Q},\mathfrak{R}}$ to the first image in the collection $\mathfrak{R}$ gets computed, a precomputed distance can be added to this distance to define an lower and upper bound on the distance:

  $$d^{lb}_{\mathfrak{Q},\mathfrak{S}} = d_{\mathfrak{Q},\mathfrak{R}} - d_{\mathfrak{S},\mathfrak{R}}$$

  $$d^{ub}_{\mathfrak{Q},\mathfrak{S}} = d_{\mathfrak{Q},\mathfrak{R}} + d_{\mathfrak{R},\mathfrak{S}}$$

[73] As an example, the VisualRank proposed in [Jing and Baluja, 2008] is used similarly to Googles PageRank by precomputing the visual similarity between images in the collection.

with

$$d^{lb}_{\mathfrak{Q},\mathfrak{S}} \leq \Delta(\Phi(\mathfrak{Q}), \Phi(\mathfrak{S})) \leq d^{ub}_{\mathfrak{Q},\mathfrak{S}}$$

The smaller the distances $d_{\mathfrak{R},\mathfrak{S}}$ and $d_{\mathfrak{S},\mathfrak{R}}$ are, the more tight the bounds get. Any index structure exploiting this fact will not preserve all partial distances, but store the features of images with low distances close to each other and keep track of the distances for these features. In case of a range search, any distance to a feature does not need to be computed if its lower bound is already outside the range. For a kNN search, the same is true is the lower bound is not better than any of the k best distances seen so far.

Another important function at this stage is to generate statistics about the distance distribution that can be used for distance normalization later.

- Not directly related to feature extraction, but very similar in practice and very important for the perception of the speed for retrieval of the overall system is *Thumbnail Generation:* The small images to present to the user in the result list can also be generated as soon as the images get inserted into content management. Thus, they will be available at the same time a query has been executed and through them the results can be presented much quicker than by displaying or scaling the full size images directly. In contrast to *Feature Extraction*, thumbnails will not be needed before distance computation takes place, but after distance computation has been finished. The generation of thumbnails however can happen exactly at the same time – whenever an image is added to content management.

These steps are illustrated in Figure 10.71. If we recall Figure 3.1 from Chapter 3 (which is displayed here again in Figure 10.72), the described possibilities to optimize ahead of query execution are all found where building blocks need to interact: Content Management can trigger actions as soon as new or updated content becomes available, but the result of the action is only useful in the context of other building blocks – features and (partial) distances for processing similarity searches, thumbnails to assist the user interaction and not delay result delivery after a query execution.

Optimizations at Query Time

Some of the optimizations at query time may exploit the work that has been performed at insertion time. For instance, it may be possible to use the pre-extracted features and partially computed distances.

Optimization techniques that are available at query time can roughly be grouped into the following steps of the execution of the search process:

1. *Filter Items:* Reduce the set of images which are considered in computation, e.g., based on keywords, metadata, sub-collection, access rights. Therefore this depends usually a lot on which information is available in content management and the possibility to express the filtering constraint as a predicate.

2. *Use Index:* Reduce time to generate result by exploiting better organization of how documents and there features are stored. This may include techniques for

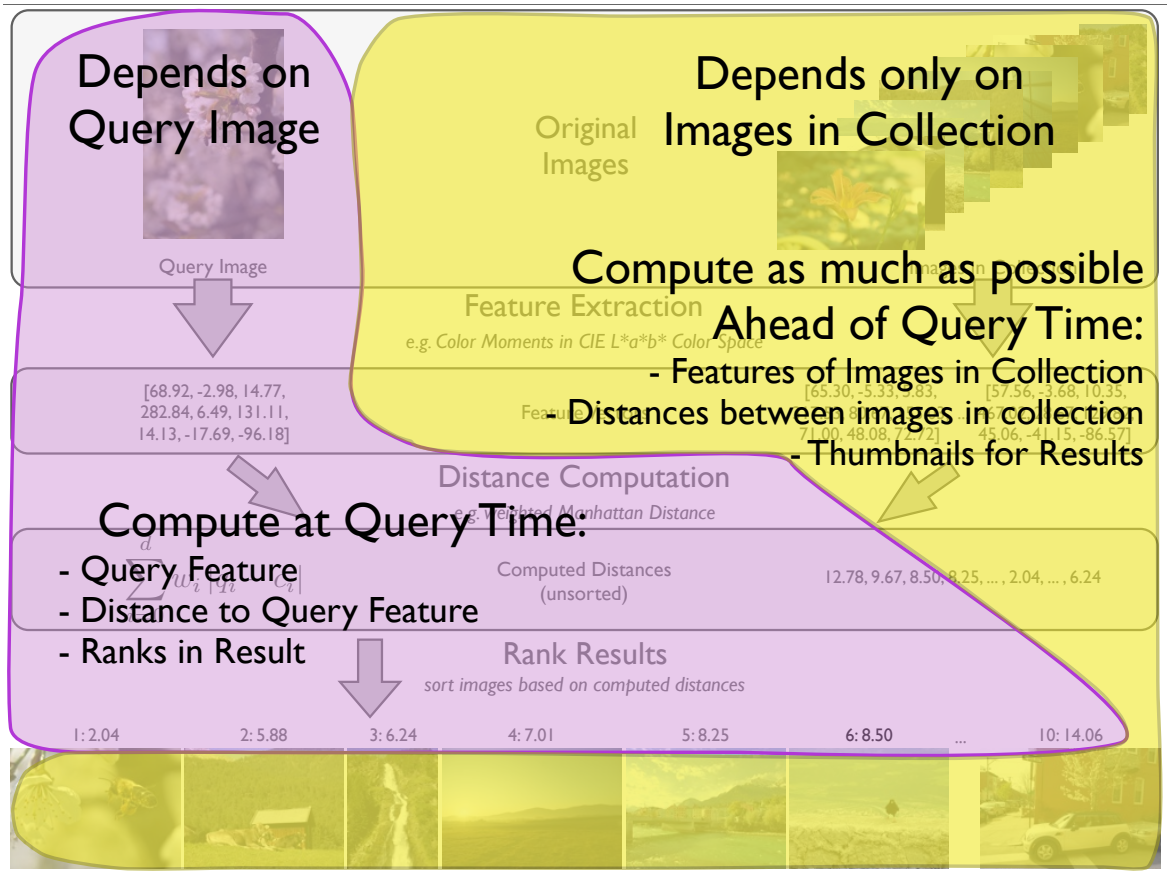


Figure 10.71: Computations during and ahead of Query Execution: While the distances to the query feature and the relative ranks can only be computed as soon as the query image is known, the features of individual images in the collection, distances between them and the thumbnails for the result list can already be computed as soon as images are added to content management.

performing the filtering mentioned before faster, e.g., an inverted index for text retrieval or a B/B^+-tree in a database to efficiently select images based on numeric properties, but also includes high-dimensional feature index structures for the similarity search of features.

3. *Parallelize Computation:* Perform Feature Extraction and Distance Computation in parallel. Common techniques to perform such computations in parallel may used multiple CPU cores in a single node, offloading part of the work to GPGPUs that are optimized on performing parallel activities, as well as using multiple nodes in a network. The optimal setup depends on the overhead to perform the computation in parallel and aggregate the results. For feature extraction, each image and each feature of an image can be extracted in isolation – which makes it trivial to parallelize. But as this parallelization requires also that the image is available at any involved node, it may require additional transfers of the image file and decompressing it.

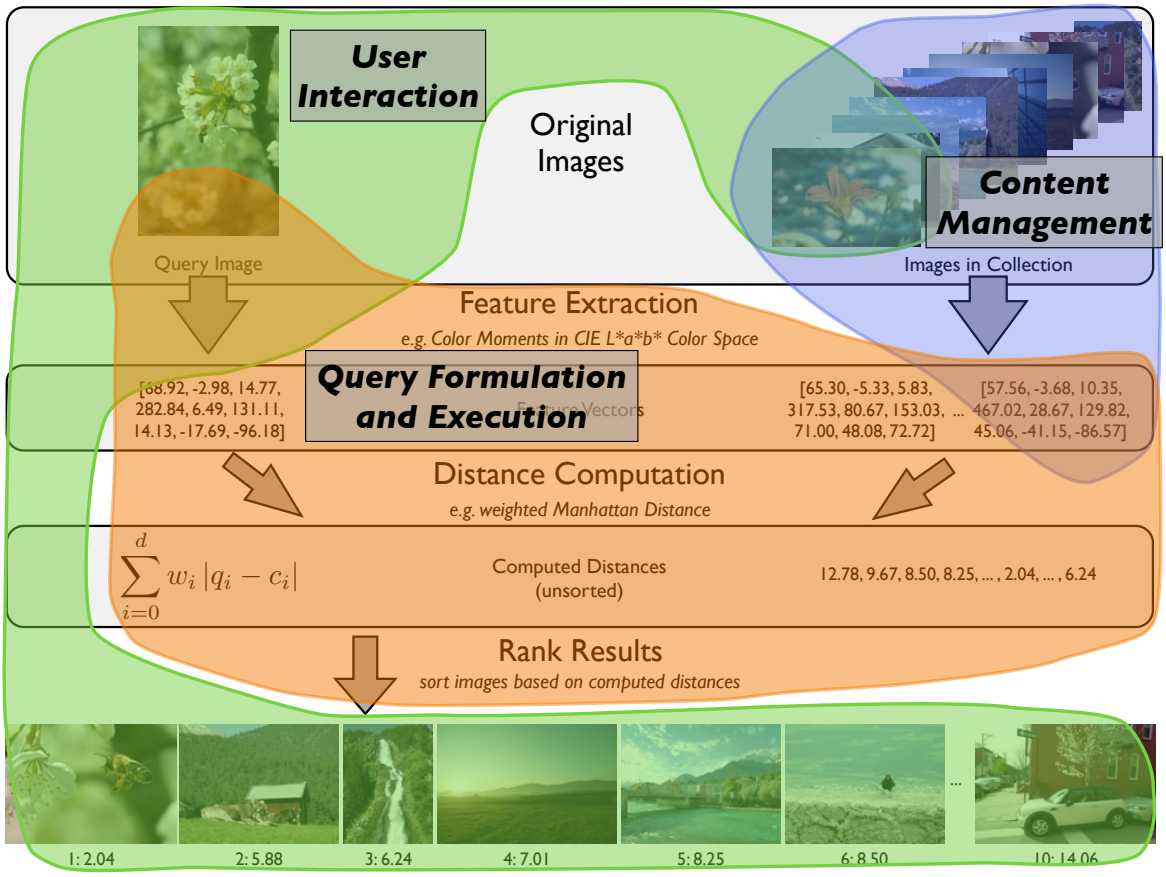


Figure 10.72: Illustration of Building Blocks involved in Search Process: Content Management (blue), Query Formulation and Execution (orange), and User Interaction (green).

4. *Approximate Answer:* Do not compute the exact answer, but approximate if an approximation is still close enough, e.g., if there is a faster and a slower distance measure and the faster measure is almost as good as the slow one, use only the fast one. Possibilities inside the retrieval process are to transform the space in which distances are computed into one where answers can be found more quidkly (e.g., by quantizing exact feature values into bins or reducing the dimensionality by performing a principal component analysis PCA)[74] or end the search process before the final result is determined completly. [Patella and Ciaccia, 2008] presents a classification scheme and overview for such approaches; [Patella and Ciaccia, 2008,

[74]Notice that dimensionality reduction can lead to very good results, even improve search quality by removing "noise" from the dataset. However, there are also many pitfalls that lead to a situation, where the findings in the space with reduced dimensionality lead to false conclusions. These dangers have been visualized in a very impressive way in [Keogh, 2011] and [Lin and Keogh, 2003]. For similarity search this means that in reduced space, some similar items may be quite distant in the reduced space and some dissimilar items might get located very close. It all depends how well the transformation to the reduced space preserves the perception of similarity of the user.

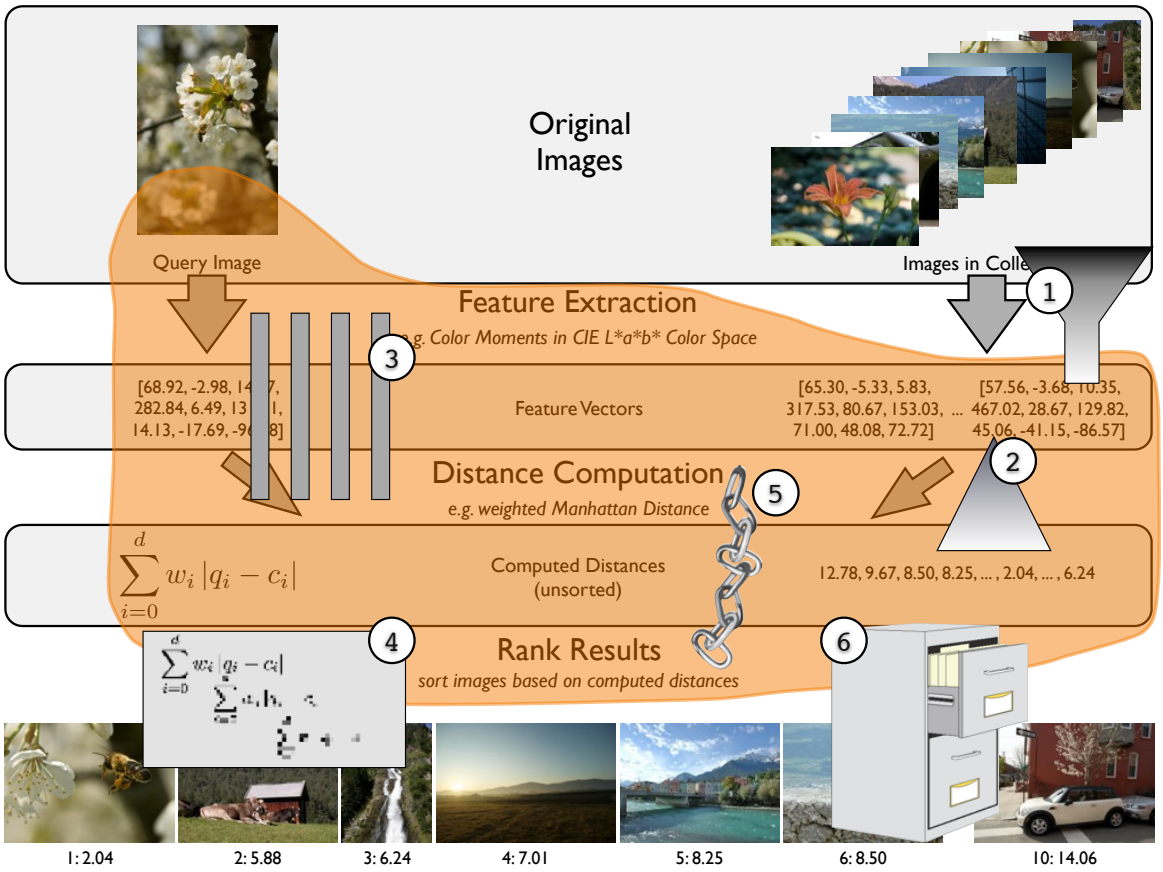


Figure 10.73: Key Approaches to Optimize Query Execution: 1. Filter Items, 2. Use Index, 3. Parallelize Computation, 4. Approximate Answer, 5. Link Computation and Ranking, 6. Cache Results

p. 310] names the first approximation type *changing space (CS)*, the second *reducing comparisons (RC)*.[75]

Some of these approaches allow to refine the results to return also the exact results if these are needed, therefore allow to trade retrieval quality for response time. For instance, [Weber and Böhm, 2000] describes a variants over the filter-and-refinement strategy in [Weber et al., 1998] for which it can determine bounds on the approximation error and switch to the exact answer by performing the complete refinement step.

Other approaches, in particular techniques that construct a so-called *visual codebook* that are also used to reduce the semantic gap (cf. Chapter 5.2) can usu-

[75] Alternative could also be to use a "coarse to fine" strategy that uses either IDM with a smaller resolution, e.g., 16×16 pixels with also scaled parameters for warp range and local context, or just determine the candidates with a much smaller local context as this is the parameter that adds most computational complexity. However, none of the approaches is able to deliver to tight probabilistic bounds on the approximation error that is introduced with such a technique in contrast to many of the approaches in [Patella and Ciaccia, 2008] – and therefore do also not integrate well with a "filter and refinement" strategy to recover the exact answer from the approximated results.

ally not deliver exact answers: The quantization of local features to *visual words* remains approximate and the main performance improvement is achieved by using a *bag-of-(visual)-words* (*BoVW* a.k.a. *bag-of-features* or *BoF*) model to reuse techniques from text retrieval like inverted indexes and *tf-idf* weighting schemes (cf. [Sivic and Zisserman, 2003, Yang et al., 2007]). [Jegou et al., 2008] proposes Hamming Embeddings instead of the common *k*-means clustering to reduce the approximation error.[76]

5. *Link Computation and Ranking:* Do not compute all answers, but only the ones that will be presented to the user, e.g., if only 20 results are displayed, the exact distance of the 21st item is of no interest. This principle is also the basis of efficient filtering (1) and indexing (2), but is not limited to these. Notice that this approach may be conflicting partially with parallelization (3) of distance computation: The final ranking will represent global knowledge – any optimization that requires such global knowledge requires additional synchronization between participating nodes. Synchronizing this knowledge may create additional overhead for parallel execution depending on the dynamic behavior of the knowledge: In case of a range search, the criterion is absolute (the range) and remains stable, therefore does not require synchronization. In case of a *k*NN search, the criterion would be the global list of best scores achieved so far – and this is changing whenever a node finds a better result than the *k* best found so far. Of course, every node can independently determine the *k* nearest neighbors, thus not fully exploiting this optimization strategy, but being able to act autonomously until the local result lists get merged into the global result; so parallelization and this strategy are far from being mutually exclusive.

6. *Cache Results:* Once a result was generated, remember it in part or total in case it gets requested again. Caching is not restricted to the end results; it can be also very effective for intermediate answers and also data that is necessary to generate intermediate answers, in particular index / feature files.

The remainder of the chapter will focus on some selected techniques and will study their impact in detail.

10.5.2 Link Computation and Ranking using Early Termination

One of the major differences between the simple query execution in Pseudo Code Listing 10.1 and RANGESEARCH in Pseudo Code Listing 10.2 / NEARESTNEIGH-

[76]The general model of treating features as "words" and the problems with scalability on large-scale tests has been criticized in [Pavlidis, 2008] as image parts with different human interpretation may get mapped onto the same features.

> It might be useful to provide an analogy of this problem in text retrieval. Suppose that we decide to use only the first three letters of each word. Thus "app" would be mapped to "apple", but also to "application", "apparition", "apparel", etc. If one tries this method on a small collection that includes mostly articles on food, the abbreviation will work fine, but it will fail in general.

BORSEARCH in Pseudo Code Listing 10.3 is the selectivity: Line 5 in RANGESEARCH and Line 10 in NEARESTNEIGHBORSEARCH contain an **if**-statement to select some images based on the computed distance. This criterion would already be known on Line 4 where the distance is computed in both cases.

Modifications to Terminate the Computation of Individual Distances Early

Computing the distance can be costly in high-dimensional feature spaces and/or with distance measures that have to solve complex optimization problems. As an example, any computation time of a Minkowski norm, e.g., the L_2-norm in Listing 10.17 grows linear with dimensionality of the vector.[77]

```
L2(x⃗, y⃗)
1  sum ← 0
2  for i ← 1 to vec-length(x⃗)
3       do
4            sum ← sum + (x[i] − y[i])^2
5  return √sum
```

Pseudo Code Listing 10.17: Simple L_2 a.k.a. Euclidean distance

```
L2(x⃗, y⃗, t)
1  sum ← 0
2  maxSum ← t^2 ▷ Compute maximum sum from distance threshold
3  for i ← 1 to vec-length(x⃗)
4       do
5            sum ← sum + (x[i] − y[i])^2
6            if sum > maxSum ▷ Threshold exceeded, terminate early
7                 do
8                      return ∞
9  return √sum
```

Pseudo Code Listing 10.18: Early Termination variant of the L_2 a.k.a. Euclidean distance

In contrast, Listing 10.18 uses an Early Termination Strategy [Springmann et al., 2008] which aborts as soon as the partially computed distance already has to exceed the

[77] At least when leaving aside low-level effects of CPU caches and vector operations / SIMD instructions that might be available on current processor technologies. However, for any very high-dimensional feature vector, those effects will not be sufficient to make the computation of the distance measure of a 128-dimensional vector as used in SIFT descriptors as fast as the computation on a 1-dimensional vector (single value) as it is common for instance for database queries.

threshold.[78] This means that for features which are very dissimilar, the computation of the distance does not need to be performed until the entire feature has been compared to the query, but just until it exceeds the threshold. This is not only possible for the Euclidean distance or Minkowski norms, but for any distance measure that considers several components and any component can only increase the intermediate results – which is the common case for distance measures that fulfill at least Equation (5.7) (*Non-negativity*). By terminating the computation when the threshold is exceeded, the overall computational cost of the comparison of features are reduced. The threshold is derived from the used search primitive.

Range Search using Early Termination

```
RANGESEARCH(Ω, Docs, Φ, Δ, range)
1  Res ← ∅
2  For each ℜ ∈ Docs
3       do
4          dist ← Δ(Φ(Ω), Φ({ℜ}), range)
5          if dist ≤ range
6             then Res ← Res ∪ {(dist, ℜ)}
7  ▷ Optional: SORT(Res) to return sorted order
8  return Res
```

Pseudo Code Listing 10.19: Range search using Early Termination in distance computation

For a range search, the threshold is trivial to derive: the threshold is the range that search results have to stay in. Pseudo Code Listing 10.19 shows the variant of the range search that uses the Early Termination Strategy. Another minor optimization tweak that is achieved by this implementation is achieved in the way the distance computation terminates early: Because Line 8 in Pseudo Code Listing 8 returns a special value (return value is ∞), it avoids for any feature that will not become a result the computation of the square root on Line 9 – which is on common hardware architectures a fairly costly operation.[79]

[78] Notice the overloaded method signature with the threshold t as parameter. All modifications w.r.t. to Listing 10.17 are highlighted in blue color. Expressing the definition in a declarative, mathematical form would not be able to show where the improvement is gained:

$$L_2(\vec{x}, \vec{y}, t) = \begin{cases} L_2(\vec{x}, \vec{y}) & \text{if } L_2(\vec{x}, \vec{y}) \leq t \\ \infty & \text{otherwise} \end{cases}$$

[79] This is not limited to the Euclidean distance, but any Minkowski norm L_m with $m = 1$ being the exception, as any dedicated implementation for L_1 a.k.a. as Manhattan distance would not compute a root anyway.

Nearest Neighbor Search using Early Termination

```
NEARESTNEIGHBORSEARCH(𝔔, Docs, Φ, Δ, k)
 1  Res ← LIST of size k
 2  t ← ∞
 3  For each ℜ ∈ Docs
 4       do
 5          dist ← Δ(Φ(𝔔), Φ({ℜ}), t)
 6          if length(Res) < k
 7            then
 8                 INSERT-SORTED(Res, length(Res) + 1, dist, ℜ)
 9            else
10                 (dist_k, img_k) ← Res[k]
11                 if dist < dist_k
12                   then
13                        INSERT-SORTED(RES, K, DIST, ℜ)
14                        (t, img_k) ← Res[k] ▷ Update threshold
15  return Res
```

Pseudo Code Listing 10.20: Nearest neighbor search using Early Termination Strategy

For a kNN search as shown in Pseudo Code Listing 10.20, the threshold for terminating distance computation early is derived from the list of best distances that have been seen so far. This is performed on Line 14 – as INSERT-SORTED maintains a sorted list of all k best distances seen, the last entry of the list has the value that should be used as the new threshold.[80]

Another approach that avoids computing the square root for Minkowski norms is, to exploit a property of convex functions that allows to search for nearest neighbors or results in a range in a transformed space (cf. [Yi and Faloutsos, 2000]). The most popular example is L_{2^2} a.k.a. sum of squared differences (cf. [Torralba et al., 2008, p. 1961]):

$$L_{2^2}(\vec{x}, \vec{y}) = L_2(\vec{x}, \vec{y})^2 = \sum_{i=0}^{n} (x_i - y_i)^2 \tag{10.15}$$

The ordering returned by L_{2^2} is exactly the same as with L_2 (proof can be found in [Yi and Faloutsos, 2000]), therefore for kNN searches it can be used directly as a replacement; for range searches it is possible to adapt the range to $range \times range$ to get the same items as results. It is therefore one of the techniques that would fall into the optimization category *Approximate Answers* by *Change Space (CS)* [Patella and Ciaccia, 2008, p. 310]; for this particular technique, the exact, non-approximated answer can simply be derived by taking the square root of the found L_{2^2} distance for the items inside the result set. As Pseudo Code Listing 10.18 avoids taking the square root of all distances that are not among the results, L_{2^2} will not perform significantly faster than L_2.

[80] Considering the discussion of the simple implementation in Section 10.1 on page 205, more optimal solutions than insertion sort do exist. One optimization that we use in our Java implementation is to determine the insertion point into the list through *binary search*. For small values of k, further optimizations, in particular the usage of trees instead of a simple array does not lead to measurable performance gains

Origin of Early Termination Strategy

This strategy was already used in the implementation of [Weber et al., 1998, Weber, 2001]. It has been independently developed or discovered several times – mostly focusing on the particular case for kNN searches with $k = 1$ (search for the single best solution, no need to keep a ranked list). Early publications dating back to [Cheng et al., 1984] and [Bei and Gray, 1985]. It has also been described in [Keogh and Kasetty, 2002, p. 358f] and named *Early Abandon* in [Keogh et al., 2006, p. 884] and [Keogh et al., 2009, p. 615f].[81]

As visible in Pseudo Code Listing 10.18 – 10.20, the changes required are very limited and one may expect that this technique is used in most implementations. But as a matter of fact, even very widely used software like OpenCV[82] does not make use of it.

Relationship of Early Termination to Indexing

The Early Termination Strategy is just one optimization technique to *Link Computation and Ranking*. Also any range search or kNN search that exploits an index uses the underlying principal, that only those items are returned from the index that will be part of the final result. Opposed to indexing, Early Termination does not require any preprocessing of the data; it can be performed entirely at query time. Furthermore, as Early Termination is effective for individual distance computations, it does not require the distance measure to fulfill the *Triangle inequality* in Equation (5.11).

Indexing achieves performance improvements due to two desired effects: Reduction of I/O costs as not all features have to be read and reduction of CPU costs as not all distances have to be computed. Early Termination Strategy to reduce CPU costs while indexing and querying an index – as it was done already in the implementation of [Weber et al., 1998, Weber, 2001]. The impact of the Early Termination Strategy de-

as operations on small arrays are already performed quickly through the locality of data in memory and implicit removal of entries that are no longer needed by overwriting them with better values in the array.

[81] Unfortunately, when we introduced the name *Early Termination Strategy* to describe that the computation of the distance is terminated early in [Springmann et al., 2007a, Springmann and Schuldt, 2007, Springmann et al., 2008], there was no established named for this algorithmic optimization – at least none that we would have been aware of even at a time when it was used since about ten years in our group dating back to [Weber et al., 1998] and beyond. Even worse, there are some publications that use the term *Early Termination* name when retrieval stops early without waiting for the final results, e.g., in [Aggarwal et al., 1999, Anh et al., 2001] – therefore implement the approximation technique named *Early Stopping* in [Patella and Ciaccia, 2008]. Within this thesis, we continue to use the term *Early Termination Strategy* for the algorithmic optimization that is not an approximation, but returns the exact results. The term *Early Termination* is also used in [Barnes et al., 2009] in refer to the non-approximating technique similar to ours to improve the efficiency to determine the nearest neighbor field in PatchMatch.

[82] Open Source Computer Vision Library developed and supported by Intel, `http://opencv.willowgarage.com/wiki/`; the version current at the time of writing 2.3.1 does not make use of Early Termination in the matchers.cpp of features2d.
The same is true for distance measures and matching code used in SISAP Metric Spaces Library (`http://sisap.org/Metric_Space_Library.html`, tested version 1.3), FIRE (Flexible Image Retrieval Engine, `http://thomas.deselaers.de/fire/`, tested version 2.3), LIRE (`LuceneImageREtrieval`, `http://www.semanticmetadata.net/`, tested version 0.9), the SIFT reference implementation (`http://www.cs.ubc.ca/~lowe/keypoints/`, tested version 4), and VLFeat (`http://www.vlfeat.org`, tested version 0.9.13).

pends on the computational complexity of the used distance measure and how much distance computations can be avoided in general due to the computation of partial distances during ahead of query execution.

10.5.3 Multi-Threading with Emphasize on Shared Memory Environments

The work performed in query execution is highly parallelizable and early works mainly focussed on works for distributed environments, where the main memory is not shared between individual nodes in the network. A similar model can also be applied for a single node with multiple processing units when independent processes are started on each processing unit when no shared memory access is used, however, with significant less costs for interprocess communication. Multiple processing units in each single node are predominant in server environments and have become also very common for desktop computers and laptops during the last years due to the availability of multi-core CPUs. To exploit their full potential, individual computations should not be performed in isolation.

Feature Extraction in Parallel

Feature extraction is particularly easy to parallelize in such an environment with no shared memory as each image and each feature can get extracted independently. Such approaches have been described for clusters / network of workstations (NoW) in [Weber et al., 1999, Weber and Schek, 1999, Mlivoncic et al., 2004a] and on Grid infrastructures for SAPIR [Falchi et al., 2007] / CoPhIR [Bolettieri et al., 2009a] is described in [Bolettieri et al., 2009b].

Although features can get extracted independently, some optimizations are possible when several (or all) features of an image are extracted in a single task:

1. The image file only need to be transferred to a single node of the network.[83]

2. Open and therefore decode/decompress the image file only once.

3. Reuse transformations on the image, e.g., if several features rescale the image to a fixed input size, need the image in a particular color space, or will apply filters like edge detection.

4. Extract the features using different schemes to define regions like segmented regions, static regions, and the global features (cf. Chapter 5.1).

[83] In [Bolettieri et al., 2009b], feature extraction is performed this way: The extraction of all five used MPEG-7 features (Scalable Color, Color Structure, Color Layout, Edge Histogram, Homogeneous Texture) are extracted at a single crawling agent that has been assigned to download and process the image for a particular image-id. The individual extraction of each feature is performed by invoking the MPEG-7 XM software. In contrast, [Weber et al., 1999] describes the a system in which features of the same image may get extracted in parallel, thus reducing the time to extract all features for a single image to the time of the most complex feature if each feature gets extracted on a different node.

For instance, in Section 10.3.4, ARP and IDM features are both extracted from edge maps of the luminance channels of images scaled to 400×400 pixels: For ARP, the number of edge pixels are counted inside angular and radial partitions – and variations are used in the number of partitions as well as their placement to better support invariances. For IDM, the edge map gets scaled to a lower resolution of 32×32 pixels. While this feature extraction takes place for one image, another image can already get processed in on a different node/in a different thread.

Similarly, in Section 10.4.3, color and texture moments as well as SIFT keypoint descriptor get extracted. It is not necessary to open the image file more than once to extract all features as processing each feature separately would do. This approach will mainly reduce the I/O load. Furthermore, whenever there are several regions that share the same preprocessing by applying filters to the image, and only perform the later stages for particular global or static regions as it is common for color and texture moments. This will also reduce the CPU time needed to extract the features.

To exploit this functionality, the easiest approach is to preserve the in-memory-representation of the image until all features have been processed – which is only possible when the features of an image are not computed independently. Such a strategy maximizes throughput of *bulk feature extraction*, for instance, when a new collection is added to content management. The strategy is not ideal when low latency for a single image is desired. The latter is of main interest in interactive settings, in particular when a user just selected a query image or has drawn a sketch and waits for the results. As extracting the features has to finish before the first results are produced, this step should finish in as little time as possible. Extracting features of the same image in parallel as in [Weber et al., 1999] is the better choice for such a time-critical setting as it reduces the time the user has to wait to the duration of the time it takes to extract the most expensive feature rather than the duration to extract all features in sequence.

Distance Computation in Parallel

Also retrieval can be performed in parallel – by load-balancing individual queries to nodes in the network and also for individual queries by processing parts of the execution on different nodes as for instance in [Weber et al., 2000b, Weber et al., 2000a].

In general, it is easier to achieve near ideal speedup for range searches than for kNN searches, as the search for images within a range can be executed using only local knowledge on subsets of the overall collection and later combined into a single result: Using m nodes, each node handles a non-overlapping subset $Docs_m$ of $Docs$ independently. Let $RES_{RangeSearch,i}$ denote the result of the subset $Docs_i$ then:

$$RES_{RangeSearch} = \cup_{i=1}^{m} RES_{RangeSearch,i}. \tag{10.16}$$

For a kNN search, using only local knowledge, each node has to generate the k nearest neighbors, so the aggregated list of m nodes each hosting a subset of the collection will contain $m \times k$ results – even if the user requested only the k best documents. Using the extreme case where each node hosts a single document ($m = |Docs|$) the difference can be illustrate easily:

- In a range search, each node can make use locally of optimization techniques like for instance the Early Termination strategy or approximations to speed up the distance computation.

- In a *k*NN search the nodes will not be able to use any optimization as they will have to deliver at least a single documents as it is the nearest neighbor. In such an extreme case with as many nodes as documents in the collection also the communication overhead between the nodes can easily exceed the time saved by using all the nodes. Also, optimizations that let individual nodes in *k*NN searches will add communication overhead as the nodes need to know how good other results are in order to not waste effort on computing the distance on documents that will not end up among the top *k* results.

Therefore it is essential for *k*NN search to rely not only to improve search speed by adding more nodes to the execution, but also speed up the execution on individual nodes. In [Terboven et al., 2006] an implementation is proposed that uses load-balancing of queries and OpenMP[84] to parallelize distance computation on multiple threads. This implementation requires all features of the images in the collection to be loaded into main memory in the form of matrix before execution can start and each thread operates in complete isolation without global knowledge of the best seen neighbors.

This situation can be improved significantly by using a dispatcher to distribute unit of work to individual threads and collecting results to update the list of *k* best results that have been found so far. The benefits of such an approach are:

1. *Smaller memory footprint*: As features do not need to remain in a matrix in memory during the entire computation, but only for the duration of the computation of a particular distance between features, less memory is occupied. This allows to compute distances for collections where the entire features of all images would exceed the available memory and also allow to use the remaining memory for other optimizations (such as Section 10.5.4).[85]

2. *Immediate start of distance computation*: The dispatcher handles the loading of the features to memory, therefore the features do not have to be read before execution starts – and this can be used to achieve parallelism between I/O heavy operations and computation.

3. *Control over order in which features are loaded*: The dispatcher remains in control in which order distances to features get computed, therefore can enforce optimal order w.r.t. I/O capabilities. For features read from traditional disks with rotating platters, sequential reads are usually much faster than random reads as the head does not need to seek the correct position. Additionally, and even when features are stored on other storage media like solid state disks (SSD) that does not have

[84] http://openmp.org

[85] Of course, the approach in[Terboven et al., 2006] could also be modified to load features only when they are needed. However, as no explicit dispatcher controls the order in which the features are requested, this approach wouldn't lead to the same benefits described in item 3.

such high penalties for random reads, feature order on storage can be important as subsequent features may get stored in the same logical block of storage. Therefore sequential access will frequently lead to a situation in which the bytes of the next feature are already in the cache of the storage controller or already in main memory through explicit prefetching of following blocks. Furthermore, sequential reads allow simple integration of compression mechanism.[86]

4. *Integration with other optimization techniques*: After each completed unit of work, the dispatcher takes the result to update the list of k best results in the same way as in Line 13 of Pseudo Code Listing 10.20. This knowledge can be passed on to the next unit of work, e.g., the dispatcher can retrieve the threshold t as in Line 14 which is needed for achieve early termination inside distance computation. Furthermore, the dispatcher may also distribute work across multiple nodes as in [Weber et al., 2000a], therefore seamlessly integrating local multi-threading with distributed computing.

10.5.4 Caching Data in Main Memory

During the last years, not only the processing capabilities of common computers have increased, but also the amount of available RAM. Today it is not uncommon that desktop PCs and laptops have more than 2 GB of installed with predominance of 32-bit operation system being the last major blockage to exceed the 4 GB limit. Servers are much more likely to be using 64-bit operating systems and for even lower range servers it very common, that they are equipped with more than 4 GB of RAM. Even smartphones are now equipped with 512 MB to 1 GB of RAM.

This increase of RAM has led to increased interest in in-memory techniques to in the field of databases, e.g., with H-Store [Stonebraker et al., 2007] and TimesTen[87]. For performing similarity search based on perceptual features, key considerations therefore should be: How many features can easily fit into main memory? How can the available RAM be used effectively if not all features fit into memory?

Caching of Features

When considering personal image collections with tens of thousand to hundreds of thousand images, this increase of available main memory should not be underestimated: Using compact feature descriptors like the MPEG-7 visual descriptors [Sikora, 2001], or color or texture moments extracted globally or with five or nine static regions, or ARP with 8 angular and 4 radial partitions, all features of a single type for the entire collection easily fit into main memory. Table 10.5 shows the approximate number of images for which the features of a single type fit in one gigabyte of memory. This estimation does not assume any compression technique or additional quantization to shrink memory requirement per feature; however, real-world numbers

[86] Compression in the same way as it is done in column-oriented database system, e.g., [Abadi et al., 2006].

[87] `http://www.oracle.com/us/products/database/timesten/`

Table 10.5: Approximate number of images for which extracted features fit in 1 GB RAM

Feature Parameters	Storage Requirement	Bytes/Image	Images/GB
MPEG-7: EHD [Sikora, 2001, p. 699]			
Edge Histogram Descriptor	240 bits per descriptor	30	35'800'000
Color Moments [Stricker and Orengo, 1995, Stricker and Dimai, 1996]			
Global	9 floats (single prec.)	36	29'800'000
5 Fuzzy Regions	9 floats $\times$ 5	180	6'000'000
Gabor Texture Moments [Manjunath and Ma, 1996]			
3 Scales, 5 Orientations	30 floats (single prec.)	120	8'900'000
9×9 Overlapping Rectangles	30 floats $\times$ 45	9'720	110'000
Angular Radial Partitioning (ARP) [Chalechale et al., 2004, Springmann et al., 2010a]			
8 angular, 4 radial partitions	1 short int $\times$ 32	64	16'800'000
Rotation Invariance (FFT)	1 float $\times$ 32	128	8'400'000
Scale and Translation Inv.	32 floats $\times$ 8 regions	1'024	1'048'000
Edge Detection (β) Inv.	32 floats $\times$ 8 β	1'024	1'048'000
Rot. + Scale/Trans. + β Inv.	32 floats $\times$ 8 $\times$ 8	8'192	131'000
Angular Radial Color Moments (ARCM) [Giangreco, 2010]			
8 angular, 4 radial partitions	9 floats $\times$ 32	1'152	932'000
Rotation Invariance (FFT)	9 floats $\times$ 32	1'152	932'000
Scale and Translation Inv.	$\times$ 8 regions	9'216	116'000
Image Distortion Model (IDM) [Keysers et al., 2007, Springmann et al., 2008]			
32×32 pixels, just edges	1024 bytes	1'024	1'048'000
intensities + combined Sobel	1024 bytes $\times$ 2	2'048	524'000
Scale-invariant Feature Transform (SIFT) [Lowe, 2004]			
single Keypoint Descriptor	128 floats (single prec.)	512	2'000'000
1'482 Keypoints per Image	128 floats $\times$ 1'482	758'784	1'415

of simple implementations may stay a little below these values as they do not take into account any overhead for in-memory-representation, in particular leave out the space required for the image ID. The latter, however, hardly gets crucial for collections less than millions of documents. In case of keypoint descriptors, also keypoint location, orientation, and scale would require additional memory. Even without this information, 128-dimensional SIFT descriptors with 1'428 descriptors per image on average as in Section 10.4.3 would be the only feature in Table 10.5 where for a collection of 100'000 images not all features would fit in one gigabyte of memory; for some of the other features even features of collections with more than a million images fit into 1 GB.

This implies that –for smaller collections like personal image collections– it is likely that all features required to execute a single query based on a single feature fit easily into main memory if we leave aside keypoint descriptors. If there is not just a single search, but several subsequent searches, the features should get read once and remain in memory. This will deliver better performance than using even sophisticated disc-based tree index structures, which have been proven to be unable to outperform a

Table 10.6: Approximate time to read extracted features from a file of 1 GB

File Source	Sequential Read Speed	Required Time
Slower 2.5 Inch HDD	10 MB/s	102.4 s
External HDD using USB2.0	30 MB/s	34.5 s
Fast 3.5 Inch HDD	150 MB/s	6.8 s
Fast SSD	250 MB/s	4.1 s
RAID bound by SATA limits	6 GBit/s	1.3 s

simple sequential scan for high-dimensional feature spaces with more than 10 dimensions [Weber et al., 1998] anyway. Furthermore, there's also little benefit in optimizing the time it takes to perform a sequential scan on disk: A file containing all feature of one type for images in the collection will also not exceed one gigabyte of storage for the same reason as they fit in main memory. Reading one gigabyte from a disk will take a few seconds as shown in the estimations in Table 10.6 based on the sequential read performance that can commonly be achieved, with the estimated times proportionally shrinking and growing for other file sizes. The reading of required features can already be performed during the startup of the application or while the user selects a suitable image as an example or draws a sketch. Therefore it does not necessarily add to or be the lower bound for the time the user has to wait for the results after issuing a query.

As soon as the features loaded to main memory, execution speed of search is commonly no longer I/O but CPU bound. For features for which the distance computation is more complex, e.g., IDM and matching images based on keypoints, this is with current technology even CPU-bound when features are read from disk. For those searches that are always CPU-bound and have features that require much space, the best strategy is to focus on speeding up the computation and simply prefetch just enough features to avoid that the CPU remains idle waiting for the next feature to process.

For some of the features, the features for personal image collections will hardly ever exceed one gigabyte, but usually just occupy a small fraction of that. But not all searches involve just a single feature, e.g., complex searches may use an aggregate the distance for several features as in Chapter 5.3.4. In such cases, multiple files containing the features have to be read. Similar situation occurs when heuristics to ARP are enabled to achieve better invariance to rotation, changes of scale, translation, and issues of edge detection all at the same time as already mentioned in Table 10.5. There can be two different cases depending on whether all features fit simultaneously into memory:

- As long as all features still fit in main memory and several searches are expected to occur one after another, features should remain in main memory as long as it is not needing for anything else. If memory is needed for a feature that is not yet present in main memory, a simple Least Recently Used (LRU) cache replacement strategy can be applied.

- If not all features needed to satisfy a single query fit into main memory, a LRU strategy does not offer any benefit as any cached item will get replaced before

the next access will occur. In such a situation, prefetching of next features to access is more appropriate and towards the end of the search, the system may decide to keep the last used features in memory.[88] For the search in general, such a search could benefit strongly from feature compression or elaborate index structures that reduce the main memory needs, e.g., the vector approximation file [Weber et al., 1998]: It uses more compact approximations in a first stage, therefore can hold more (approximated) features in memory. In many cases this will be sufficient to access the disk only in the second stage for a limited number of candidates and therefore reduce the number of blocks that have to be read from disk. Another possibility of course would be to use more nodes in a network: [Weber et al., 2000a] has shown that exploiting the main memory of more nodes as caches for the features of sub collections greatly improves search performance.

The last observations are of course also occurring for huge collections, in particular collections at web-scale, for which even a single server might not be sufficient. In such settings, distribution and the use of approximate similarity search techniques as mentioned in Section 10.5.1 on page 324 are highly desirable with locality-sensitive hashing (LSH) [Datar et al., 2004] as one currently popular example – but certainly leave the scope of caching.

Caching of Images

When execution is already performed quickly, in highly interactive image search task, loading images from disk can become the bottleneck in user experience. To be able to present results quickly, previously retrieved thumbnail images are held in an LRU cache (similar to the features), such that only slightly modified searches will display results almost instantly while newly retrieved items are loaded in a background thread from disk. For web-based retrieval systems, such a behavior is usually already built inside the browser. The only aspect the image retrieval system on the server side has to implement then is using thumbnail URLs that remain stable between different searches and to return appropriate HTTP status response 304 “Not Modified” when being asked for an image with if-modified-since field [Fielding et al., 1999, p. 63]. For other implementations, in particular stand-alone software, this has to be implemented by the developers.

The same certainly applies also for the images in full resolution that the user selected from search results. And aside from the presentation of search result images, caching of images can also improve feature extraction if it is not already optimized to read an image exactly once (as described in Section 10.5.3). Furthermore, caching of requested metadata in content management is generally also beneficial (which frequently used database management systems and text-retrieval engines will perform anyway).

[88] In database terminology, what will be performed with several searches will be looping sequential accesses – for which a Most Recently Used (MRU) cache replacement strategy is optimal [Chou and DeWitt, 1985]. Keeping the last features in memory implements already MRU.

10.5.5 Impact of Optimization on Medical Image Classification

In Section 10.2 we described our approach to medical image classification that is based on earlier work that applied IDM for medical image classification [Thies et al., 2005, Güld et al., 2005, Keysers et al., 2007]. According [Thies et al., 2005], the execution of a single query in a collection of 8'728 images with a local context of 3×3 pixels and a warp range of 5×5 pixels takes about 5 minutes on a standard Pentium PC with 2.4 GHz – which is clearly not even close to an interactive response time (as it would be required for many applications in the area of content-based image retrieval) and also comparably slow for image classification.

Therefore [Keysers et al., 2004] proposed to use a sieve function inspired by [Simard, 1993] to reduce the number of expensive IDM computations to a smaller subset of images, which have been pre-selected using the less expensive Euclidean distance. This approximate similarity search approach, allowed them to reduce the time for a single query to 18.7 seconds [Thies et al., 2005], but this is only possible with some degradation in retrieval quality. Using such an approach therefore requires to find a good tradeoff between speed and quality. It also does not scale ideally with state-of-the-art hardware that allows for many concurrent threads, therefore achieving less speedup at the cost of the same degradation of retrieval quality.

Commonly, retrieval speed can be improved by using an appropriate index structure. However, such techniques like the R-Tree [Guttman, 1984], the R*-Tree [Beckmann et al., 1990], X-Tree [Berchtold et al., 1996], M-Tree [Ciaccia et al., 1997], or VA-File [Weber et al., 1998] require either a fixed feature vector with a fixed number of dimensions or at least a metric distance function. IDM does not satisfy either of these properties: If the aspect ratio of images is preserved as described in Section 10.2.2, the number of values in the extracted features depend on the image. Furthermore, as optimization for the lowest distance is performed over one query feature, it cannot be guaranteed that the distance measure fulfills Equation (5.10) (Symmetry) and therefore may not satisfy all required properties to be a metric.

Applying Early Termination

We propose an approach that increases the speed of IDM without any negative impact on the retrieval quality. In our experiments, we use the parameters for IDM as in [Thies et al., 2005, Keysers et al., 2007], but apply an early termination condition to the individual distance computations. This leads to an execution time of 16 seconds per query over the entire image collection on similar hardware without any degradation of quality.

For the classification step, only the ordering and distance of the k nearest neighbors will be used. Therefore the exact distance of any image with rank $> k$ is unimportant and we can approximate the distance to ∞ as it is done in Equation (10.18).

On average, this function returns after only a fraction of all pixels have been processed. With increasing number of images n in the collection, the absolute number of pixels processed is increasing, but the relative number (i.e., compared to the computation without early termination condition) is decreasing. This property is illustrated in Figure 10.74 for IDM and, in addition, also the Euclidian distance. If the entire train-

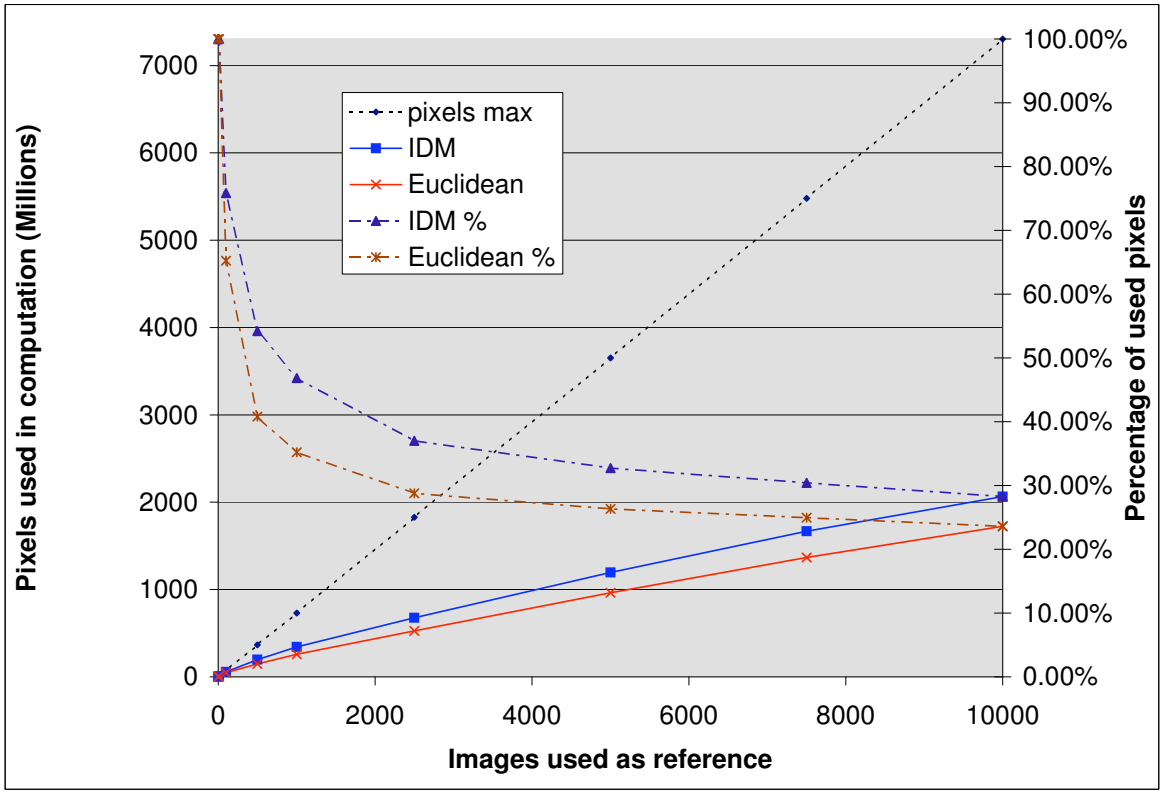


Figure 10.74: Impact of Early Termination Strategy on Euclidean Distance and IDM: Absolute and relative number of pixels for run with 1'000 query images with increasing number of reference images and $k = 5$

ing data is used as reference and no early termination based on the maximum sum is applied, a total of 7.30 billion pixels need to be compared between the 10'000 reference images and the 1'000 query images for an entire run. Using the early termination strategy and $k = 5$, the number reduces to 2.06 billion pixels for IDM (28.27%) and 1.73 billion pixels for the Euclidean distance (23.60%). Notice that already for 1'000 images, more than half of the computations could be saved and the number of pixels with regard to the reference pixels increases sublinearly. For $k = 1$, the early termination strategy reduces the processed pixels of IDM even to 1.58 billion (21.73%) and 1.38 billion for the Euclidean distance (18.84%).

Since the algorithm is orthogonal to the used pixel distance computation function, it can be applied to IDM with or without local context as well as the Euclidean distance that might be used in a sieve function; in particular using the Euclidean distance as a sieve function with a cutoff c of 500 nearest neighbors was proposed in [Thies et al., 2005] (cf. Section 10.2.5 on page 233). This cutoff is significantly larger than k, therefore the early termination reduces the processed pixels only to 4.13 billion (56.40%). The subsequent IDM computes out of the remaining $c = 500$ images the $k = 5$ nearest neighbors, therefore processing 0.19 billion pixels (51.90% of the 0.73 billion to compare the 1'000 query images with 500 filtered images or, in relation to the entire collection, 2.60%).

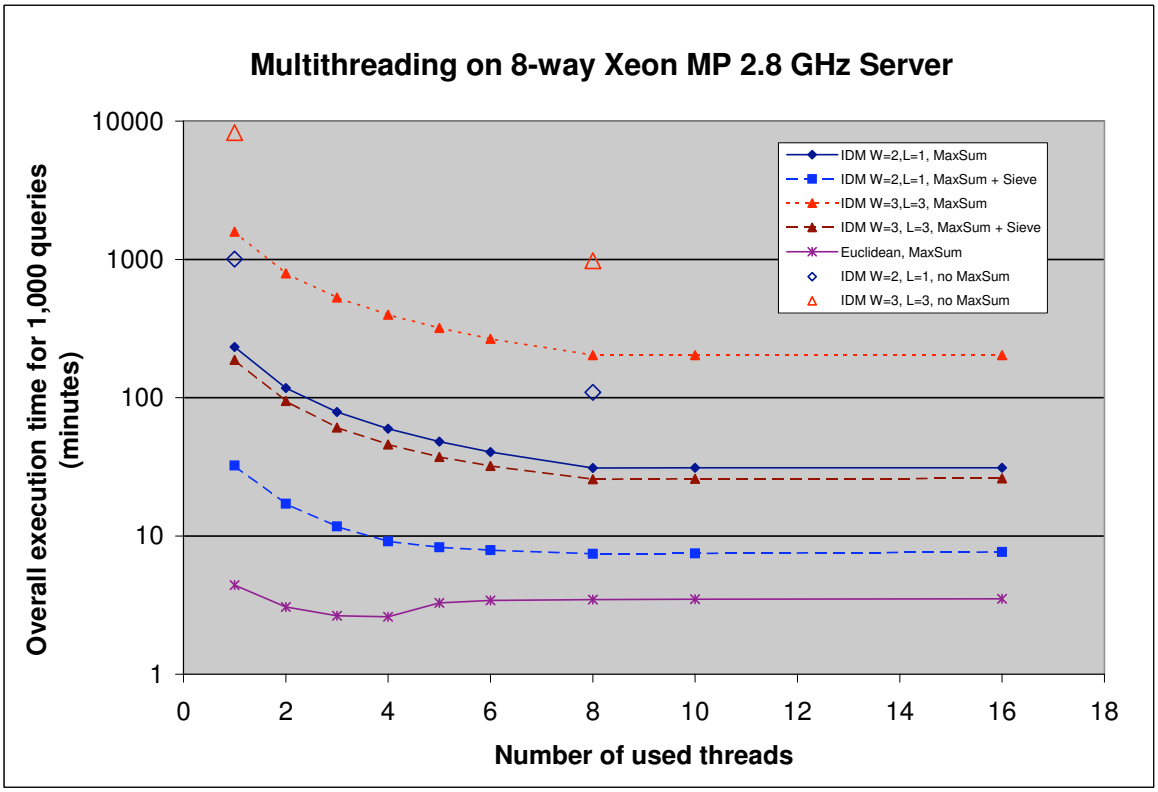


Figure 10.75: Runtimes for IDM on 8-way server for increasing number of threads

In order to keep the size of allocated main memory low, the threshold on the distance is also used to filter results on Line 11 in Pseudo Code Listing 10.20: Only results with a distance less than the threshold will be kept and added to the list of results in main memory. Existing entries with distances greater than the threshold are dropped during insertion of new items. This leads to a fixed length of k for the list, in which the values are kept in sorted order.

If not only the distance, but also the identifier of the image that achieved the value is stored in the entry, the list also contains the final result of the kNN search as soon as all feature vectors have been processed. As an intended side effect, this approach sorts all kNN in $O(n \times k)$ with k being a comparably small constant factor and a single pass over n. For the sieve function, one additional list of size c is used.

Impact of Multithreading on IDM

Within the last years, multi-core CPUs became very popular and affordable. Therefore it becomes more and more important to design applications in a way that they can use multiple threads in order to utilize all the capabilities provided by current hardware.

As described in Section 10.5.3 our implementation uses a dispatcher that takes the computed result and updates the threshold for the next unit of work. Through this we could achieve almost linear speedup on multi-core CPUs, since IDM is much more CPU-bound than I/O-bound. Additionally, accesses to the disk get serialized through

the dispatcher and therefore the concurrent execution does not lead to slow concurrent disk accesses.

This approach is similar to the "parallelization on second level" described in [Terboven et al., 2006], except that we did not use OpenMP but plain Java threads and explicitly take into account the read order from disk to enforce use of sequential reads whenever possible.

Figure 10.75 displays the execution time, using logarithmic scale, with respect to the number of threads on a server with 8 physical processors. All distance computations using IDM scale very well with the number of processors. Saturation is reached with one thread per CPU. Offering more threads then available CPUs does not harm the performance, except in the case of the simple Euclidean distance. In that case, the overhead through multithreading for context switches etc. reduced the performance as the comparison with all 10'000 reference images was performed so fast, that some of the concurrent threads trying to submit results to the dispatcher and get new workloads blocked each other. Synchronization is unavoidable to fully exploit the maximum sum, otherwise each thread could compute its own top k results, thus having an effective value for the maximum sum of about $k \times$number of threads. Filtering with a sieve function scaled well until six threads, then the Euclidean distance used during filtering limits the gained speedup. All runs depicted with lines used the early termination strategy with a maximum sum criterion. Individual runs were performed without the early termination strategy with a single and eight threads.

Query Execution Time Measurements

All experiments in Table 10.2 and Table 10.7 are based on IBM xSeries 445 with 8 Xeon MP CPUs at 2.8 GHz. The execution times have been measured for entire runs on 1'000 images. If not stated otherwise, features were cached in main memory. Each query image has been processed in strict batch order, that is, one after another by first extracting the features of the image, then performing the nearest neighbor search and finally waiting until all classifiers (if more than one) have finished. We did not apply further optimizations for the throughput of the entire run such as extracting the features of the next image while the classifier for the last image is still running. Therefore one can simply divide the runtime of the entire run by 1'000 to get the average execution time per query when performed interactively.

The execution times are displayed in Table 10.7 for a single query and for the entire run and the label refers to the label in the qualitative evaluation the runs in Table 10.2 on page 233 which have been submitted to the ImageCLEF automatic medical image annotation task [Deselaers et al., 2008a] benchmark competition of 2007. If parameters differ only in the number of the used classifier, e.g. runs (23), (24), (25), the actual retrieval has only be performed for the largest k, here $k = 5$, and the other submissions where generated by using only the required first k of the available nearest neighbors for classification, thus avoiding time-consuming recalculation of subsets of a set, that has already been determined. Therefore only a single execution time is listed for such runs and the label column shows the aggregated form, e.g. (23,24,25). A '*' indicates that the run achieves exactly the same result as the run with the label without '*', but

Table 10.7: Execution times on average for single query and entire run

Parameters	CPUs	1 Query [s]	Run	Labels
$W = 3$ $L = 3$ -MaxSum	1	495.12s	5d 16h	(*19)
$W = 2$ $L = 1$ -MaxSum	1	60.36s	16h 46m	(*d)
$W = 3$ $L = 3$ -MaxSum	8	58.68s	16h 18m	(*19)
$W = 2$ $L = 1$ $k = 5$	1	13.98s	3h 53m	(*d)
$W = 3$ $L = 3$ 369	8	13.26s	3h 41m	(19,e)
$W = 3$ $L = 2$ 4816	8	9.48s	2h 38m	(20,21,22)
$W = 3$ $L = 2$ 369	8	8.58s	2h 23m	(23,24,25)
$W = 2$ $L = 1$ -MaxSum	8	7.14s	1h 59m	(*d)
$W = 2$ $L = 1$ $k = 5$ D	8	2.81s	46m 53s	(*d)
$W = 2$ $L = 1$ + Sieve 500L1 D	8	2.51s	41m 50s	(*b)
$W = 2$ $L = 1$ + Sieve 500L1	1	1.94s	32m 18s	(*b)
$W = 2$ $L = 1$ $k = 5$	8	1.81s	30m 12s	(d)
$W = 2$ $L = 1$ $k = 1$	8	1.38s	23m 2s	(c)
$W = 2$ $L = 1$ + Sieve 500L1	8	0.41s	6m 52s	(b)
$W = 2$ $L = 1$ + Sieve 500L2	8	0.40s	6m 39s	(a)

with different execution time. '-MaxSum' means that no early termination strategy was used. A 'D' as the last parameter means, that the features were read sequentially from disk and not cached in memory, therefore requiring an entire scan for each of the 1'000 query images.

For our runs (20,21,22) with $l = 2$, a single query took on average less than 9.5 seconds (2h 38m for the entire run) and 13.3 seconds for our best run (19) with $l = 3$ (3h 41m in total). For comparison: The exact same result was computed in 16h 18m on the same server when no early termination based on the maximum sum was used. This long duration even increased to entire 5 days, 17h and 32m on the same machine when we limited the number of used threads to a single one – as it was the case in our starting point of the implementation. This means that our optimizations achieved a speedup of 4.42 and 37.34, respectively.

We also performed runs with the parameters proposed in [Keysers et al., 2007]: IDM with a deformation in 5×5 pixels ($w = 2$) and a local context of 3×3 pixels ($l = 1$) and the nearest neighbor decision rule ($k = 1$). On a standard Pentium 4 PC with 2.4 GHz, this run finished within 4 hours and 28 minutes (16.0s per query) – without any sieve function. The same run was finished on the 8-way Xeon within 23 minutes and 2 seconds (less than 1.5 seconds per query). When we turned off just the early termination, the durations increased to 19 hours 21 minutes on this P4 (69.77 seconds per query, factor 4.33) and 1 hour 59 minutes on the 8-way Xeon MP server (7.14 seconds per query, factor 4.86).

For comparison, the labels (a) and (b) in Table 10.7 contain the times when a sieve function with a cutoff after 500 nearest neighbors similar to [Thies et al., 2005] is used. The sieve function uses a modified version of the Euclidean distance that uses the same

method for scaling images that differ in height or width and also both layers (intensities and gradients) and the same threshold for maximum distance of distance created by a single pixel. By this, it approximates the distance of IDM as close as possible. We also replaced this distance function with one based on the Manhattan distance, which is not that sensitive to single occurrences of extreme values, that IDM would avoid through displacements anyway. This resulted in fewer cases where some of the nearest neighbors were not among the preselected 500 images and therefore less degradation of results.

When the early termination strategy is used with the sieve function, the entire run took no longer than 7 minutes on the 8-way server (0.42 seconds per query) and therefore achieved a speedup of 3.29 compared to the direct IDM computation. Without the early termination strategy, the run took 11 minutes and 56 seconds (0.71s per query) – whereas IDM would last more than ten times as long with 1h 59m (7.14s per query). This shows on one hand that the early termination strategy reduces also the time required by the sieve function significantly, but on the other hand it reduces also the need for it. When features are not cached but read from disk for each query, there was very little speedup achieved by the sieve function – 2.51s per query, 41m 50s in total compared to 2.81s per query, 46m 53s.

When only a single thread with early termination strategy was used, the sieve function achieved a speedup of 7.21 (2.81s compared to 13.98s per query). Similarly, on the P4, the same run took 33 minutes 17 seconds (2.0 seconds per query, speedup 8.12). Since both are significantly higher speedups than 3.29 which was achieved when all 8 CPUs on the server are used, this shows that fast distance computations cannot equally benefit from multithreading and therefore the sieve function does not scale that well on state-of-the-art hardware while still demanding some degradation of retrieval quality.

Conclusion for Medical Image Classification

As described in in Section 10.2.5 on 232, increasing the warp range from 2 to 3 and using a local context to an area of 7×7 pixels instead of 3×3 significantly improve the retrieval quality of IDM. Modifications of the used *k*NN classifier can further improve the quality, but in all our experiments only to a much smaller extent.

For being able to perform such experiments within reasonable time, we propose an early termination strategy, which has proven to successfully speed up the expensive Image Distortion Model by factors in the range of 4.33 to 4.86 in our experiments. We could reduce the computation time to perform a single query in the collection of 10'000 reference images to approximately 16 seconds on a standard Pentium 4 PC. Making proper use of multithreading, we could even perform a single query within 1.5 seconds on an 8-way Xeon MP server. Even with the increased range, we could perform this on the Xeon server in maximum 13.3 seconds per query. Caching features in main memory turned out to be very effective as a single set of features is used, for which extracted features for all 10'000 reference images fits easily into main memory even of "not up-to-date" desktop PCs.

10.5.6 Impact of Optimization on Sketch-Based Known Image Search

As described in Section 10.3, we reused for our implementation of the QbS prototype for sketch-based known image search the same implementation as for the medical image classification with a slightly modified set of perceptual features and a different interaction environment.

Perceptual Features and Cache Management

ARP is used as a compact, fast way to retrieve images as described in Section 10.3.4 and IDM is used when more thorough comparison of sketch and images is needed as described in Section 10.3.4. For both features, only edges detected in the lightness channel of the image in CIELAB are used. The edge detection algorithm is a variant of the Canny edge detector [Canny, 1986] as described in [Chalechale et al., 2004] and will therefore result in binary decisions edge pixel / non-edge pixel rather than the edge intensity determined with a Sobel filter that was used in Section 10.2.2. While it is very unlikely that two different medical images will ever have not only similar, but identical edge intensities even in some areas of an image, this gets far more likely for binary edge maps. Therefore we adapted our implementation of IDM to not only use the early termination when the maximum sum was reached, but also not to attempt to further minimize the distance for a single pixel and its local context when already an exact match was found.

Without this modification, a straight forward implementation of Equation (10.6) would further shift the local context area inside the warp range – even though the distance of 0 of the exact match is the already the optimal value that could ever be achieved. The impact of this minor optimization depends on the combination of warp range w and the local context range lc: The greatest impact is achieved for great warp ranges with small local context as aborting iterating through big warp ranges saves more computations and small local contexts increase the probability of finding exact matches. For the extreme case of $w = 0$ (or any Minkowski norm, e.g., L_2), this added check can never improve the performance as their will only a single value gets computed within this narrow warp range. For another extreme case of $lc = 0$ (no local context, just compare individual pixels) the probability of finding an exact match is very high – highest, for binary edge maps, but also fairly high for edge intensities with at most 256 possible values – therefore even medical image annotation could benefit from this optimization. However, as explained in Section 10.2.5 on page 229, $lc = 0$ delivers very poor results – even worse than the simple Euclidean distance L_2 and for $lc > 0$, the probability of finding any exact match tends quickly towards zero; therefore making this optimization only beneficial for the use with edge maps in sketch-based searches or distance measures that do not deliver good results for medical annotation.

The edge detection depends on the value β as illustrated in Figure 10.17 and there is no single value that would be ideal for all searches. We empirically selected the eight values $\{2, 5, 8, 10, 12, 15, 20, 25\}$ as described in Section 10.3.5 as a reasonable compromise between allowing fine-grained selection whether only prominent or also fine edges have been detected (and drawn by the user in the sketch, respectively) and having to extract at insertion time and load at query time enormous amounts of data. Because:

Depending on what the user selects as the range of β, only a single, some, or all features will have to be loaded to execute the query. Additionally, to support more variances with ARP, we also extracted the features from eight subregions of the original image's edge map shown in Figure 10.18 and decide at query time depending on the selected option, which feature files need to be considered.[89]

When using color for sketch-based searches as described in Section 10.3.6, there is no need for edge detection and also no need to provide invariance for that. But the ARCM features keep 9 values instead of a single one in ARP per partition, and the heuristics to add scale and translation invariance will still result in the extraction from eight region subimages. As shown in Figure 10.34 on page 267, best results are achieved when ARCM and ARP are used in combination.

There are plenty of different combinations of features and corresponding files on disk that will become relevant depending on the options that the user selects for a search. Therefore –and also due to the fact that the MIRFLICKR-25000 dataset contains 2.5 times as many images as the training set for the ImageCLEF automatic medical image annotation task– it is no longer that simple to keep all features in main memory and memory management through an adequate cache replacement strategy gains significantly greater importance. Our implementation uses the LRU and prefetching strategies that have been detailed in Section 10.5.4 on page 336.

Interactive Usage of the System

In case of the medical image classification, the task requires the system to act autonomously – hence the name *automatic* medical image annotation task. For this it was not necessary to ever present the images that have been determines as the nearest neighbors: All that was needed was the class label associated to them, making the result usage of the task entirely representation-oriented as defined in Chapter 2.4.1.[90]

In an interactive known image search, restricting on the metadata of the results will clearly not be sufficient. Also the number of nearest neighbors k to display should not be as low 5 as used in the medical annotation case, but provide enough results to let the user browse and find the sought item without having to modify the sketch and search options too many times. $k = 20$ might be sufficient to fill a single screen with results, but in order to let the user scroll through several screens, we found k between 40 and 200 more appropriate for most searches as this turned out to be the number of images where the user would give up browsing for the sought image and either retry the search

[89] Please keep in mind that not all queries require or even benefit from enabling invariance options. Only if the sketch deviates significantly from the sought image, invariance is needed – if more options than needed are enabled, this relaxes the matching tolerance to a degree were other images than the sought image may now appear more similar than the sought image and therefore will degrade retrieval performance. Therefore some searches benefit from added invariance, other do not – and the likeliness if this is the case can only be determined when both, the user-drawn sketch and the sought image is known. Such knowledge is usually restricted to the user. In case of doubt, the user can try out several searches with different options enabled to find the sought image, but simply enabling all options is commonly a bad approach. See Section 10.3.5 for details.

[90] The only reason to show the found nearest neighbors would be to let the user visually assess the quality of the results and determined class label – mainly for cross-validation, development, and debugging purposes.

with a modified query or end the task unsuccessfully. Loading 200 thumbnail images from disk does require some time and therefore is performed in a background task in our implementation, but as searches that are only slightly modified will have a large overlap between the result images, the thumbnail loading is also using an LRU cache as described in Section 10.5.4 to avoid reloading images in subsequent searches.

Another aspect that only occurs in tasks with user interaction is the ability to define areas inside the sketch that should receive less impact or even be ignored as the user cannot remember the content of that area (cf. *unknown areas* in Chapter 2.3.3 and Section 10.3.4): When the optional *background invariance* is enabled in our implementation, any area that was left entirely empty will get ignored by assigning a weight of zero to that region in the weighted distance measure. For simple distance measures as weighted Manhattan distance used for ARP for such cases, a minor improvement can be achieved by skipping the computation of the absolute difference between feature vectors at positions were the weight is zero – and therefore this difference would not contribute anyway to the overall distance. For more complex distance measures as IDM, the same minor modification reduces the computational complexity almost proportionally to the area left empty inside the image.

Typical Query Execution Times using ARP

To get realistic results for the retrieval times as the end user will perceive them, all time measurements have been performed on the Lenovo X200t Tablet PC that was also used for the acquisition of the sketches. It has an Intel Core 2 Duo L9400 CPU –therefore two cores running at 1.86 GHz– 2 GB of RAM, a regular internal 2.5" HDD and is running Windows XP Professional Tablet PC Edition 2005 with Service Pack 3 and multithreading is performed in the same way as for the automatic classification of images.

Due to the compactness of the ARP features and the early termination strategy, common times for the plain distance computation for up to 200 nearest neighbors are between 15 and 40 milliseconds for a single value of β. It does not exceed 500 milliseconds even for big ranges of β and using many image regions for invariances as long as features are accessed from cache in main memory. That latter is frequently the case in such a highly interactive task as searching for known images through a user-drawn sketch. In particular, the next search will find all required features in the cache when the last search execution could fit all required features into main memory and the user modified only the sketch, e.g., just added some new edges, removed edges about which the user feels uncertain, or moves part or the entire sketch. Only when the user also changes the search options, e.g., enables some additional invariance option, new features will get loaded – and in many settings, the new set of features will overlap with some features that were used for the last search, therefore some features found in cache are still of interest while other features get prefetched from disk.

As soon as features cannot be read from cache, e.g., directly after the start of the application, the search is I/O bound. Each individual ARP feature file consumes between 3 MB (8/4 partitions) and 13 MB (16/8 partitions) and it takes on average between 0.5

and 2.5 seconds to load it from disk[91]. This time certainly could be optimized, for instance by adopting an appropriate high-dimensional index structure. However, it only is beneficial when additional features have to be read – for subsequent searches, most features can usually be read from the cache and therefore, this only occasionally affects the users' experience during usual query sessions. Also, due to the effect of caching of image thumbnails and loading of additional thumbnails in background, result presentation for subsequent and only slightly modified searches appear to the user usually without noticeable delay. Thus, for known image searches using ARP, ARCP, and EHD, the user will not feel interrupted in performing the search task when pressing the search buttons and getting results usually in less than 40 ms, for more complex purely visual searches still in less than 500 ms. Adding keywords to the search will reduce the search time as Lucene provides fast inverted indexes to determine the set of images that match the keyword and similarity computation is restricted to these images.

Typical Query Execution Times using IDM

The features for IDM are considerably larger than for ARP and the distance measure case much higher computational complexity. The time for performing the search is CPU bound, as minimizing the distance within the warp range is costly, in particular when a large local context has to be considered. So even when performing search with a single value of β and a moderate size with warp range of 2 and a local context of 1, search takes commonly 10 to 15 seconds even when features are cached in memory. This is three orders of magnitude slower than simple ARP searches.

For a search with IDM and a very high tolerance for translations with a warp range of 5 and a local context of 3, search may take more than 5 minutes, which is certainly not acceptable for interactive use. Even high-dimensional index structures cannot resolve this issue easily as the image distortion model is not a metric distance measure and uses variable length features with –in our case– a maximum dimensionality of 1024.

The easiest "solution" certainly is to switch to a faster machine. The two-processors quad-core Nehalem server that we used for batch evaluations finishes such searches in 20 to 50 seconds. Such an approach would certainly be appropriate also for collections of much bigger size instead of the 25'000 images that was used to simulate a personal image collection.

Typical Query Execution Time using IDM and Filter

Another, in most cases more appropriate approach, can be to use IDM only together with a filter. It can either be used in combination with a keyword search or in a two-step approach to search and re-rank results generated by ARP. The latter implements a strategy falling into the category of approximate similarity search that reduces the comparisons in the original retrieval space (*RC* in [Patella and Ciaccia, 2008]) through selecting candidates in a simpler retrieval space (*CS* in [Patella and Ciaccia, 2008]). Of course, also both filters can be combined, but that only achieves additional speed up

[91] Actual measurements running the software including seek times on 2.5" HDD and overhead to create in-memory object representation in Java

when the number of images the are associated with the query keywords exceeds significantly the number of candidate images that are selected with ARP.

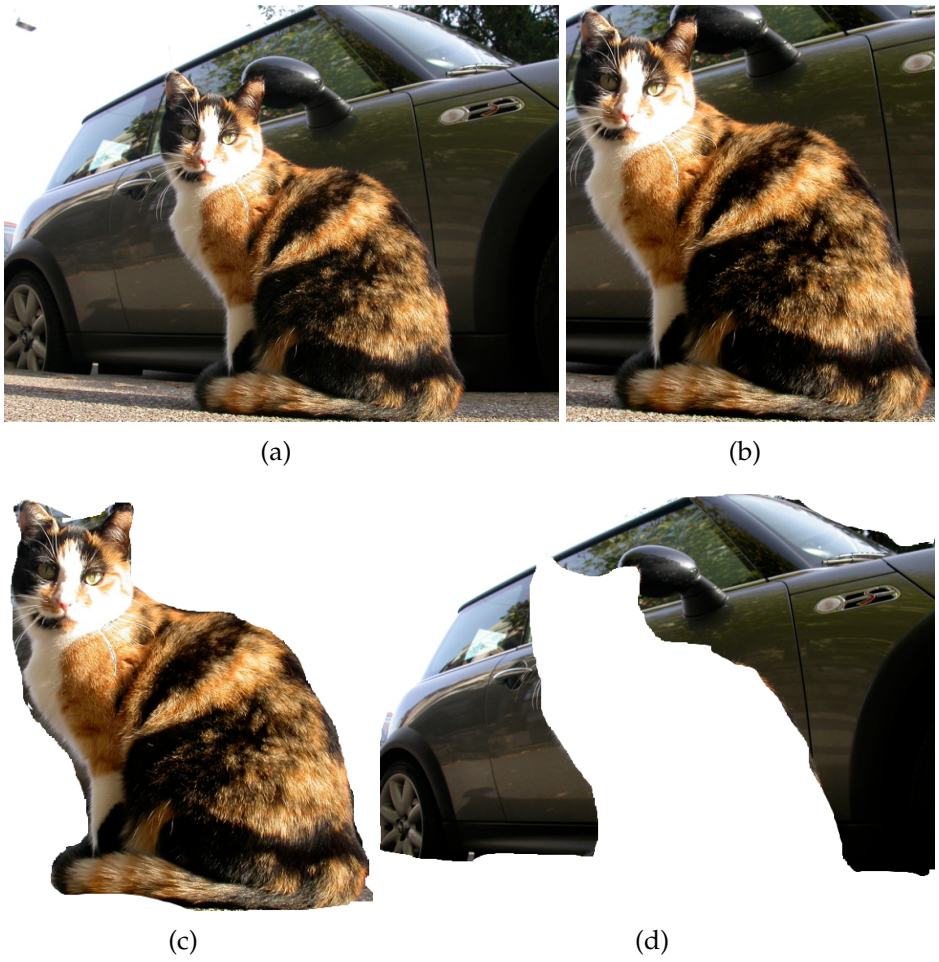
In the case of keywords, even when one of the least selective keywords for this collection is used ("sky" which has 846 hits), and features are read directly from disk, the search takes less than 20 seconds on the Tablet PC for a warp range $w = 5$ and a local context $lc = 3$ with background invariance disabled. As this is the biggest warp range and local context that we found helpful for the sketch-based searches combined with the least selective keyword filter and the slowest invariance option to process, this is the worst-case performance. Any other option that the user selects will make the system return the results in less time.

For the combination of ARP and IDM, we select four times as many candidates with ARP as we want results ranked with IDM. With $k = 200$ that means we search for 800 nearest neighbors with ARP, which is commonly still performed in less than one second. The re-ranking using IDM considers only those 800 candidates, for which it takes less than 20 seconds just as it did when using keywords to filter out undesired results.

Therefore, both filters are able to deliver results on the same hardware with the very expressive but also expensive IDM feature in a time the user might be willing to wait given the very good result quality found in Section 10.3.5. For the smaller collections consisting of 791 cartoon characters and 812 paper watermarks mentioned in Section 10.3.6, such a filtering would not even be necessary.

10.5.7 Impact of Fine-Grained Matching Tolerance through User-defined Regions of Interest

We incorporated user-defined regions of interest for our approach to retrospective geotagging in Section 10.4. As described in Section 10.4.3 on page 289ff, if the user selects an area on the map, a time range, or a region of interest inside the query image this will have the two desirable benefits of improving the result quality through less false positive matches as well as reduce execution time. For time ranges and map selection, these are enforced as filters on the candidates that proportionally reduce the number of distance computations that have to be performed between the features of the query image and candidate images. The dataset used for experimentation contained 1'408 geotagged reference images; for simple features like color and texture moments extracted from up to 9×9 regions as described in Section 10.4.3, nearest neighbor search can be performed using the same basic implementation as for sketch-based known image search fast enough to not harm user interaction – and enforcing filters on the area and time reduces the query execution time even further as the distances between many pairs of images does not even need to get computed. Therefore the main attention of the remainder of this section will be given to searches using a region of interest in the query image and keypoint descriptors. For regions of interest inside the query image, it's not the number of distance computations between images itself that gets reduced, but the computation becomes less expensive as the next paragraphs will explain in more detail.



(a) (b)

(c) (d)

Figure 10.76: Different regions of the same image: An example query image (a), a rectangular bounding box of the cat (b), a manually segmented region containing the same cat (c) and the car in the background (d).

Selecting Regions of Interest (ROI) in Images

In interactive systems, the user can contribute easily to the search by selecting a region of interest. For this, novel input devices like the one shown in Figure 2.3 and that are used in Section 10.3 for query by sketching can be used to select ROI which are not just simple bounding boxes (BBox). In case of relevance feedback, a tablet PC or digital pen can not only used to select relevant or irrelevant result images, but also to select the regions that make them relevant. Notice, that this might be much easier and intuitive to end users than assigning numeric values or preference judgements to images, since the task can be easily described as: "Select all relevant regions of the presented result images."

Figure 10.76 shows images and regions that will be used during the remainder of this section to illustrate the concepts and to perform quantitive measurements. Figure 10.76(b) shows the subimage that can easily be selected through a bounding box, e.g., using a mouse. Figure 10.76(c) and (d) show regions that are already much harder

(a) (b)

Figure 10.77: Keypoints in images: The 1481 SIFT keypoints extracted from the query image (a) and a very similar image of same size with 2536 keypoints (b) to which we will later on compare it.

to select using a mouse, but still fairly easy to select using a semi-automatic segmentation lagorithm like GrabCut [Rother et al., 2004] or even without any segmentation algorithm on a Tablet PC or with digital pen and interactive paper.

Our approach does not rely on pre-segmented images in the database, therefore it is more flexible in adapting to the user's information need. The user can, for instance, also select areas high contrast in itself, but low contrast from the background like the dark parts of the fur of the cat in Figure 10.76(c) or partly covered objects like the car in Figure 10.76(d). The latter is the common case where fully automatic image segmentation frequently fails as already mentioned in [Carson et al., 2002].

Keypoints Descriptors Extracted from Images with ROIs

For images in the database we extract salient keypoints and descriptors using SIFT [Lowe, 1999], which has been proposed for object identification and is invariant to scale, rotation, and to a certain extent to variations of the illumination or viewpoint. For every such keypoint SIFT extracts a 128-dimensional descriptor [Lowe, 2004]. In addition to this vector, the main orientation of the gradient and the scale at which it was detected are computed and stored with the keypoint. Figure 10.77 shows an arrow for each of the 1481 keypoints that have been extracted from the 682 $\times$ 512 pixel image in Figure 10.76(a).

The keypoint detection and descriptor extraction on the images in the database can be performed when the image gets added to the database, therefore ahead of time. For the query image, the keypoint detection and descriptor extraction can either be performed as soon as the user has selected the region of interest or even while the user selects the ROI – query keypoints can simply be filtered based on their position.[92]

[92]To a certain extent this will be done in a similar way to Pseudo Code Listing 10.16 in which keypoints are filtered based on rotation. However, Line 5 in Listing 10.16 filters the keypoints in reference images, while here the keypoints inside the query image will get filtered. Additional filters also on the

Matching Keypoints Descriptors using the Early Termination Strategy

To remain in line for time measurements with the proposed approach in [Lowe, 2004], matching keypoints in the ROI with corresponding keypoints in the database uses the squared Euclidean distance of the SIFT descriptors. For each keypoint in the query image the best match in the compared image needs to be determined and also the second best match, if the ratio between the distance of the two is used as quality measure (nearest neighbor ratio; cf. Section 10.4.3). Therefore a search for each keypoint in the query image for the k nearest neighbors in the reference image is performed with $k = 2$. This computation is quite intensive – in the particular example when matched with the image shown in Figure 10.77(b) it requires 1481×2536 computations of the distance between 128-dimensional vectors. Therefore this takes about 924 ms using a straight forward implementation based on Pseudo Code Listing 10.3 on a Intel Core 2 Duo processor with 2.33 GHz.

Taking a closer look at the keypoints in our query image, we see that many of them are located outside the region of interest. In fact, the area containing the cat as in Figure 10.76(c) contains only 684 keypoints and matching only these takes 438 ms. The car shown in Figure 10.76(d), despite of covering an area significantly bigger in size, contains only 536 keypoints and matching takes 350 ms. Notice also, that both image, Figure 10.76(a) and the original image of Figure 10.77(b), have exactly the same size and very similar content, yet SIFT detected approximately 70% more keypoints in Figure 10.77(b). Many of these additional keypoints are located in the lower part of the image, where there is additional road visible, which is in our case of little interest. When the rectangular selection shown in Figure 10.76(b) is used, also additional keypoints are contained, 900 for total, and matching takes 576 ms and therefore more than 30% longer than our region of interest.

In order to speed up finding the matches, we can apply the early termination strategy of Pseudo Code Listing 10.20 for the computation of the squared Euclidean distance. The squared Euclidean distance using early termination is identical to Pseudo Code Listing 10.18 except for Line 9 – it returns the sum directly without computing a square root. The use of the early termination reduces the computation time to 629 ms for the entire image, 311 ms for the cat region, 236 ms for the car region and 401 ms for the rectangular bounding box without changing the result.

These results are based on the nearest neighbor ratio matching strategy. Other strategies can be used with early termination as well – or used even better: When the sum of best-matched keypoints or average inverted squared keypoint distance is used for ranking matches as described in Section 10.4.3, there is no need anymore to determine also the second best match and therefore $k = 1$ in nearest neighbor search, which further increases the effectiveness of the early termination strategy. When matches are determined based on a distance threshold as in the match counting ranking approach of Pseudo Code Listing 10.12, early termination can be used in range search of Pseudo Code Listing 10.19 or the variant Listing 10.21. The latter is commonly the most effec-

reference image would only make sense if the placement of matches should also be restricted – which would only make sense if no translation invariance should be achieved, which is commonly not the case for retrospective geotagging or other applications of keypoint descriptors.

```
RANGESEARCH1(𝔔, Docs, Φ, Δ, range)
1  For each 𝔑 ∈ Docs
2       do
3          dist ← Δ(Φ(𝔔), Φ({𝔑}), range)
4          if dist ≤ range
5            then return {(dist, 𝔑)}
6  return ∅
```

Pseudo Code Listing 10.21: Variant of RangeSearch of Pseudo Code Listing 10.19 that returns only the first match with a distance within the range

tive application of early termination as it returns directly the first result that remains below the threshold; this is already sufficient to decide whether a query keypoint can be matched or not which is all that is required on Line 5 of Listing 10.12 to increase the match count.[93]

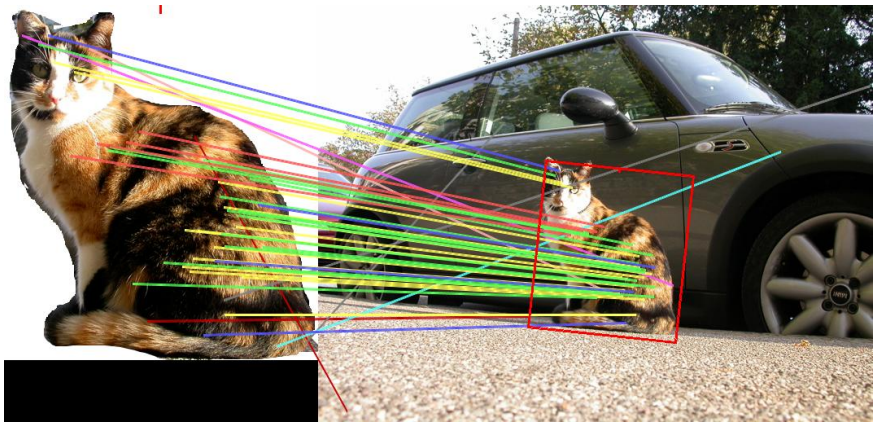
Computing and Constraining Transformations from Keypoint Matches

Using not only the number of matched keypoints, but also their location and the resulting rotation, scaling and translation of the image, an affine image transformation can be constructed, which maps the query region to the relevant region of the image from the database. When the query image is provided by the user and not part of the database of known images, it might not be possible to define a single threshold on distance or nearest neighbor ratio as proposed in [Lowe, 1999] for the selection of matched keypoints. In particular, if the threshold is tight to find only very accurate matches, the system may return few or not even a single match for the region of interest. Instead, an ordered list of matches can be computed such that it always contains enough entries even in unfavorable cases, e.g., where illumination is very different or only similar objects are present.

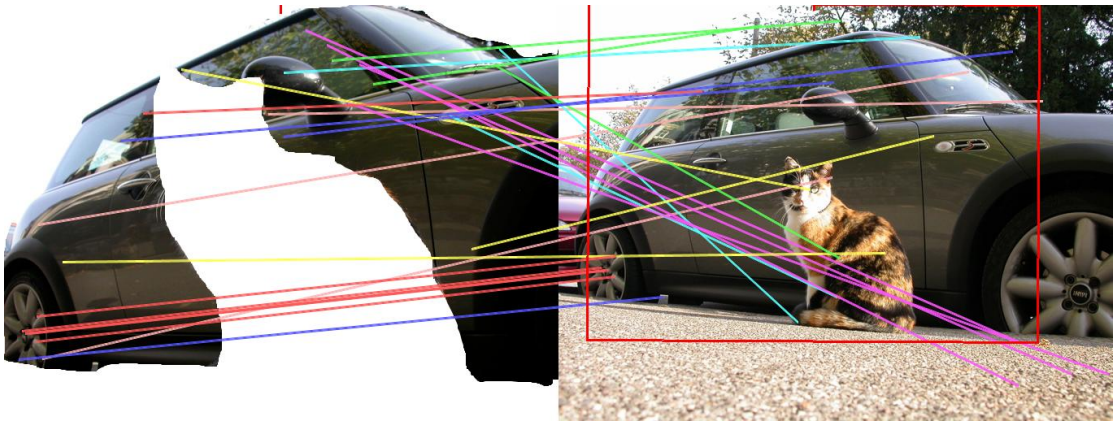
These matches can be clustered based on the scale and rotation in which this transformation would result. Selecting only the cluster with the most consistent transformation (lowest error in homography) and filtering with RANSAC, the transformation gets more robust to many mismatched keypoints as long as at least three good ones remain [Studer, 2008]. The results of this matching step are displayed in Figure 10.78. Notice if entire images containing several objects like in Figure 10.76(a) were used, either only one region can be matched or both regions just a little – with the user having no control on what will happen.

As a result, the matching region of the ROI in the images of the database can be determined. Using regions in relevance feedback on result images, the corresponding keypoints in the query can be identified and weighted or excluded. Also certain types of transformations can be excluded, e.g., if the user does not want to allow matches

[93] The particular effectivity depends on the distance threshold used as the range. This threshold is not always easy to determine – as already mentioned in Section 10.4.3 and also in [Lowe, 2004].



(a)



(b)

Figure 10.78: Result of determining corresponding regions: Lines connect keypoints which are matched, color of lines indicate cluster. The red lines have been used to determine the affine transformation which maps the query image inside the red bounding box.

of the ROI being upside-down or scaled to small size. Such constraints can already be enforced within the loop over each keypoint since we have stored their orientation. If the orientation of a keypoint in the matched image differs too much from the query keypoint, we can already reject it and do not even need to compute the distance between the descriptors as described in Section 10.4.3 on page 301. This further reduces the matching time, in our experiments with an allowed angle of 10 degrees for the cat region to 152 ms without noticeable changes in the result, because the determined transformation would stay within that limit anyway.

In general, if the user can restrict the allowed transformation, this will *improve* the quality of the results as unwanted results are not considered. Figure 10.79 shows the corresponding regions next to each other after transformations were applied.[94]

[94]Restricting the allowed transformations *decreases* the time to compute the matching – similar to selecting areas on the map reduces the number of images the query image needs to be compared to in retrospec-

(a)

(b)

Figure 10.79: Images placed next to each other after affine transformation was applied.

Table 10.8: Times to Match SIFT Keypoints of Figure 10.77(a) with Figure 10.77(a) on a Core 2 Duo with 2.33 GHz

Criterion	Full Image	Cat BBox	Cat ROI	Car ROI
number of detected keypoints	1'481	900	684	536
time to match w/o Early Termination	924 ms	576 ms	438 ms	350 ms
time to match using Early Termination	629 ms	401 ms	311 ms	236 ms
limit match to rotation $\pm 10^{\circ}$	n/a	n/a	152 ms	n/a

Summary of Fine-Grained Matching Tolerance through User-defined ROI

Table 10.8 summarizes the numbers. In conclusion, just by applying the Early Termination strategy in the identification of the 2 nearest neighbors for each keypoint, the overall time to match the two images was reduced from 924 ms to 629 ms, which cor-

tive geotagging in Chapter 10.4.3. Another approach in which adding constraints to image deformations improves both, the result quality and the time to compute results, is described in [Barnes et al., 2009].

responds to a reduction by 31.9%. Letting the user also select the region of interest and define a degree of rotation which will not only reduce the time for matching to 152 ms (just 16.4% of the original, unoptimized matching time or a speedup of 6.08), but will also increase the quality of results – at least if these restrictions correspond to what the user actually is hoping to find.

At this point, further improvements might also consider optimization of the clustering and RANSAC, since those two steps together require approximately the same time after matches have been determined. Furthermore, any available knowledge about the desired content should be evaluated to limit the number of images for which correspondence needs to be evaluated. The similarity between the corresponding regions itself does not need to be based on SIFT, but can be any appropriate feature(s) and distance measure computed on regions [Springmann and Schuldt, 2008]. For retrospective geotagging as described in Section 10.4, neither RANSAC to determine a transformation was used nor any additional distance measure for the identified corresponding regions is used.

Other Optimizations used in Retrospective Geotagging

For the purpose of retrospective geotagging more relevant was multithreading and prefetching. When interested in the fastest computation of the transformation for a single image pair, the looping over all keypoints in the query image could get parallelized. But as the database contains many reference images, it is sufficient to parallelize the computation for distances between different image pairs as it was done for the medical image classification and sketch-based search for known images with a dispatcher as described in Section 10.5.3. On some architectures, this will also have the added benefit of better locality of data when the distance to a reference image is processed in a single thread, therefore access to the keypoints in the reference image remains thread local.

To evaluate the filter predicates for areas selected on a map and time ranges efficiently, the features extracted from the reference images of the dataset are managed in a relational database with extensions to handle geolocations. We use PostgreSQL[95] for storing the image metadata; PostGIS[96] adds datatypes for storing and querying geographic information in SQL. For time ranges, the index structures of relational database are very effective; for areas selected on the map, the PostGIS extension provides index structures either based on the R-tree [Guttman, 1984] or GiST [Hellerstein et al., 1995] – depending on the used version and particular configuration.

To reduce I/O latencies, prefetching the features of the next image can be of great importance: The dispatcher assigns distance computations from results of the PostgreSQL database which enforce the filters. By prefetching the features of the next image in the result set from the database in a background thread, the dispatcher assures that the query execution remains CPU bound. To also reduce further the cost of finding the best matching keypoints, [Lowe, 2004] proposes the use of a Best-Bin-First (BBF) strategy on a k-d tree that resulted in experiments in reduction of the search time by two orders of magnitude. However, this speedup was only achieved by accepting approximate

[95] http://www.postgresql.org/

[96] http://postgis.refractions.net/

search results. In contrast, neither early termination strategy, nor multithreading, nor caching have any degrading effect on the retrieval results. Adding a region of interest, restricting the area on a map and time of relevant images, limiting the amount of rotation may even increase the accuracy of results. Combining these optimizations with the BBF strategy or other approximation techniques is still possible.

10.5.8 Conclusion on Reduction of Execution Time

We have presented in detail our optimized implementations using three different generic approaches to reduce the execution times of searches: Early Termination Strategy in Section 10.5.2, Multi-threading in Section 10.5.3, and Caching in Section 10.5.4. All three approaches can be applied without any degradation of retrieval quality. Furthermore, all three approaches can be used with any of the distance measures that are used for the features that have shown good retrieval quality in the initial version that was not optimized for speed. In particular, all these strategies can be applied to the computation using the Image Distortion Model (IDM) and matching of SIFT keypoint descriptors without a need to modify or adapt the approach or parameters of matching as metric indexing or visual codebook approaches would require.

We have shown that the generic optimizations of our building blocks are applicable to particular search applications:

- In Section 10.5.5, we showed how we can speed up the automatic medical image classification by factors of 3.29 to 4.86 just by applying the early termination strategy. Combining this with multithreading on an 8-way server achieved an overall speedup of 37.34 and therefore allow the classification using IDM with a warp range $w = 3$ and local context $lc = 2$ in 13.3 seconds per image.
- In Section 10.5.6, we showed that query by sketching can be used for searching known images in a collection of 25'000 images with nearly no noticeable delay for common searches with results being presented to the user in less than 40 ms on a Tablet PC without being backed by an additional, more powerful server. More complex searches can take more time, but remain below half a second as long as required features are found in cache and IDM is not needed as a distance measure. When IDM is needed, the search becomes CPU bound and presented filtering techniques help to let the user not wait for more than 20 seconds for the results even in worst case with $w = 5$ and $lc = 3$.
- In Section 10.5.7, we have shown that exactly the same optimization techniques can also be used with keypoint-based approaches for matching images. In particular the optimizations can be applied to user-selected regions of interest in query images for retrospective geotagging. Furthermore we showed that these optimizations can be integrated with user-selected areas on maps and time ranges that will not only further reduce execution time, but also reduce the number of false positives and therefore improve retrieval quality.

Of course, far more possibilities exist to reduce the execution times. An overview was presented in Section 10.5.1. Nevertheless the mentioned approaches were sufficient to reduce the execution times enough to let the delay between issuing the search

and revising the search results not harm the user experience; thus making further optimization not necessary to allow the user to interact well with the system.

11

User Interaction

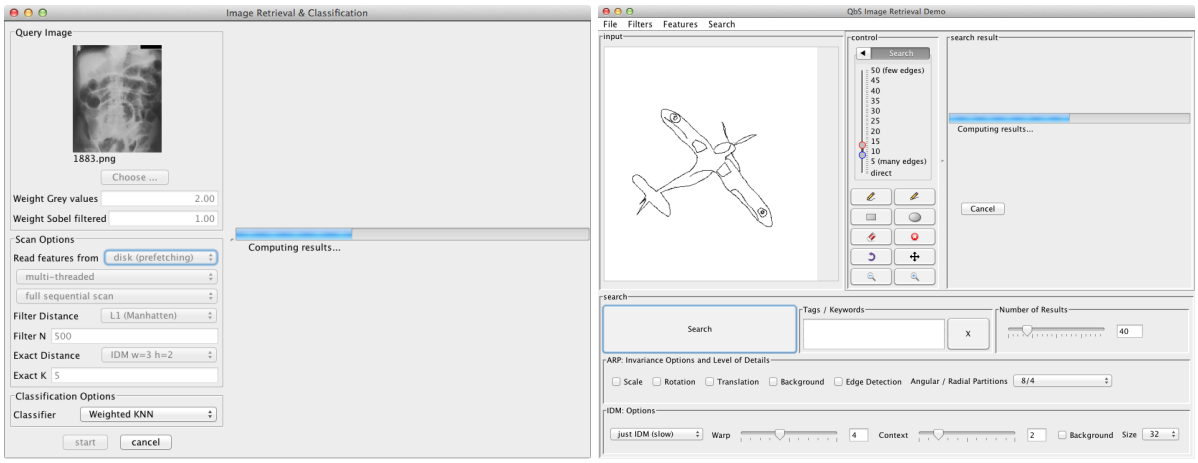
[van den Broek et al., 2004] and [Jörgensen, 2003, pp. 272ff] mention the fact that research in the field of content-based image retrieval has seen primarily work on the underlying structures for searches, whereas no or only little attention has been paid to the user interface. In this chapter, we will give some insights to the implementation of the user interfaces of our prototype applications and the way the user interacts with them. Particular emphasize will be given to concepts that are not only specific to one particular implementation, but can be of greater interest as similar challenges may occur for other applications in the same domain or even any digital library with similarity search functionality.

Section 11.1 presents an implementation that is generically applicable to allow the user to monitor or abort long running activities. Section 11.2 and 11.3 focus on aspects of user interaction that have to be considered when using not-so-common input devices, in particular Tablet PCs and Interactive Paper. Section 11.4 concludes.

11.1 User Control of Search Progress

When search results are delivered without much noticeable delays, there is no need to do anything else than perform the search and present the results.[1] For similarity search with computational expensive distance measures like the Image Distortion Model (IDM) or matching of SIFT keypoints –at least without approximate search techniques that quantize features very heavily– such quick search results will frequently not be possible for collections containing thousands of images. Therefore the user has

[1] For presenting the search results quickly, making the query execution fast alone is not sufficient. Chapter 10.5 already mentioned generating images of various resolutions, in particular thumbnails, and caching of these images. [Sobel, 2010] introduces the term *Time-to-Interact (TTI)* for web-pages that is determined by three aspects: *network time*, *generation time*, and *render time*. For more general settings,*network time* should probably get replaced with *data transmission time* and caching of images is one technique to reduce it. The actual query execution is usually the step in image searches that requires most of the *generation time*. Providing thumbnails in the appropriate size and format for displaying them on the user's screen may reduce the *render time*. But in every application there will be many other components contributing to the TTI in some way – and they might also require attention to deliver a good user experience.



(a) Medical Image Classification (b) Sketch-based Known Image Search

(c) Retrospective Geotagging

Figure 11.1: User Interfaces Displaying Progress and Ability to Abort Search

to experience the feeling of giving away control to the system for the duration it takes to execute the search until the user can resume working and issue new searches.

It is common knowledge that people will get impatient and even assume that the operation failed after few seconds of inactivity. [Nielsen, 1993, p. 135ff] describes a rule of thumb that for activities that take more than 10 seconds, percent-done indicators should get displayed to the user to inform about the progress. The importance about this kind of feedback has been investigated in [Myers, 1985]. Another important aspect for the user might be to abort the search to leave the user in control of long-running operations.

Figure 11.1 shows three different user interfaces to similarity search applications that all provide information about the progress of a long running search execution in the form of a progress bar (in blue and in the right-hand side of the screen in Figure 11.1(a) and (b), yellow and at the bottom of the screen in (c)) and the ability to abort the search through a button labeled “cancel”. All three implementations are backed by the same implementation of a query execution building block. Implementing the progress indi-

cator functionality in a very generic and reusable way can get achieved easily – when respected early enough in development.

11.1.1 Starting Point: Search Primitives using Iterators

Pseudo Code Listing 10.2 and 10.3 described simple implementations to perform the commonly used search primitives for range search and kNN search. Their method signature did not have to be modified for any of the optimizations in Chapter 10.5.2 – 10.5.4 and remained RANGESEARCH$(\Omega, Docs, \Phi, \Delta, range)$ and NEARESTNEIGHBORSEARCH$(\Omega, Docs, \Phi, \Delta, k)$.

In most implementation languages, passing on the entire set of documents *Docs* would not be ideal and the elements would get processed anyway one – or very few if considering multi-threading– at a time. *Iterators* are a concept to provide programmatic access to a single element at a time (cf. [Gamma et al., 1995, pp. 257–272]) and object-oriented programming languages like C++ and Java and the basic definition as well as implementations for many use-cases are part of the Standard Template Library (STL) for C++ and the *java.util* package of the Java class library. Assuming interfaces for the basic concepts of *Feature* and *DistanceMeasure* do exist and features get extracted from the documents in the collection ahead of search and the feature of the query image just ahead of invoking the search primitives, the method signatures can get expressed in Java using Generics as:

```
1 public interface QueryExecutor {
2     Set<DistanceScore> rangeSearch(Feature queryFeat, Iterator<Feature>
          referenceFeatures, DistanceMeasure<Feature> dm, float range);
3     List<DistanceScore> nearestNeighborSearch(Feature queryFeat, Iterator<
          Feature> referenceFeatures, DistanceMeasure<Feature> dm, int k);
4 }
```

Pseudo Code Listing 11.1: Simple interface to define search primitives for RANGESEARCH and NEARESTNEIGHBORSEARCH in Java.

DistanceScore is assumed to be a container for the distance as well as a reference to the document to which the feature belongs that resulted in that score.[2] A simple implementation can then take one feature at a time by invoking *referenceFeatures.next()* until *referenceFeatures.hasNext()* returns ***false***. A multi-threaded implementation using a dispatcher as described in Chapter 10.5.3 will simply take several elements from the iterator and assign them as units of work to the worker threads.

11.1.2 Observing the Iterator to Track Progress

When the iterator gets initialized to the extracted features of the images in *Docs* we should keep track of the number of images in total: $numTotal = |Docs|$. As the iter-

[2]For this, the interface *Feature* should require to provide a method to get the document to which the feature belongs. Complex distance functions can get represented in this class hierarchy by having a specialization of *Feature* like *CompoundFeature* that acts as a container for a set of features and implement corresponding distance measures satisfying *DistanceMeasure<CompoundFeature>*.

ator gets called for every element that gets processed, the iterator is an ideal place to implement a counter *numProcessed* that tracks the number of elements that have been processed. The information about percentage done then can simply be computed from *numProcessed/numTotal*.

Let us assume that there is some user interface component providing the method *UI. reportProgress (**long** numProcessed, **long** numTotal)* that can display a progress bar to the user. The generic functionality to provide an iterator that can be observed by any implementation of the interface *UI* by Pseudo Code Listing 11.2.

```
1  public class ObservableIterator<Iterator<Feature>> {
2      final long numTotal;
3      long numProcessed = -1;
4      final Iterator wrappedIterator;
5      final UI observer;
6      public ObservableIterator(Iterator realIterator, UI ui, long numTotal) {
7          this.wrappedIterator = realIterator;
8          this.observer = ui;
9          this.numTotal = numTotal;
10     }

12     public boolean hasNext() { return wrappedIterator.hasNext(); }

14     public Feature next() {
15         numProcessed++;
16         observer.reportProgress(numProcessed, numTotal);
17         return wrappedIterator.next();
18      }
19 }
```

Pseudo Code Listing 11.2: ObservableIterator: A decorator for an existing iterator that allows to track the progress of an ongoing iteration.

In the terminology of design patterns, the *ObserableIterator* is a *Decorator* [Gamma et al., 1995, pp. 175–184] that extends the functionality of an existing iterator, the *realIterator* . The UI implementation would follow closely the *Observer* pattern [Gamma et al., 1995, pp. 293–304] that describes a method call to invoke to get notified of any important change in the system.

11.1.3 Allowing the User to Safely Abort Long-Running Iterations

There can always be situations in which the user does not want to wait for the results anymore, e.g., because the user recognizes that there was a mistake like a misspelling in the keywords used in the query or found a better image to be used as an example or just realizes that the expensive computation will last longer than the user is willing to wait and therefore wants to change the query options to something that is faster to process. Also this fundamental functionality can easily be integrated using a very similar approach to the *ObservableIterator* .

Pseudo Code Listing 11.3 provides the basic implementation using Java syntax to add the ability to abort any search simply by telling the iterator to abort which will

```
1  public class AbortableIterator<Iterator<Feature>> {
2      final Iterator wrappedIterator;
3      boolean isAborted = false;
4      public AbortableIterator(Iterator realIterator) {
5          this.wrappedIterator = realIterator;
6      }

8      public boolean hasNext() { return !isAborted && wrappedIterator.hasNext
           (); }

10     public Feature next() {
11         if (isAborted) throw new NoSuchElementException();
12        return wrappedIterator.next();
13     }

15     public void abort() { this.isAborted = true; }

17     public boolean wasAborted() { return isAborted; }
18 }
```

Pseudo Code Listing 11.3: AbortableIterator: A decorator for an existing iterator that allows to abort any ongoing iteration.

result the *QueryExecutor* to be no longer able to get next units of work and therefore will end performing the search as soon as the current unit of work has been finished. By invoking the method *wasAborted()* of the iterator, the *QueryExecutor* can even check whether the results will be needed or not as there might be no need to return the results if the user expressed no interest by clicking a cancel button.

11.1.4 Extending the Use of Iterator Decorators

As both functionalities, being able to track the progress of an execution over many items as well as allowing the user to cancel the execution, occurs frequently in combination, the two implementations *ObserableIterator* and *AbortableIterator* can be merged into a single class providing both functionalities simultaneously. In order to allow flexibility and alternative implementations, the original classes can be turned into interfaces.

Another use case in our prototypes for exactly the same implementation is batch feature extraction for letting the user add a set of new documents to a collection (a.k.a. *bulk insert*): Depending on the number of images that the user adds, this operation is likely to take not only more than a few seconds, but minutes or even hours – even when features are extracted efficiently as described in Chapter 10.5.3. It is therefore essential to report the progress to the user and allow to abort the operation in such a way, that the system remains in a consistent state – that is, all new documents that have already been processed are available through content management and the indexes are up-to-date. As *AbortableIterator* implements an abort by signaling that there is no next element even if the wrapped iterator would have more elements, to the outside this appears as if the regular end of iteration was reached and the system will end the bulk insert as any regular bulk insert by updating and closing files and database connections after

the last item has been processed. If an application prefers special treatment of aborted iteration, the kind of end of iteration can be determined by invoking *wasAborted()* of the *AbortableIterator* .

Other aspects, not directly related to user interaction, but query execution that can be implemented in a very similar way are:

- A *PrefetchingIterator* to assure in a background thread that the next data elements are already queued in memory for processing – independent of whether the data elements in this iteration are image files or features read from disk.

- Predicate filters evaluated by a *SelectiveIterator* , that simply skips any feature that does not satisfy the predicate. This already provides a straight-forward implementation to evaluate Equation (5.29) of page 141. The combination with a *PrefetchingIterator* can help to perform the usually not computationally challenging, but I/O intensive evaluation of the filter in a different background thread isolated from the worker threads that perform the CPU-intensive distance computations.

- More optimized implementations can use dedicated index structures to perform a lookup of all images in the collection and their extracted features that satisfy the predicate. The results of such a lookup can frequently be turned into an iterator. For instance, the cursor to a *ResultSet* returned by a relational database always provides the functionality to test whether the end has been reached and otherwise move on to the next result.

- Another source for such an iterator can be the results of previous invocation of a search primitive with different parameters – thus requiring very little extension to provide a multi-step approximate search that uses for instance the Euclidean distance or ARP for the faster execution to determine candidates that will then be re-ranked using the more expensive Image Distortion Model (IDM).

As all these operations can get wrapped transparently with the *ObservableIterator* and *AbortableIterator* , hence providing the hooks required for user interaction without adding new complexity to the implementation of the functionality.

11.2 User Interaction using a Tablet PC for QbS

The idea to use a pen for interacting with a computer has been reported already in the early 1960s in [Sutherland, 1964]. Since then, many further investigations on enhancing the user experience with computer systems have been undertaken.

Chapter 10.3 described in detail our approach to performing known image search from user-drawn sketches. Due to the challenges in QbS that have been described in Chapter 10.3.3, successful use of this approach is only possible with a user interface and interaction design that is pleasant to use. The prototype that has been implemented in [Kopp, 2009] and significantly extended in [Springmann et al., 2010a] and further in [Giangreco, 2010, Kreuzer, 2010] tries to achieve this on Tablet PCs through an implementation for which we will explain the design choices in this section.

Figure 11.2: Tablet PC for Sketching Visual Examples

A Tablet PC provides in general much better user experience for drawing tasks. However, as most interaction will happen using a stylus, this also requires careful consideration in the design of the user interface. In this section, we will therefore concentrate on the peculiarities when using a Tablet PC and provide suggestions how to implement a user interface that can be pleasantly used with such a device.

11.2.1 Peculiarities when Using Tablet PCs for Collecting User Input

Figure 11.2 shows the prototype in use, running on a Lenovo ThinkPad X200t tablet PC with a 12.1 inch screen. Notice that tablet PCs like these are predominantly used with a stylus. The particular model does also support direct interaction by touching the screen with the finger –even supports so called "multi-touch" events when several fingers are used similar to smartphones and the now popular "tablet" (without the appended "PC") like the Apple iPad–, but the built-in digitizer by Wacom can deliver much greater precision when used with a stylus. The device is pressure sensitive and able to distinguish between a hand resting on the screen to use the stylus and fingers

Figure 11.3: Tablet PC running the QbS software for Sketch Acquisition

trying to trigger some action – something which common capacitive touch screens of smartphones and tablets struggle with.

This aspect is extremely important as any longer use of the device will tire the hand used for drawing if the hand cannot find some rest on the screen bezel or screen itself. Another important aspect of usage is the fact, that in contrast to a regular mouse or trackpad or pointing stick[3] which are based on relative cursor movements, the digitizer in tablet PCs use absolute positioning. Therefore the stylus and with it the hand has to be moved to the location on screen where some activity should get performed.

In the center of the screen of the application, there is a field to enter keywords. Typing can be done using a virtual keyboard, but when using this feature a lot, it is more convenient to use a physical keyboard – either an external keyboard connected via USB or Bluetooth, or the keyboard that is built into this so-called "convertible" Tablet PC. When the keyboard is mostly used, the screen can be turned around and up such that the keyboard gets accessible and the Tablet PC can be used like a regular laptop. Figure 11.3 shows the Tablet PC with the screen only turned a bit to the side to reveal both

[3]Pointing stick is the generic term for the pointing device that is best known under the product name "TrackPoint" that IBM introduced for the ThinkPad product line.

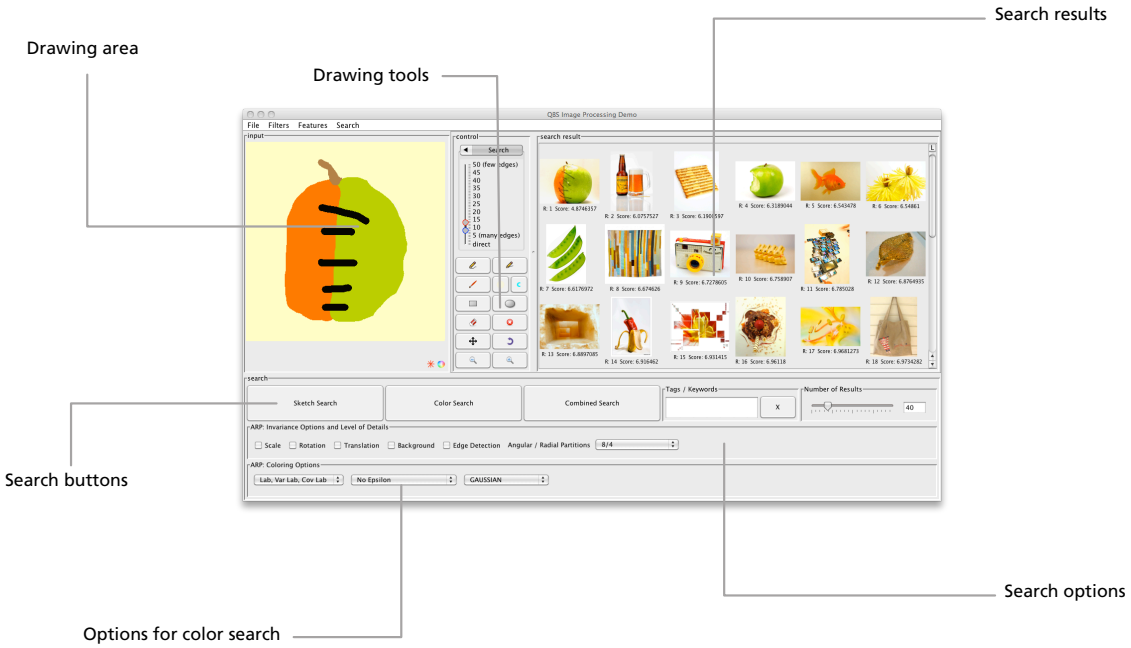


Figure 11.4: User Interface Elements

modes of operation. When mainly using the stylus, the screen is usually placed like shown in Figure 11.2 as this provides better opportunity to rest the arm on the bezel.

To avoid the need to constantly move the hand across the screen, the interface is arranged into groups of UI elements that are commonly used together during particular phases of interaction. Figure 11.4 shows the UI with the extension to color [Giangreco, 2010, Giangreco et al., 2012] and annotations of the different groups.

- The *drawing area* to the upper left provides 400×400 pixels and therefore enough space for drawing a sketch that gives an impression of what the sought image looks like. It would not be sufficient to draw also very fine details, but these are not needed since the sketch is only used to search the known image as the underlying assumption remains that the tasks for which this system will be used are known image searches. For other intended tasks, like image composition and drawing of new images that will be used directly, a different UI design would be needed. But at the chosen size it allows to draw the sketch fairly quick without the need to move the hand a lot and leaves enough space for options and search results, such that the user can modify the sketch in response to the images found by the system.

- Multiple *drawing tools* are located in the middle of the screen, including tools for drawing lines and ellipses as such shapes are not easy to draw accurately with the pencil tool that is otherwise predominantly used for sketching. To perform minor corrections, the image can be moved inside the drawing canvas, rotated, scaled, and erased – either in parts or entirely. All these tools can be reached with very

little movement of the drawing hand while the clear separation from the drawing area and there placement prevent accidental clicking.[4]

- When the sketch has been drawn, the attention will shift to the *search buttons* and *search options* below the drawing area and the *search results* to the right. The most important buttons are located just below the drawing area and are big in size, such that they can be used fast and frequently with the stylus without effort. Furthermore, the placement of the search options in the lower part assures that neither the sketch nor the search results will ever get covered with the hand invoking the action. The slider to right side below the search results allows to adjust the number of nearest neighbors that get displayed. At this position, it is very close to where the stylus would point whenever the user has scrolled to the end of the search result list using the scrollbar at the right side of the screen.

- One element breaks with this overall layout: Just above the drawing tools, there's a slider that controls the edge detection (the range of β values). This has a double function as will be explained later. For now, it should simply get ignored – as the program would do when invariance to edge detection is enabled, which is equivalent to setting the range over all β values.

In order to make the usage even more convenient, many options are not only available via buttons. For a traditional setup using keyboard and mouse, keyboard shortcuts would certainly be of great help. But in an application that is mainly controlled using a stylus, the shortcuts have to be available through strokes of the stylus itself.

11.2.2 Menus Optimised for the Use of a Stylus

[Giangreco, 2010] collected the findings of several studies that investigated ways to organize tools used with mice and Tablet PCs, for instance in form of *pie menus* (cf. [Kurtenbach, 2004]). Pie menus order the elements of a menu circular around the mouse, allowing to have the same distance from the current mouse position to each of the entries in the menu. Because of the equidistant placement of the items, pie menus avert that the user has to cover large distances to select certain tools. Empirical studies revealed the superiority of pie menus over linear menus [Callahan et al., 1988], which are still widely used in most of today's applications, not only in time-efficiency, but also in reducing the number of errors.

Later on, the idea of pie menus was expanded to use marks for tool-selection as described in [Kurtenbach and Buxton, 1994], leading to so called *marking menus*. Marking menus are characterized by the fact that the user does not need to lift the pen or finish the click to choose an element from the menu. Instead, after pressing down the pointing device and waiting for a certain time, a menu appears on whose elements the user can release the pressure on the device and therefore choose a menu item.

[4] On hardware that would not reliably differentiate between intended clicks of the stylus and unintentional activity caused by the hand resting on the screen as it sometimes happens with smartphones and tablets, the same placement would clearly be not optimal. In particular right-handed users would have to fear that they accidentally erase the entire sketch. But the Tablet PC we are using does not suffer from such problems and reliably detects the resting hand as such.

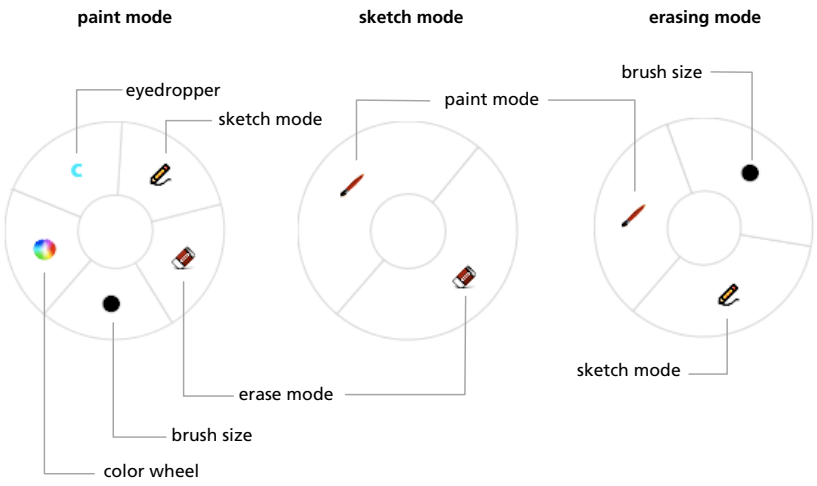


Figure 11.5: User Interface Elements: Marking Menus

Adding the possibility to colorize sketches to the existing QbS-system increases the number of possible tools and options for the user. To enhance the usability of the system we add a pie menu to the application as described in [Kurtenbach and Buxton, 1994]: A circular menu appears after the user presses and holds the pen for around 300ms, from which the user is then able –without having to lift the pen– to choose an element.[5] Furthermore, by making the menus context-aware, the elements displayed in the menu adapt to the currently activated action (i.e. drawing, painting, erasing, etc.). Figure 11.5 shows the different implemented menus for the possible states in the software.

11.2.3 Interaction with Search Results

For any image-related search task, reviewing the search results is of utmost importance. Scrolling through result lists is commonly more convenient that stepping through pages, in particular when a Tablet PC is used with a stylus: The stylus can rest pressed on the scroll bar and moved with little effort to browse through results. In contrast, navigating with clicks through result pages requires to press and raise the stylus, which can be more tiring. This does not only relate to the effort of the hand holding the stylus, but also the eye that frequently will refocus from the result list being browsed to the button that is pushed with the stylus. The latter is caused by the fact that there is –in contrast to mouse and keyboard– no precise tactile feedback when pushing virtual buttons on screen: Whether a button was pushed successfully or the stylus missed the UI control and pressed any other area on the screen doesn't feel any different. Therefore, even when the stylus remains in place above the appropriate button, as soon as the button

[5] [Kurtenbach and Buxton, 1994] also suggests that improvements in time efficiency can be gained, if the user can choose elements from the menu without having the need to wait for the menu to pop-up, but instead, just by drawing certain marks. Although this might work for most of the applications, using only marks without having the need for any menu to appear fails in the usage of a drawing applications, as the system will not be able to distinguish between a mark and a drawing stroke. We therefore did not implement this idea in the system presented here.

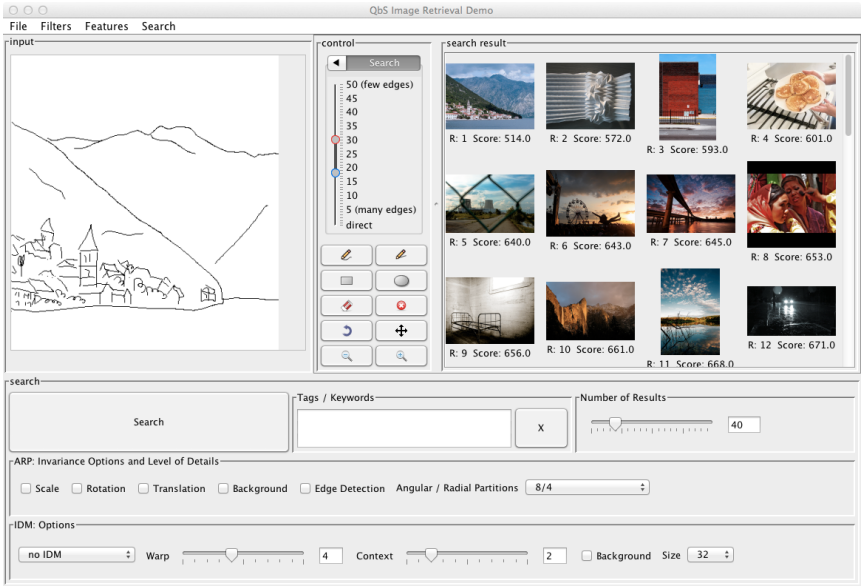


Figure 11.6: Search Results using ARP with 8 angular and 4 radial partitions.

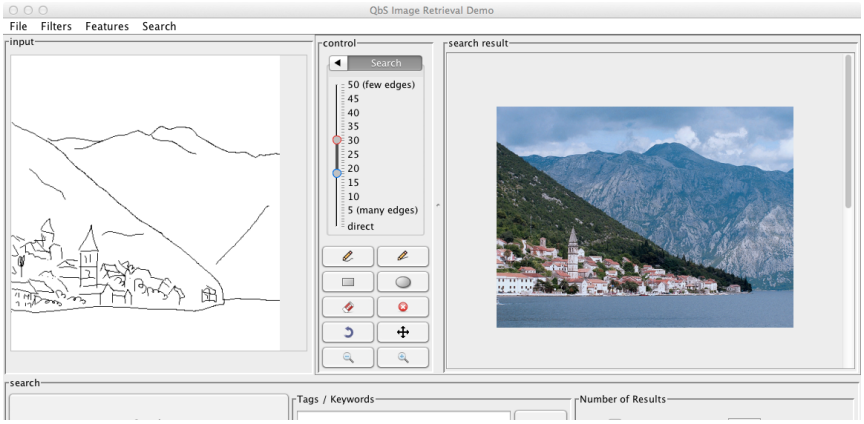


Figure 11.7: Enlarged View of the Top Ranked Result

is released and the eye will focus on the result list again, it will have to focus again the button for being able to push it again. With scrollbars, the stylus can simply rest in the pressed position and moved without the need to move the eye away from the result list.

For known image searches, frequently the user will be able to detect the sought image very quickly even if the system presents just small thumbnails. Figure 11.6 shows the nearest neighbors to the query sketch on the left using ARP with 8 angular and 4 radial partitions.

To enable result usage, the user must have access to the image at full size as well as the metadata associated to it. When the user clicks on any of the result images, the image gets enlarged to occupy the entire area for search results as shown in Figure 11.7. The related metadata displayed below the image; Figure 11.8 shows the same result view when scrolled down. The user can export the found full size image using simple drag-and-drop gestures on the image. A simple click anywhere on the image or the result

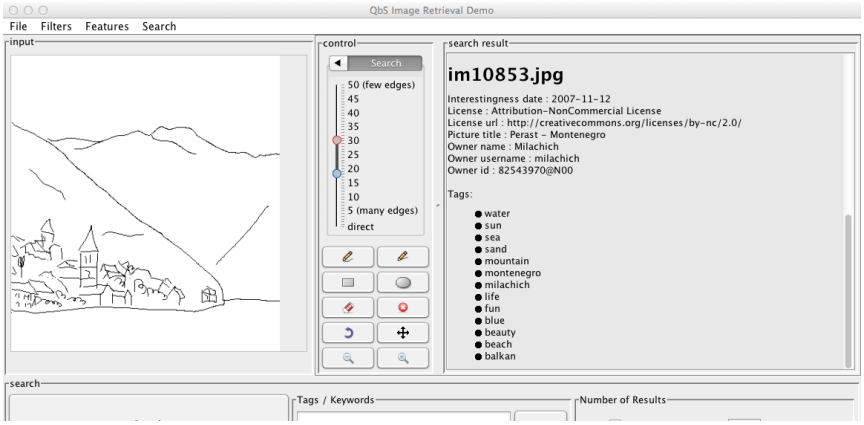


Figure 11.8: Metadata for the Top Ranked Result

screen will return to the result list, therefore allowing to switch very easily between the two views. The drag-and-drop gestures for the full size images can even be invoked through the thumbnail images, thus making it possible to use full size result without ever having to switch to a different view.

In known image search tasks, the user must be able to adjust the query quickly. In an approach with user-drawn sketches, this implies that the sketch must remain visible and editable at all times. The latter is a consequence of the observation in Chapter 6.3.6: As only the known image can be considered relevant to the search, interaction techniques like relevance feedback that modify the query implicitly in subsequent iterations are not applicable. Therefore the query reformulation has to be more explicit and this is done by modifying the sketch and other search parameters like enabling or disabling invariances.

For comparison: Some interfaces for image search systems that were mentioned in Chapter 6.2 use an interaction flow where once the query was issued, it gets hidden or at least reduced in size to give more space to the search result thumbnails. When hovering the mouse over the images, additional metadata and/or a slightly enlarged view of the result image may appear and when clicking on the image, the user gets redirected to a completely new page showing just the image and its associated data. Such an interface causes problems for the use with a Tablet PC for several reasons:

1. Hovering over an image is possible also with a stylus on a pressure-sensitive Tablet PC, but it is much easier to simply click on the image as any firm touch of the tip of the stylus will issue such a click.

2. Navigating back and forth between several entire screens is cumbersome as in different screens the location of UI elements changes and the hand always has to move to reach these elements. In particular the placement of a back button would be crucial for quick navigation. Several web browsers are now capable of recognizing mouse gestures or multitouch inputs to perform a back navigation as many users don't use the keyboard much when browsing the web. In our QbS prototype, it is not necessary to even perform gestures since the result area as a whole acts as a back button.

3. As the drawing area and search options always remain visible and accessible, the user can directly modify the query and issue searches. Thus, there is no need to switch first back into a query screen to issue modified searches. Being a research prototype, we also find it important that there is no separate "advanced search options" screen required to get access the options – in particular the options for enabling invariances or using a different perceptual feature are always available at the bottom of the screen.

11.2.4 Visualization of the Used Perceptual Features

One of the greatest challenges in content-based image retrieval is to narrow the *semantic gap* and *sensory gap*.

The semantic gap, as defined in [Smeulders et al., 2000, p. 1353] and described in Chapter 5.2 is created through the different interpretation of the image by the human user of the system and the way the system perceives the image through the use of low-level features. Keywords that the user or other human annotators have assigned to the image may reduce this gap; also approaches to automatically analyze and classify the image content may help.

In contrast, the sensory gap, as defined in [Smeulders et al., 2000, p. 1352] and described in Chapter 2.3.3 exists due to the difference between real world objects and the way they can be perceived in an image. In QbS, the only information perceived of the real world are edges and colors in images. Notice, that due to focus on known image search, the sensory gap is already not as wide as in other tasks: The user is not searching for real-world objects, but for particular images of which the existence is known and both, the user and system, know the same representation.

What remains very challenging in QbS is the fact that the user has to express the mental image of the sought document through a sketch – and perceptual features of sketches and real images, in particular photographic images, differ significantly as explained in Chapter 10.3.3. The gap between the sketch and the content of the collection would not be as wide if the images in the collection would be sketches itself or images taken from cartoons or comic books or historic paper watermarks (cf. [Kreuzer, 2010, pp. 64–70] and Chapter 10.3.6).

We attempted to built the QbS system that allows the user to draw sketches as naturally and effortless as possible. Yet, best results in searches are clearly achieved when the user draws the sketches already knowing how the system will "understand" the sketch and the images in the collection. Providing visualizations of the extracted feature can help w.r.t. this goal: Letting the user see the images through the eyes of the system.

Whenever the user right-clicks on a result thumbnail or the actual image, a view as in Figure 11.9 is presented.[6] This shows the edge map of the image.

[6]On the stylus for the used Tablet PC, there are two buttons located where the thumb of the drawing hand rests. These buttons can work exactly like mouse buttons. Alternatively, pressing the stylus long onto the screen emulates a right-click. This functionality was already used in Section 11.2.2 for revealing the menus. When used with a keyboard, also holding down the Ctrl-key while clicking or tapping on an image also performs the desired action.

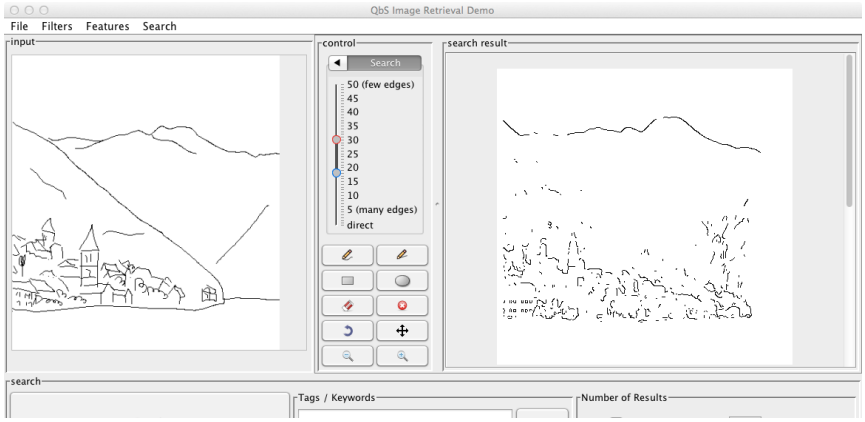


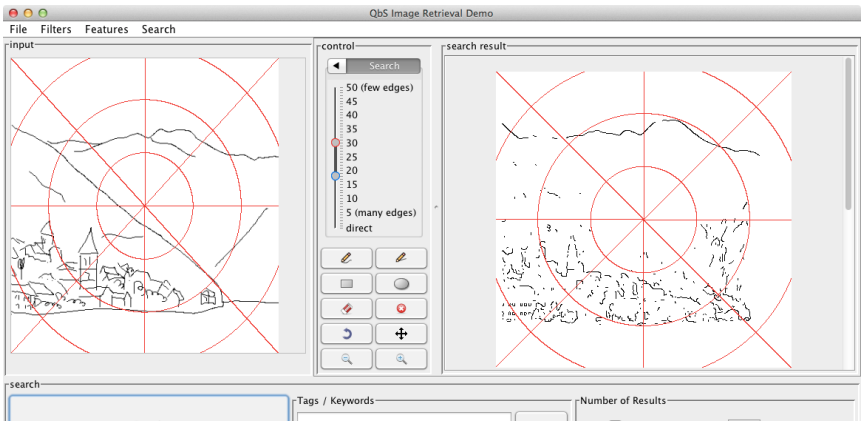
Figure 11.9: Edgemap View of the Top Ranked Result

From the features menu in the QbS application, the user can also activate an option to show the partitions that are used for Angular Radial Partitioning (ARP) and Angular Radial Color Moments (ARCM). These partitions are presented as overlays for both, the sketch of the user as well as the result image or result image edge map. Figure 11.10(a) shows the screen of the sketch and edge map for 8 angular and 4 radial partitions; Figure 11.10(b) for the much more fine-grained scheme using 16 angular and 8 radial partitions. The user can still modify the sketch, in particular rotate, scale, and translate it which will result in edges being placed in different partitions. Of course, the user should not need to adjust the sketch used in the search to this in order to find the sought image – the heuristics described in Chapter 10.3.4 are able to provide the needed invariance in most cases. But being able to "see" the sketch in a similar way to the system lets the user understand why and when such invariances are needed. Finally, Figure 11.10(c) shows in the lower right corner of the sketch and image how it is perceived in the lower resolution used for the Image Distortion Model (IDM). Notice that sketching at such a small scale would not be possible for almost any user and even imagining what an image looks like in 32×32 pixels is hard. Therefore it is much easier for the user to draw a sketch in the drawing area as described in Section 11.2 and let the system turn this into a representation suitable for IDM.

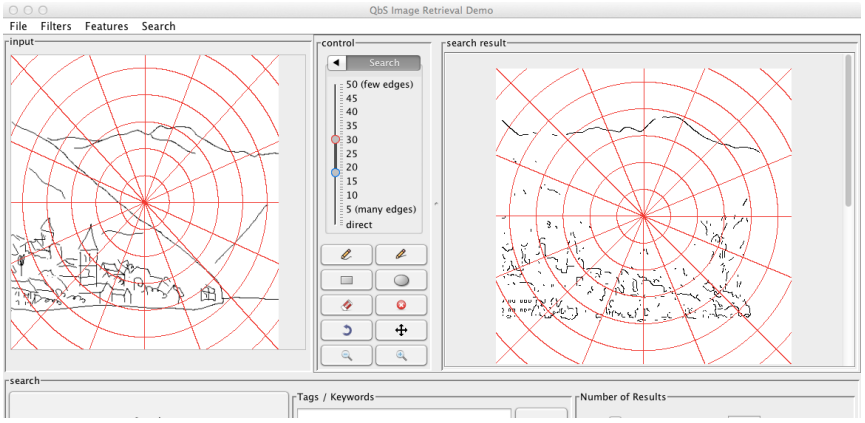
For convenience in the usage in experiments, the sketches can get saved to and loaded from disk. Furthermore, images can get loaded into the drawing area and turned into an edge map.This loading can be triggered simply by dragging any image and dropping it on the drawing area – for instance dragging of a search result thumbnails.

When a real image is loaded into the drawing area, the slider for controlling the edge detection enables to adjust β as visualized in Figure 11.11. The results always have the original image as the best-ranked result – which is not surprising at all as the features in the database have been extracted using the same transformation.[7] The ability to load a real image into the drawing area, of course, has never been used in the evaluation of our approach in Chapter 10.3.5 as this would be "cheating".

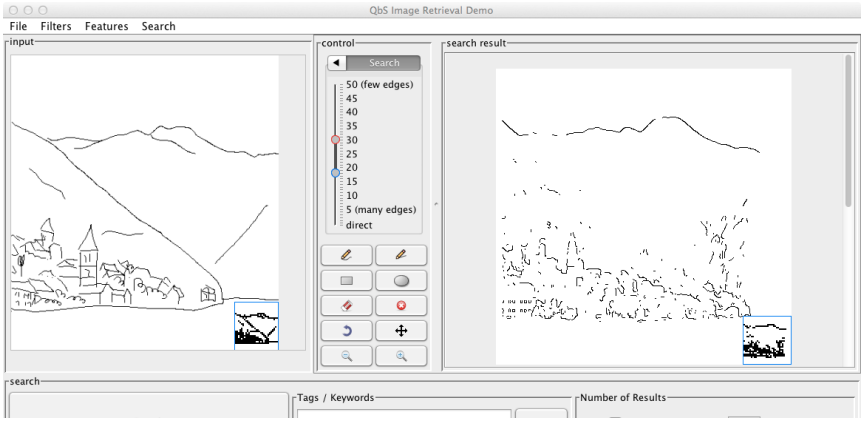
[7]The distance score shown next to the rank is greater than 0 just because the slider may not have matched exactly the same value and scaling artifacts.



(a) ARP 8/4



(b) ARP 16/8



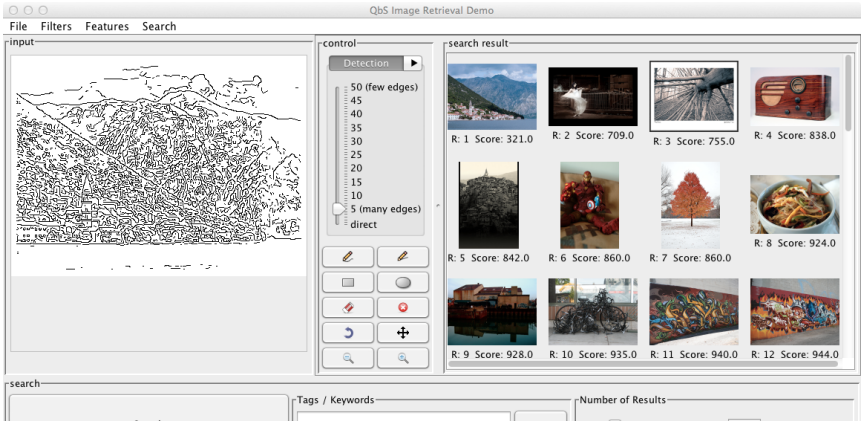
(c) IDM

Figure 11.10: Edgemap view of the top ranked result with overlays for 8 angular and 4 radial partitions in (a), 16 angular and 8 radial partitions in (b), and IDM with at most 32×32 pixels in (c).

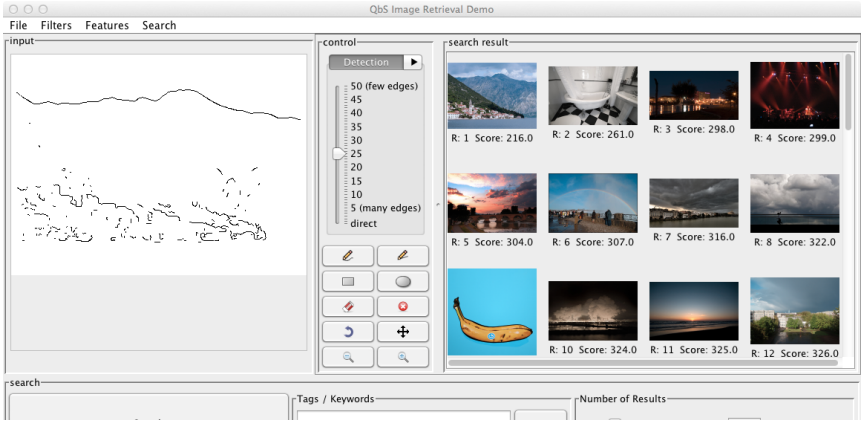
But what this tools reveals in experimentation such as in Figure 11.11 is, that other images in the results are very different depending on the value of β. This shows that the edge detection has very strong impact on the retrieval results and (a) that it would not be

possible to find a single β value that works well for all searches and (b) that it would not be acceptable if the user would have to select a single β value for the search. Therefore when the user has sketched an image as in previous screenshots, we allow to select a range of values for the edge detection. Using a broad range increases the invariance towards the detection of edges and therefore increases the matching tolerance; using a very small range reduces the invariance which increases the expressivity of the query.

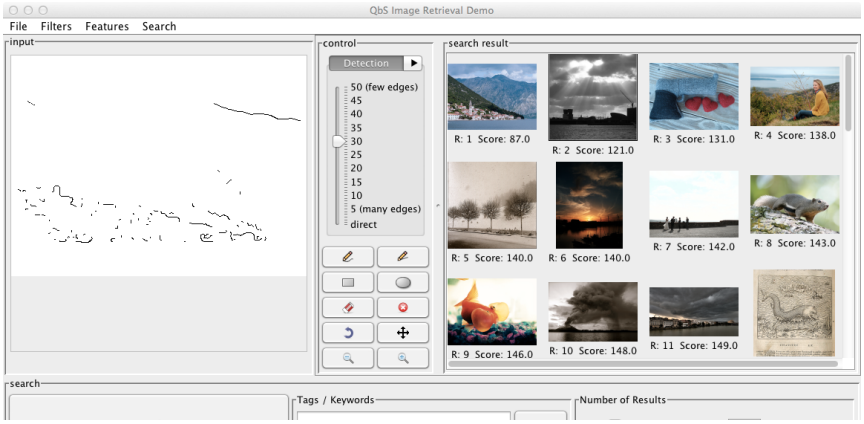
When an image has been turned into a query sketch using the edge detection slider, the range of β-values in search automatically gets centered around this β value. The user can overrule this selection by either enabling full edge detection invariance or switching into the search mode for the edge detection control. The last screenshots in Figure 11.12 show the negative effect when setting a very narrow range to either extremes: Detection of very many as well as very few edges. In both cases, the image from which the edges have been extracted at $\beta = 25$ is no longer among the top results. This illustrates that some invariance is always needed to cope with the sensory gap.



(a) $\beta = 5$

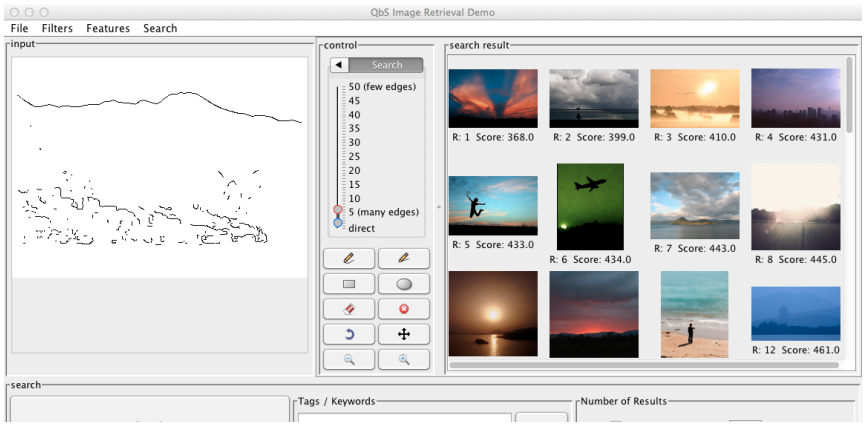


(b) $\beta = 25$

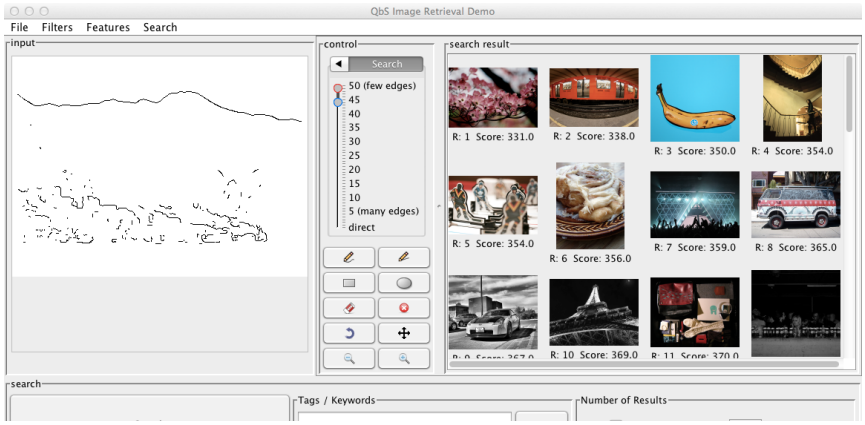


(c) $\beta = 30$

Figure 11.11: Search results for image im10853 turned into an edge map at various β values: While many edges are preserved at a β value of 5 as shown in (a), only very prominent edges are persevered at $\beta = 25$ as shown in (b) and even these diminish at even higher threshold values. In (c), also the edges between the sky and the mountains in the back start to disappear for $\beta = 30$. Some edges of the houses in the front of the image are still detected. This is due to the fact that they were focused when taking the image and even though the chosen depth of field was very long, the mountains in the back are still more blurred than the houses in the center of the focus.



(a) Search with β being 2 or 5



(b) Search with β being 45 or 50

Figure 11.12: Search results for image im10853 turned into an edge map with $\beta = 25$ and searching with other, narrow β ranges – these searches lead to bad results.

11.3 User Interaction using Interactive Paper for QbS

In order to make QbS more user friendly, we have added an interactive paper and digital pen interface [Kreuzer et al., 2012]. This is based on commercial pen and paper technology developed by Anoto[8]. In short, strokes written on paper are sent via the Bluetooth interface of the pen to a computer and integrated as sketch into QbS. The paper will only be used for input, the search results are still displayed on a PC.

11.3.1 Digital Pen Interface

Digital pens are designed for drawing on normal paper on which a proprietary and irregular dot pattern is printed. The pattern consists of very small dots arranged on a grid with a spacing of approximately 0.3 mm. Each dot can be placed on the pattern in four different ways: above, below, left or right of the center defined by the grid lines (as visible behind the letter A in Figure 11.13(b)). As soon as a user draws on paper, the pen which is equipped with an infrared LED camera can localize the position on paper by reading a 6×6 dot area on paper, corresponding to an area of 1.8×1.8 mm in size. By reading 6×6 dots, in total $4^{6\times6} = 2^{72}$ unique combinations can be supported. Therefore, the uniqueness of the pattern is ensured on 60 million km^2 (this exceeds the total area of both Europe and Asia). As the pen moves along the pattern, a camera and an infrared LED take digital snapshots of the local patterns at a rate of 100 fps.

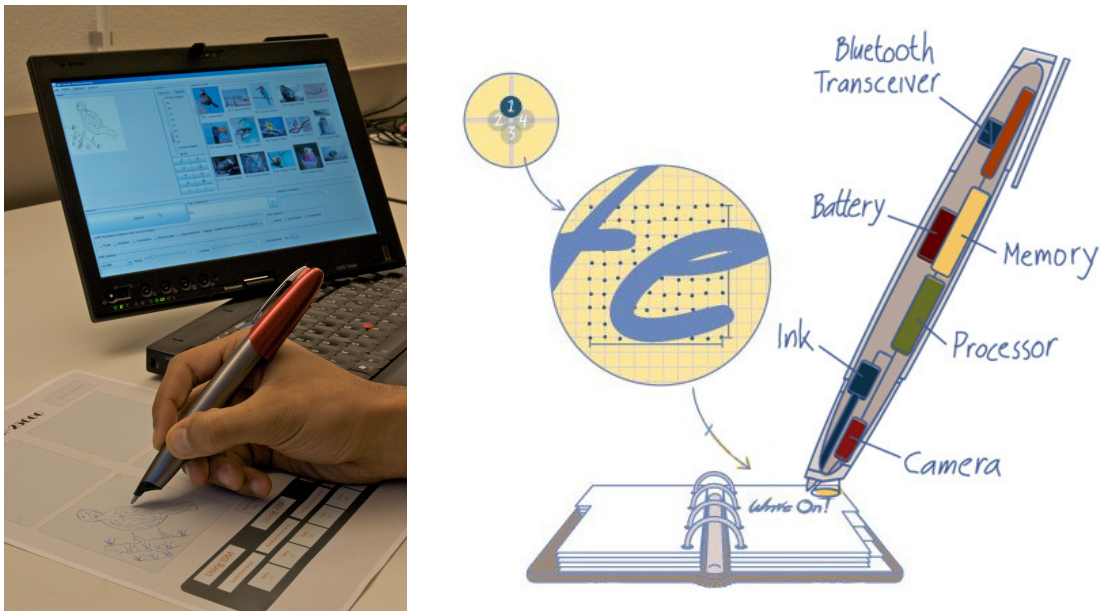
In addition, the pen has a pressure sensitive tip and a pen-down-message starts the transmission of the pen data. The pens store the pattern information in the form of pen stroke data, which are continuous curves made up of coordinates. The image processor calculates the exact position in the entire Anoto proprietary pattern. During image processing, snapshots are compared and information about how the pen is held is also gathered and stored. All the data from the image processor is packaged and loaded into the pen's memory, which can store several fully written pages. The pen strokes are transmitted to a computer via Bluetooth or via a USB connection [Koutamanis, 2005].

11.3.2 Linking the Paper Interface to the Application

The QbS interactive paper and digital pen interface consists of an executable (streaming client) that receives streaming data from the pen via a Bluetooth socket and forwards it to a server on which the QbS application runs. This can be either a local or a remote server, so it is possible to use a notebook supporting Bluetooth for receiving the pen data and run the QbS application on a more powerful desktop computer. The components involved in transferring pen strokes to an application are shown in Figure 11.14.

QbS allows pattern pages to be printed with any desired interface, as long as the Anoto pattern is not rendered invisible to the pen. All functionality is achieved by using XML configuration files. Only the global Anoto coordinates of the start page and the layout of the used pattern pages must be known in advance. By setting these values in the configuration file, areas with specific functionality can be defined by their actual local coordinates for all the pages used. Through the use of the XML configuration files,

[8] http://www.anoto.com



(a) Interactive Paper (b) Digital Pen (www.anoto.com)

Figure 11.13: Digital Pen and Interactive Paper for Sketching Visual Examples

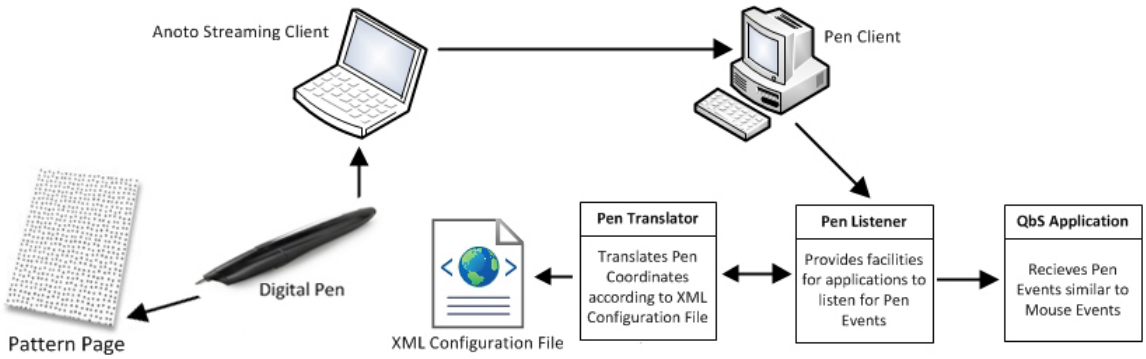


Figure 11.14: Dataflow – from Interactive Paper to QbS

users are able to employ the same pattern pages for multiple paper interfaces. Hence, the QbS pen and paper interface can easily be used for other applications as well.

11.3.3 Digital Paper Interface for QbS with the MIRFLICKR-25000 Collection

Figure 11.15 finally shows the paper interface that has been used for the evaluation in [Kreuzer, 2010, Kreuzer et al., 2012]. The overall design has to deal in part similar challenges as for Tablet PCs in Section 11.2.1 as also the digital pen will use absolute positions, therefore the pen has to be lifted and placed on any UI element that should get invoked. For this reason, the most used button –the search button– is placed twice

on the paper interface: One search button in top right position, one to the lower left of the drawing area.

What is significantly different from any screen interface that the user interface on paper and any sketch drawn by the user cannot be changed. Therefore not many drawing tools are fully meaningful for interacting on paper, in particular an erase tool is of limited use as the drawn sketch on paper will not be removed. Instead of erasing an entire sketch, the user can just use the next drawing area on paper. Erasing part of the sketch would only affect the representation inside the application that is still shown on a screen. Therefore it makes sense that any such tool is applied not on paper, but on the PC,that is needed anyway for sketch-based retrieval to present the search results.

Similarly, for any other functionality that is easier to use on a PC, there's no need to represent it on paper – leaving more space for the user interfaces that can be effectively used on paper, in particular drawing areas. This also leads to a situation, in which any knowledge about the intended search task and searched collection can and should be incorporated in the design of a paper interface. In particular, for the MIRFLICKR-25000 collection, several options for added invariances have shown to improve search results, in particular the β parameter to adjust edge detection. Such options should be made available to the user and are therefore included in the bottom area of the paper interface in Figure 11.15. For other collections with different content like cartoon images and paper watermarks used in [Kreuzer, 2010], these options have not shown to improve the results as the images in the collection are much closer to user-drawn sketches, therefore the sensory gap is not as wide and less options for invariance are needed in common searches. As a consequence, the paper interfaces developed in [Kreuzer, 2010, p. 49–53] for these collections provide less options to enable invariances or set specific β values for edge detection.

However, a flexibility that interactive paper does provide and should be used whenever needed: Any application can use more than a single sheet of paper and switching between different sheets can be as simple as placing the two sheets next to each other on a desk, such that both papers and there active areas like buttons to control options are visible simultaneously. The area available for the user interface is therefore easily expandable, which can be used to provide different sheets of papers with different options.

For sketching, this can be used (and has been used in [Kreuzer, 2010]) to provide drawing areas with different aspect ratios. When the user starts drawing a query sketch for a known image, selecting a drawing area with similar aspect ratio to the known image eases the placement and relative sizes of drawn objects significantly.

For search options, this allows to provide many or all of them without the need to hide them in hierarchical menus: Instead of several such menus, options can be laid out in flat hierarchies and the needed structure achieved through different sheets of papers.

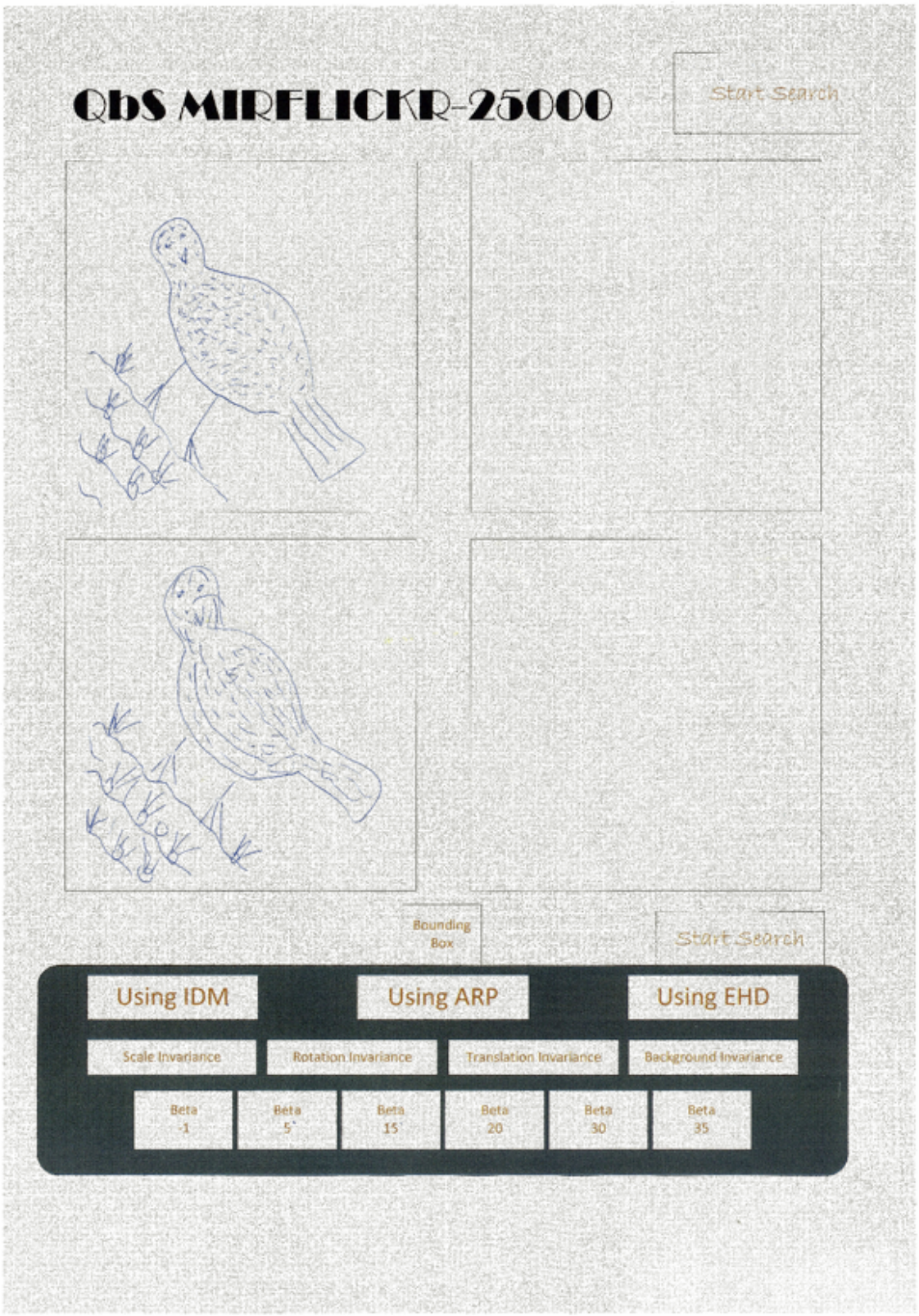


Figure 11.15: Paper Interface for the MIRFLICKR-25000 collection

11.4 Conclusion on User Interaction Implementation

For image search tasks, user interaction is always of great importance. In this chapter, we have given some insights on how user interaction can be implemented for image search in digital libraries. We presented a number of concepts and techniques that we have used and refined for the use in image search tasks that act on different levels:

- On one hand, the fairly generic and versatile concept of an *ObserableIterator* , that makes it very simple to implement features like monitoring the progress of a long-running similarity search or batch feature extraction as well as enabling the user to abort the long-running activity. And this concept can be integrated or even be the core of predicate filtering, prefetching, and multi-step approximate search.

  The *ObserableIterator* has been used in all three of the prototypical applications that have been built for solving particular image-related search tasks.

- On the other hand, we identified peculiarities in using novel input devices for image search tasks, including challenges and opportunities. The novel input devices includes in particular Tablet PCs as well as digital pens and interactive paper. Emphasize was given to tasks that involve drawing sketches – as this is the area, where these new novel input devices can excel. This is also the area, in which traditional input devices like mouse and keyboard are not adequate, since they limit too much the ability of the user to express a precise query and therefore have been a major cause of query by sketching approaches not finding greater acceptance.

  Since novel input devices make query by sketching a more appealing search paradigm, the focus can then be shifted towards which tasks can be effectively supported by this paradigm and how interfaces for these tasks have to be built when mainly used with such an input device. For both input devices, Tablet PCs and digital pens, moving the device far distances over the interface has to be avoided and in particular, traditional menus that are used in graphical and text-based screen interfaces are critical to use. Alternatives to such menus have been presented:

  - For Tablet PCs, Pie Menus can provide an alternative that can be used elegantly in drawing applications.
  - For interactive paper, the number of times a particular sheet of paper can be reused is limited since the traces of previous uses (in particular drawn sketches) cannot easily be removed. Therefore new sheets of interactive paper have to be printed at some point in time – and these can be adapted to the particular task and collection as reusability is limited anyway and printing interactive paper is fairly inexpensive. For options that are not used frequently, separate pages can provide additional space for the user interface.

These observations and experiences allow to design and implement ways for the user to interact with the system effectively and pleasantly.

12

Summary of Implementation, Usage, and Evaluation

We have presented our implementation w.r.t. the three main building blocks *Content Management*, *Query Formulation and Execution*, and *User Interaction* and have shown, how they can assist the user in solving image-related search tasks. Our implementations are based on the thorough analysis of conceptual building blocks in Part II which helped in developing solutions that are not limited to single task, but that are reusable and adaptable for new tasks.

- We have implemented and investigated different approaches to show how content of various scales can be managed in Chapter 9. The two extremes that we presented have been:

  - The layered implementation used in DILIGENT that can be scaled on Grid resources.
  - Specialized implementations to handle small-sized, static benchmark collections.

  By providing several implementations, a broad range of the spectrum is covered and the discussion highlights benefits and drawbacks of each of them. The *Generic Storage Model* has served as a model for discussing the differences in the ability to handle structured content of the different implementations.

- For the execution of queries in Chapter 10, the search primitives *k*NN search and range search can be used for a wide variety of tasks. The choice of appropriate perceptual features and distance functions are essential to deliver satisfying results.

  We have shown how enabling the fine-tuning of the allowed matching tolerance can improve the retrieval quality in automated medical image classification, sketch-based known image search, as well as retrospective geotagging of images based on finding images which have already been tagged. The quality improvement were mainly achieved by:

  - Increase local context in IDM and adjusting the used *k*NN classifier.
  - Adding new options for invariances to scale, translation, and edge detection to ARP and apply IDM to sketch-based retrieval with the same invariance to edge detection. All edge features can be combined with keyword search, color moments, and EHD as additional search clues.
  - Allow the user to select areas on map, regions of interest on images, and allow to limit keypoint matches heuristically to certain rotations.

  We have provided an overview of potential techniques to reduce the execute time of searches. Out of these, we analyzed in depth how the presented Early Termination Strategy, use of multithreading, and caching can reduce the execution time in any of the application domains without any degradation in retrieval quality. Using the Early Termination Strategy alone achieved a speedup up to a factor of 4.86. Combining it with multithreading on a 8-way server achieved a speedup of 37.34, which allowed the use of the computationally expensive IDM even with big local contexts. Applying all three approaches enables interactive applications like QbS to return search results to common queries in interactive response below 200ms.

- The interaction with the user has to be adapted to the task in order to deliver good user experience. In Chapter 11 we presented on one hand some implementation techniques to provide interaction functionality such as to monitor the progress of a search as well as to abort. On the other hand, we also show how novel input devices can be used to enable the user to exploit query by sketching for known image search.

After we have now investigated and evaluated particular building blocks in isolation, we want to shift our view on systems as a whole.

Part IV

Systems

13

Overview of Existing Systems

In Part II we have identified the main functional building blocks that are required to build complete systems: *Content Management* (Chapter 4), *Query Formulation and Execution* (Chapter 5), and *User Interaction* (Chapter 6). During the last two decades, many systems have been developed. Most of them focus on very particular functionality – as they have been be targeted for particular user communities and satisfy the needs of different research communities.

Figure 13.1 tries to give an overview – reusing again the illustration of the building blocks involved in the similarity search process introduced in Figure 3.1.[1] It tries to place the general work areas in which systems have been developed near the conceptual building blocks that they are mainly focussed on. In the remainder of this chapter, we will highlight some of the systems that have been developed in the respective area, but also describe why many of the systems do not provide everything that would be needed to be considered a complete digital library system with support for image-related search tasks.

13.1 Digital Libraries

Traditionally, digital library research has focussed on the management and access to text document. Modern system allow to add content of arbitrary type; however, the search functionality remained very much text- and metadata-driven. As a consequence, exact matching and very limited matching tolerance to cope with small deviations like spelling errors are the common approaches for these systems. Figure 13.2 illustrates this focus using the Image Task Model (ITM).

[1] Of course, this overview is a strong (over-)simplification and cannot represent entirely all the aspects.

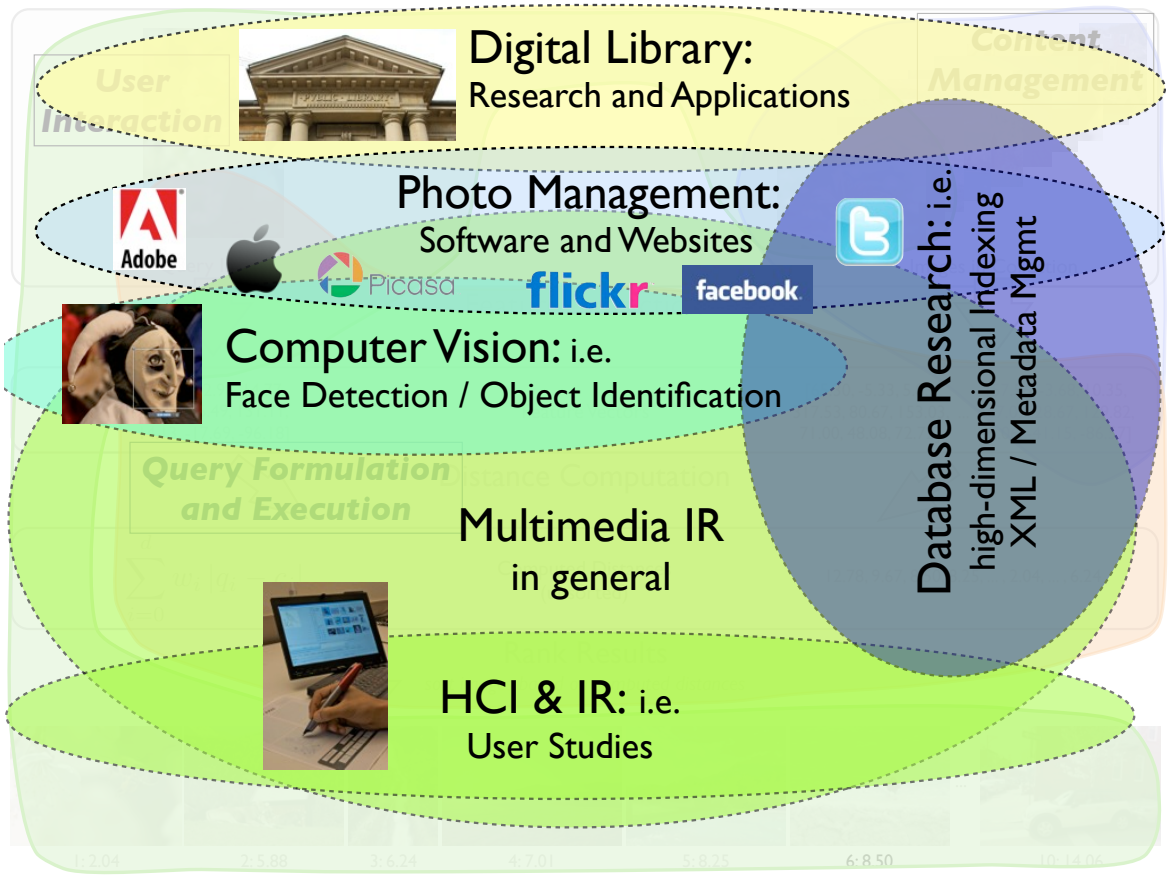


Figure 13.1: Rough grouping of areas of work in relation to the building blocks involved in the similarity search process of Figure 3.1: Areas do overlap a lot, no single area covers all aspects of Content Management, Query Formulation and Execution, and User Interaction.

Examples of such systems are digital library software packages Greenstone[2], DSpace[3], Fedora[4], OpenDLib[5]. Other examples are EU-funded research projects like BRICKS[6] and DRIVER[7].

Only few systems explicitly consider similarity search and therefore provide a wider range of matching tolerance for visual content. One of them is MutliMatch[8]. It is focused on cultural heritage [Amato et al., 2008, Amato et al., 2009] – but it is generic in the sense that it is not focussed on particular tasks. The managed content consists mainly of crawled content, which includes images, still images generated from movies, movies, and audio. Strong emphasize has been put into developing suitable ontologies for the domain that enhance the management of the content of the digital library and allow

[2] http://www.greenstone.org/
[3] http://www.dspace.org/
[4] http://fedora-commons.org/
[5] http://opendlib.iei.pi.cnr.it/home.html
[6] http://www.brickscommunity.org/
[7] http://www.driver-repository.eu/
[8] http://www.multimatch.eu/

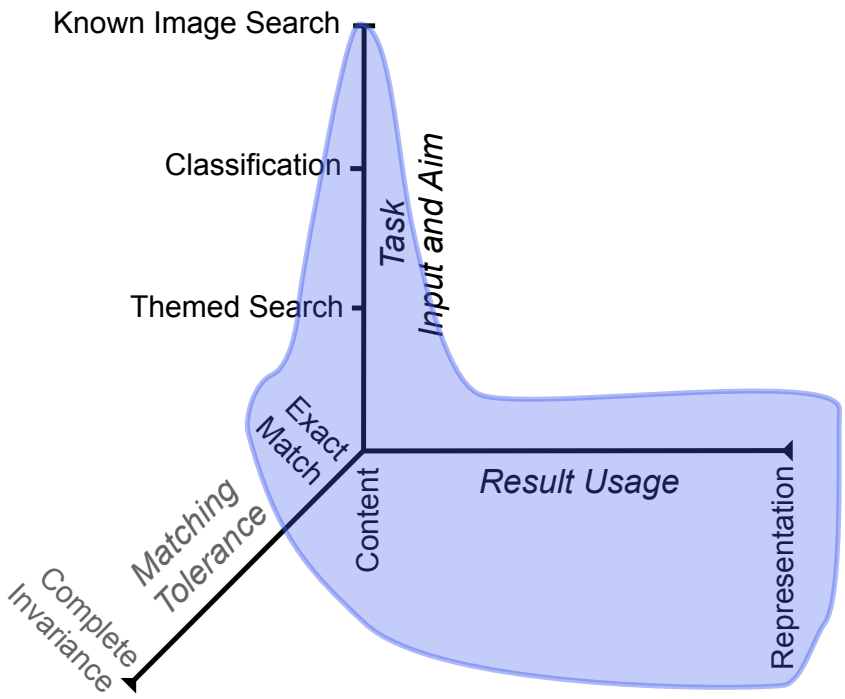


Figure 13.2: Aspects in ITM covered by common digital library management systems.

text- and speech-based queries. In addition, also similarity search for the particular media types are supported – alone or in combination with text-queries.

The uBase system[9] has been described in [May, 2004, Browne et al., 2006]. It allows the management of hierarchical image and video collections and text-based and visual search in them. Particular emphasize was given to the user interface design, for which NN^k networks [Heesch and Rüger, 2004, Heesch et al., 2006] are used. These networks place documents in a graph, where nearest neighbors w.r.t. at least one of the k perceptual feature are connected.

Mainly dedicated to aspects of content-based image retrieval on very large scale is SAPIR, another EU-funded research project [Falchi et al., 2007]. As a showcase, more than 50 million crawled images from Flickr have been collected and five different MPEG-7 features extracted and indexed [Batko et al., 2010]. Indexing and searching is performed using MESSIF[10] and a peer-to-peer infrastructure MUFIN[11].

IM3I[12], in contrast to some of the systems for crawled content, has a strong focus on the management of multimedia content including curated collections and automatic annotations and making the content available through flexible user interfaces that can

[9] `http://technologies.kmi.open.ac.uk/ubase/` and `http://mmis.doc.ic.ac.uk/demos/ibase.html`

[10] MESSIF: Metric Similarity Search Implementation Framework, `http://lsd.fi.muni.cz/trac/messif/`, [Batko et al., 2007]

[11] MUFIN: Multi-feature Indexing Network, `http://mufin.fi.muni.cz/tiki-index.php`, [Batko et al., 2010]

[12] IM3I: Immersive Multimedia Interfaces, `http://im3i.in-two.com/`

be adapted to the particular needs inside the authoring process [Bertini et al., 2011]. It also uses MPEG-7 descriptors for content-based image retrieval, but also provides a fusion of MSER and SURF interest points with a Pyramid Match Kernel [Grauman and Darrell, 2007] for automated video annotation using the bag-of-(visual)-words approach like [Sivic and Zisserman, 2003]. It is implemented using a Service-Oriented Architecture (SOA).

Two other generic digital library systems that support similarity search for images and are implemented as a SOA will get covered in more details in Section 14.1: Delos-DLMS and gCube for which the author of this thesis has been actively involved in the development.

13.2 Photo Management and Sharing

Within the last five to ten years, digital cameras, affordable disks with high storage capacity, and fast internet connections have become almost ubiquitous. And with this technology reaching end users, also the area of managing and sharing digital pictures has gained attention that used to be of interest only to a fairly small group of image professionals.

Photo management applications like Adobe Photoshop Lightroom[13], Apple Aperture[14], Bibble[15] still target the market of professional photographers, but also enthusiastic non-professional home users that have also adapted to a nondestructive workflow to process the images that they collect in huge numbers mainly with digital SLR cameras in RAW format. These applications provide in addition to the fundamental image processing capabilities also sophisticated collection management and the ability to assign and edit image metadata in various formats. More consumer-oriented software like Apple iPhoto[16], Google Picasa[17], digiKam[18] make such functionality available to a even wider range of camera users.

In combination with photo sharing websites and social networks like Flickr[19], Facebook [20], Twitter [21], ImageShack / yFrog [22], Google Picasa Web Albums / Google+[23], and Wikimedia Commons [24] these systems provide much of the functionality that would also be found in digital library systems, in particular w.r.t. adding documents, managing metadata, sharing the available information with other users. What is certainly

[13] http://www.adobe.com/products/photoshoplightroom/

[14] http://www.apple.com/aperture/

[15] http://bibblelabs.com/, recently acquired by Corel and integrated into a new product named AfterShot Pro http://www.corel.com/aftershotpro

[16] http://www.apple.com/ilife/iphoto/

[17] http://picasa.google.com

[18] http://www.digikam.org/

[19] http://www.flickr.com

[20] http://www.facebook.com

[21] http://twitter.com

[22] http://imageshack.us/, http://yfrog.com

[23] http://picasa.google.com, http://plus.google.com

[24] http:/commons.wikimedia.org

different is the target audience and the exclusive dedication to photos (and video) instead of any kind of document in digital library systems.

Due to this dedication to a particular media type, the systems are also more likely to adapt to functionality needed for similarity search. For instance, many of the applications and websites to manage pictures provide now functionality to automatically detect faces and assist through automatic recommendations of names the process of annotating images. For the latter, not only object detection, but also object identification functionality is required. imgSeek [25], retrievr [26] and digiKam even provide the ability to search for images using a color sketch based on multi resolution wavelet decompositions [Jacobs et al., 1995].

Compared to the digital library community, much less of the development and advances in research result in scientific publications. However, this area has gained great importance for the content-based image retrieval community as many people openly share the pictures they took and annotated. Only with the huge corpora of annotated images that these websites made available without too many copyright issues enabled large-scale machine learning and benchmarking of image classification and retrieval in research communities.

13.3 Multimedia Information Retrieval

This is a very broad, inhomogeneous community. In particular during what has been called the "early years of content-based image retrieval" [Smeulders et al., 2000], many small prototype systems have been developed. [Veltkamp and Tanase, 2000, Veltkamp et al., 2001] collected more than 40 systems including the system website, references in literature, used perceptual features, supported query modes, distance computation / matching, indexing, result presentation, relevance feedback, and other details about the application. Notably absent from the analysis of functionality is everything related to content management: While the building blocks of query formulation and execution as well as user interaction has always been where these applications demonstrated their contribution to the state-of-the-art, content management was not of great interest – in particular for research prototypes that were used with static collections.

Famous examples of early systems in this area are QVE [Hirata and Kato, 1992], QBIC [Niblack et al., 1993, Flickner et al., 1995], PicHunter [Cox et al., 1996] and [Cox et al., 2000], MARS [Rui et al., 1997], PICASSO [Del Bimbo et al., 1998], Blobworld [Carson et al., 1999, Carson et al., 2002], and SIMPLIcity [Wang et al., 2001] – not necessarily because these systems would have found strong usage outside the research community[27], but mainly because of their original contribution to research:

- Being one of the first, sophisticated approaches to address content-based image retrieval: QVE, QBIC

[25] `http://www.imgseek.net/`

[26] `http://labs.systemone.at/retrievr/`

[27] The widest adoption outside the research community has probably been achieved by QBIC, which was integrated into IBM's DB2 Image Extender (cf. `http://wwwqbic.almaden.ibm.com/`).

- Being first to present all aspects of a complete system including application architecture and high-dimensional index structures: QBIC

- Introducing novel interaction paradigms: sketch-based retrieval in QVE, relevance feedback in PicHunter and MARS

- Novel perception of the images: segmented color regions in PICASSO and Blobworld which also allowed to respect spatial relationships, SIMPLIcity for its focus on semantics

- and sometimes even more for the added theoretical foundation rather than technological advances like the classification of image searches into the three broad categories *Target Search*, *Category Search*, and *Open-Ended Search* in the PicHunter publications

Today, most of the early prototype systems are no longer available. New systems have appeared, but the initial enthusiasm to build immediately entire systems to present what the systems are capable of seems to have cooled down significantly. This is certainly also influenced by the way the approaches were evaluated back then and how it is done today:

- In the early days no generally agreed benchmark collections did exist and in the lack of better collections with some annotation, most people used either there own small test collections or –unfortunately different– subsets of the Corel Photo CDs as those could be acquired for a fairly small price and reproduced in publications without many copyright issues.

  As different people used different sets of images for the evaluation, the only way to give some impression of the quality of the retrieved results was showing them. Presenting a screenshot of a running application captured more of this than just formatting the result list as a table in the scientific paper – and, together with a link to the running system on the web, added a little more credibility as every reader of the paper could visit and try out the system. The latter was not free of problems, as the systems had to remain available and not change too much compared to what was presented in the paper. Furthermore, on one hand the system had to handle peek loads when many people heard about the system at some conference, on the other hand this link to the system could not be provided for the blind reviewing of publications where it would have benefited most from the added credibility.

  Another issue with presenting systems as a whole is, that unless the used methods are described in some more technical papers, there is usually not much space left inside the publication to explain them in detail. Without such a detailed explanation that includes all steps of the image processing pipeline, the precise extraction of features and formula to compute the (dis-)similarity and ranking, algorithms to retrieve fast results, nobody was able to reproduce the works on his own test collection to compare the performance of different approaches.

- Today, there are established benchmark collections for various tasks, for instance from ImageCLEF[28] such as in [Clough et al., 2005, Nowak et al., 2010] and PASCAL Challenges[29] as in [Everingham et al., 2010].

  Of course, a publication can still contain screenshots, but for members of the community, numbers and graphs of the performance of a well-known dataset are much more important. Therefore there is not that much a need to describe the system as a whole; it's sufficient to describe clearly the used methods, which dataset is used, and how the approach performs compared to other existing works.

  For the scientific community, this clearly is how empirical analysis should work and has worked for research communities where sharing the used datasets had less copyright and privacy issues than images. And research on feature descriptors, (dis-)similarity measures, and result ranking has clearly improved and made much faster progress through the general availability of standardized benchmark collections.

  However, as the necessity to present an entire system decreased, also the work invested into the design and implementation of such systems has decreased or became at least less visible. And this may have also led to a situation, in which systems have a lesser chance of maturing.

Multimedia Information Retrieval reuses and depends on techniques that may have been developed in a different context. For instance, techniques from the research area of computer vision, databases, and human-computer interaction. For evaluating the ability to perform particular image-related search tasks, it is in many cases necessary to have complete systems that provide at least basic functionality for content management, query formulation and execution, and user interaction – which is frequently not the case for the original areas in which the used techniques have been developed.

13.3.1 Overlap with Computer Vision, Pattern Recognition, and Machine Learning

For computer vision, it is inherent in the overall assumption that a human user will not be or only partially involved, for instance, in supervised learning. Therefore much of the research leaves aside all aspects of user interaction. For building complete systems for a digital library system, the approaches from computer vision can be reused, but the aspects of user interaction have to be added. [Del Bimbo, 1999, p. 6] points out that the existence of a user in the retrieval loop underlines the importance of flexible interfaces and visualization tools.

For instance, many of the approaches for object detection and identification have been motivated by the desire to let computers and robots perform repetitive, tiring tasks that had to be done by human workers before. After an initial training of the systems, they should act autonomously and detecting the units to work on from a camera signal

[28] http://www.imageclef.org/

[29] http://www.pascal-network.org/?q=node/15

if frequently a prerequisite for performing a task. Other examples are related to surveillance and biometric authentication – also here the aim is to use computer systems to reduce the need for human labor.

For classification tasks, much progress has been achieved in terms of reducing the error rate through less false negatives (not detecting instances of a class present in an image) and less false positive (mistaking instances of other classes). This can build an important foundation for systems to solve *Image Classification* and *Retrieval by Class* task, but also to assist faceted search for *Known Image Retrieval* and *Themed Search*. However, classification is a different task than ranking based on similarity that is frequently closer to the aims of a *Themed Search*. For the latter, the application of machine learning techniques to relevance feedback as described in Chapter 6.3.6 can also be very beneficial.

Many of the systems developed in this area of research do not deal with aspects of content management. In particular for early research prototypes in this domain, MATLAB is frequently used as a convenient programming and visualization tool and only computational heavy parts are implemented in other languages (mainly C/C++), for instance the SIFT Keyoint Detector demo software[30]. For visualizing results and measuring the success of an approach in terms of error rate, there will frequently not be a need to implement more functionality of content management than what can easily be achieved with MATLAB, command line scripting, or simple programs processing images directly from the filesystem. As a user interface, simple means to load images, adjust parameters, show effects of the processing of images, and ultimately draw some error graphs.

More complete systems can therefore be found today among photo management software and websites mentioned in Section 13.2 that reuses approaches from this domain, in particular face detection and identification capabilities for implementing "face tagging" as mentioned in Chapter 5.1.2 on page 104. Figure 13.2 tries to highlight this aspect through the overlap of the Photo Management and Computer Vision area – that also touches Database Research.

13.3.2 Overlap with Database Research on High-Dimensional Index Structures

One area, where much of the attention on Multimedia Information Retrieval went into particular aspects with measurements of performance that did not require complete systems of even displaying images at all was Database Research, in particular the area of high-dimensional indexing. From the early days of content-based image retrieval it was obvious, that similarity search on huge collections could only be successfully used by a broad audience when the results to queries could be delivered fast enough and this area provided the tools to achieve this goal.

Therefore this domain has delivered important insides, but not many complete systems: To measure the success of a method, only the time matters that is needed to deliver the same results for performing the similarity search given the features extracted from a dataset and a query feature. And for this, the experiments can be performed either

[30]Demo software is available at `http://www.cs.ubc.ca/~lowe/keypoints/` for SIFT [Lowe, 2004].

using real datasets consisting of feature vectors extracted from a benchmark image collection or synthetic data that mimics similar value distribution characteristics. It is not required to implement and describe of a full digital library system with similarity search to evaluate which data structure can perform the task in least time.

This mostly holds also for approximate similarity search methods [Patella and Ciaccia, 2008] which will return inexact, but very close answers in significantly less time, thus allowing to further speed up the query execution at the cost of retrieval quality like [Weber and Böhm, 2000], locality-sensitive hashing (LSH) [Datar et al., 2004], the Best-Bin-First strategy in a k-d-tree [Lowe, 2004], approaches based on *visual codebooks* mentioned on page 326 and using the Euclidean distance as a sieve for the Image Distortion Model [Keysers et al., 2004] or angular-radial partitioning (ARP) for a similar purpose on page 347. However, as these approaches always require a trade-off between accuracy of results and the time to determine them, it is not sufficient to evaluate these approaches only based on the retrieval time, but also need to proof that the approximated results are still suitable to let the user successfully finish the image-related search tasks.

13.3.3 Overlap with Database Research on Metadata Management

Another important aspect of complete digital library system that benefits from research from the database community is management and transformation of metadata. In particular XML is frequently used for storing and exchanging either bibliographic and bibliography-like metadata as in MARC-XML or OAI-PMH [Van de Sompel et al., 2004], image metadata as in XMP [Tesic, 2005], or rich metadata that may even contain extracted perceptual features as in MPEG-7 [Chang et al., 2001]. In the area of database research, support to store and querying has been developed. Furthermore, when several data sources are combined, data integration becomes an issue. However, also in this area, the development of complete digital library systems with support for image search is commonly not needed to measure the quality of an approach.

13.3.4 Overlap with Human-Computer Interaction

This is the area, in which there is the greatest need to work with complete systems. For particular aspects for instance in result presentation may be possible to evaluate in isolation; whenever the effectiveness of a system is measured for users performing a particular task, it needs a complete system.

Unfortunately, as mentioned already in Chapter 11 little research has really dealt with user interfaces for content-based image retrieval as reported by [Jörgensen, 2003, van den Broek et al., 2004]. And probably it is also caused by the mutual dependence: Without the availability of modular and fairly stable CBIR systems that deliver good search results, it is almost impossible to perform in-depth evaluation of various user interfaces. And without such an intense usage of CBIR systems in user studies, there is less need to develop modular CBIR systems as they can be implemented as a "black box" that is just used as-is without many subsequent modifications – at least for research

prototypes for which commonly little resources are available for long-term support of a system once the initial research results have been published.

The main aspect of user interaction that has created strong interest so far has been relevance feedback. However, while systems like PicHunter [Cox et al., 1996, Cox et al., 2000] and MARS [Rui et al., 1997] received much attention and raised hopes to reach a level of sophistication in content-based image retrieval, the interest in relevance feedback techniques has gradually declined. More recent attempts like [Jing et al., 2004] and [Shrivastava et al., 2011] have not yet recreated a strong interest in this aspect of user interaction. In our understanding, this might relate to the critical issues that do exist in relevance feedback that we described in Chapter 6.3.4 – 6.3.6 which might have been underestimated initially and therefore resulted in too many frustrating end-user experiences.

14

Detailed Discussion of Systems

In this chapter we will provide some in-depth knowledge on the implementation of particular systems. One important aspect in this discussion will be which areas are covered by the systems and to what degree. The major difference exists here between systems that are generic and therefore try to cover a broad spectrum of potential user tasks, and those that are dedicated to specific user tasks. Generic systems will be presented in Section 14.1, specialized systems follow in Section 14.2. For both, we will again use the Image Task Model (ITM) developed in Chapter 2.5 to characterize them.

Another aspect of interest will be the modularity and extensibility of the systems, as these properties have strong impact on the reusability. This aspect will mainly be addressed in the comparison that concludes this chapter in Section 14.3

14.1 Generic Systems

In this section we will present two systems that provide a complete set of functionality including similarity search which do not target particular tasks, but basically attempt to provide enough flexibility to serve the entire space of image-related search tasks as illustrated in Figure 14.1.

14.1.1 ISIS and its extension to DelosDLMS

ISIS stands for "Interactive SImilarity Search" [Mlivoncic et al., 2004a], has originally been developed in the Database Research Group of ETH Zürich and has its roots in a system named Chariot [Weber et al., 1999]. It has subsequently been maintained and extended by our group. [Brettlecker et al., 2007, Agosti et al., 2007] ISIS is based on a Peer-to-peer distributed workflow engine named OSIRIS [Mlivoncic et al., 2004a, Brettlecker et al., 2007, Schek and Schuldt, 2008].

ISIS supports several features for images, video, and audio and fast retrieval based on the VA-File [Weber et al., 1998]. ISIS has been extended during the lifetime of the DELOS project to the DelosDLMS, which added new collections and media types like medical images, 3D shapes. [Agosti et al., 2007, Ioannidis et al., 2008].

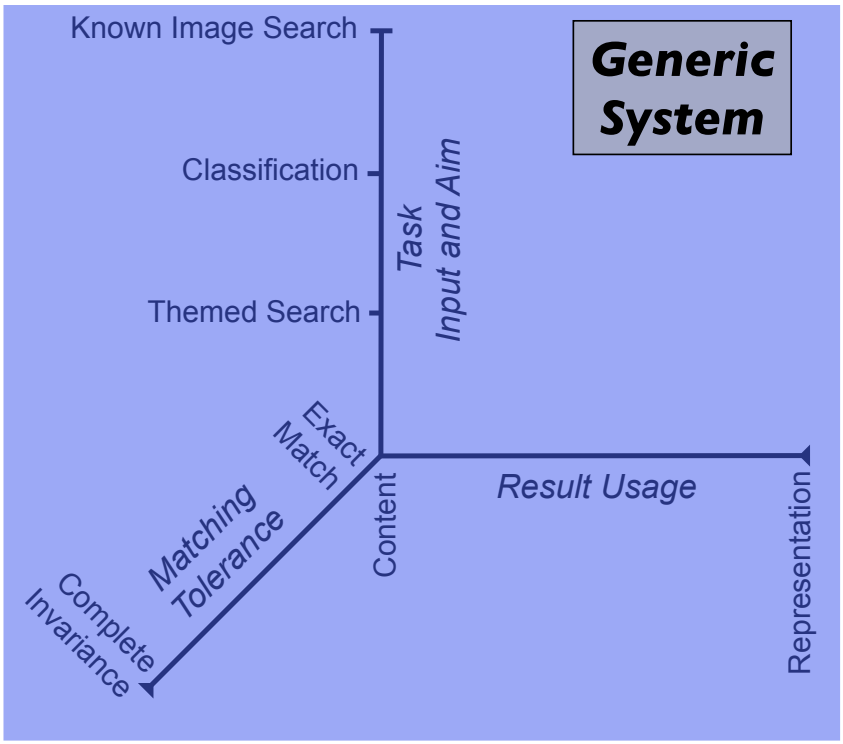


Figure 14.1: Aspects in ITM covered by a generic system: all kinds of tasks, wide range of matching tolerance, any content- or representation-oriented result usage.

With various interfaces for query by example and faceted search as well as displaying the entire collection in a self-organizing map (SOM), the system supports well exploratory search tasks as they are common in themed searches. It even explicitly considers the extreme case of matching tolerance –random results– for this purpose.

DelosDLMS has been used as a showcase for the research performed in the context of the EU-FP6-funded DELOS Network of Excellence. It is therefore intended to give an impression what state-of-the-art techniques from individual research areas can provide when being integrated into a complete system.

The following paragraphs will highlight how this has been achieved technically.

OSIRIS: Distributed Infrastructure for Processes

OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) [Schuler et al., 2003, Schuler et al., 2004] is a platform that allows combining different distributed services into processes. The OSIRIS platform itself does not provide any application functionality but, by combining specialized application services, supports the definition and reliable execution of dedicated processes (this is also known as "programming-in-the-large"). When different specialized digital library application services are made available to the OSIRIS platform, users can define and run powerful digital library processes by making use of these services. OSIRIS processes themselves are wrapped by a service interface. Therefore, a process can be invoked just like any other service (and used in other processes as well).

Following the model of transactional processes [Schuldt et al., 2002], processes in OSIRIS contain two orders on their constituent services: a (partial) *precedence order* specifies regular execution while the *precedence order* is defined for failure handling purposes (alternative executions). Data flow between services of a process can be defined independently of control flow. Activities in a process are invocations of application services. Ideally, the transactional behavior of each application service is known. This transactional behavior includes information on compensation (how can the effects of a service execution be semantically undone; this is needed for compensation purposes in case a failure in a process execution exists) and on whether a failed service can be re-invoked (retriability).

In addition to transactional guarantees and reliability, OSIRIS focuses on scalability of process execution. The decentralized peer-to-peer approach for process execution in OSIRIS, which is realized by sophisticated replication mechanisms for control flow dependencies, avoids any single point of failure during process execution and provides a high degree of scalability. Peer-to-peer process execution also incorporates sophisticated load balancing in order to distribute process load among available, suitable peers.

Finally, OSIRIS is equipped with the O'GRAPE (OSIRIS GRAphical Process Editor) [Weber et al., 2003] user interface for process definition. It allows for easy creation of process descriptions without programming skills. In addition, O'GRAPE supports the integration of existing application services by leveraging existing Web service standards like SOAP and WSDL.

The service in OSIRIS can either be tightly or loosely coupled – and both kinds of services can be used inside a process.

- Implementations as *Tightly Coupled Services* follow a proprietary OSIRIS service specification and offer additional interfaces to allow controlling the life-cycle of service instances and to retrieve information about the current state of the service instance (e.g., the load situation, reliability and correctness of service invocations) which is needed for load balancing and failure handling of processes.

- *Loosely Coupled Services* are the fastest and easiest way to integrate application-specific services into OSIRIS-based processes. Loosely coupled services are built upon the existing Web service standards, SOAP and WSDL. Therefore, they are easy and fast to implement (or even already in place). However, due to the loose coupling, these services cannot benefit from load balancing and advanced failure handling. In addition, a proxy component, called SOAP component, is needed which is in charge of calling the loosely coupled SOAP service.

Figure 14.2 highlights the OSIRIS middleware with some core services (*Load Managament*, *Service Registry*, and *Process Types*) and additional services to implement digital library functionality.

ISIS: A Digital Library Build On Top of OSIRIS

ISIS is implemented as a set of dedicated of Digital Library services (like feature extraction, index management, index access, relevance feedback, etc.). In addition, it encom-

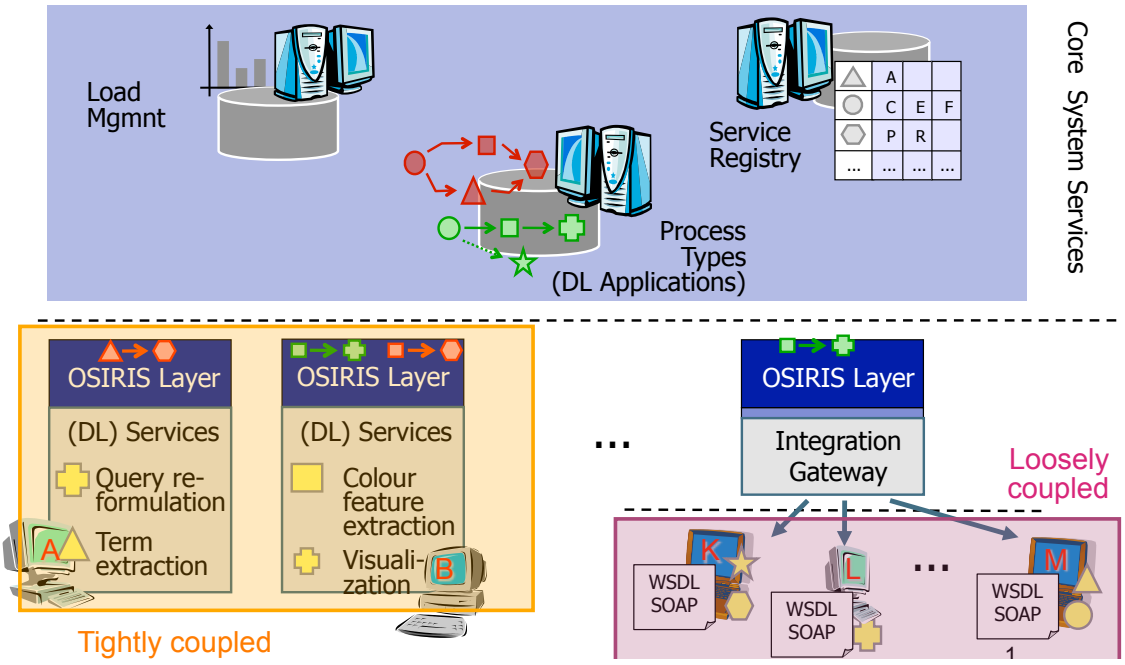
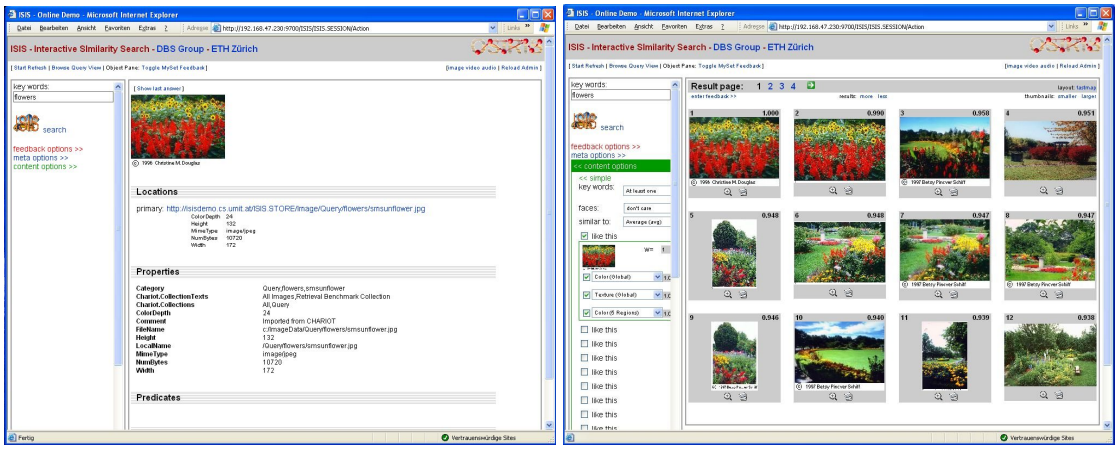


Figure 14.2: The OSIRIS Middleware at a Glance (Illustration from [Agosti et al., 2007])



(a) Query (b) Result

Figure 14.3: Combined Text and Images Similarity Search

passes a set of specialized processes in which some of these services are defined for the definition of complex DL applications.

The screenshots in Figure 14.3 show a combined search for flowers. Starting point is the query frontend, depicted in Figure 14.3(a), where keyword and reference image can be specified. Figure 14.3(b) then shows the query results for this combined query.

One of the main considerations in designing ISIS was to ensure high scalability and flexibility. Therefore, instead of implementing one monolithic application, ISIS consists

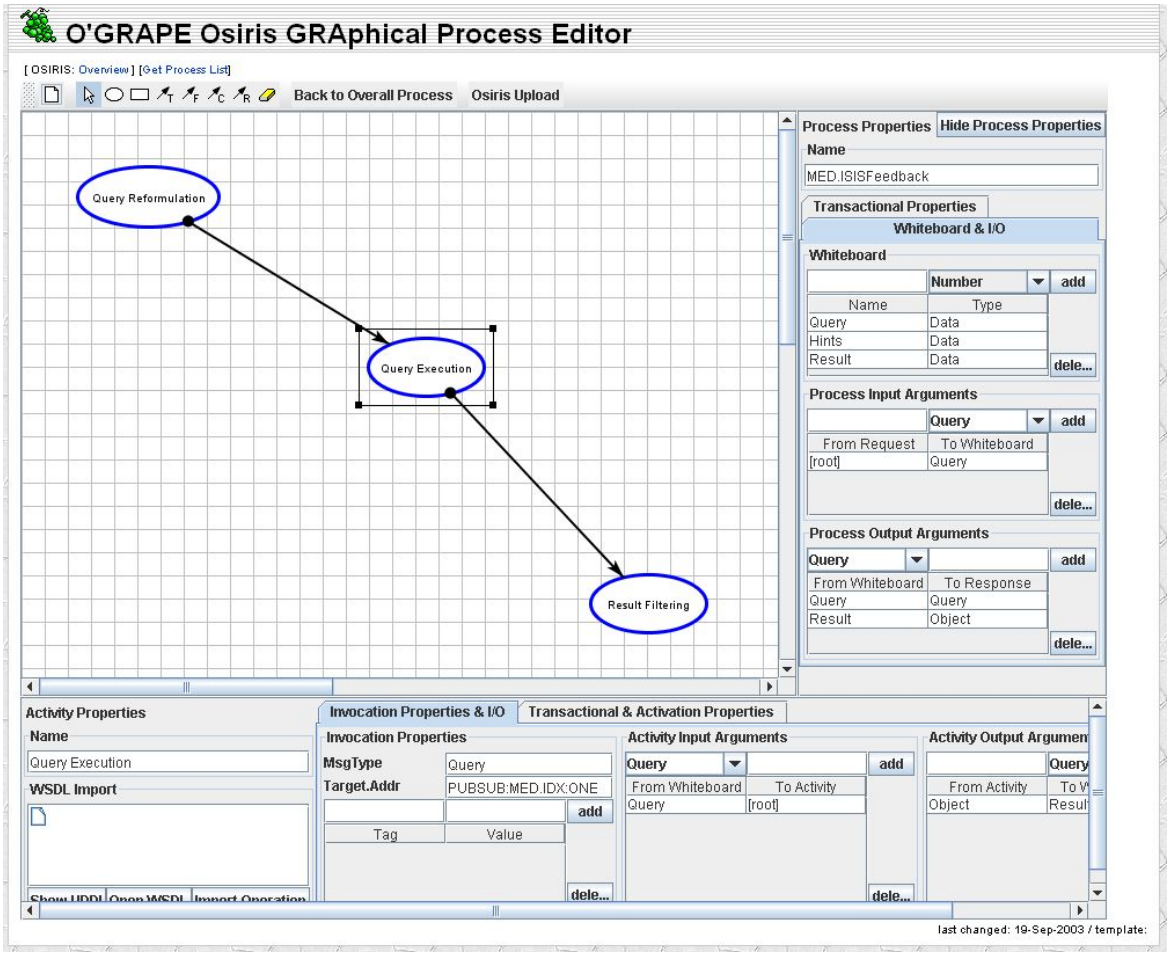


Figure 14.4: Design View of an ISIS Search Process (Encompassing Relevance Feedback) in O'GRAPE

of a set of specialized application services for similarity search which are combined by the OSIRIS middleware. The ISIS services can be easily distributed among several nodes in a network [Weber et al., 1999]. The query presented in Figure 14.3 is therefore implemented as a process. It is important to note that the process specification just contains the details of all application services it encompasses (WSDL description) and the orders within the process. The actual service providers where the service is invoked are determined at run-time. Therefore, information on the location of these providers is not part of the process description. Hence, each step of the process can be executed by any node providing the required service. After issuing the query a first time, a user can refine and re-issue her query. The query process (including user feedback) consists of the steps *Query Reformulation* (based on relevance feedback the user has issued), *Query Execution* (index access), and *Result Filtering* (which may again take user feedback into account). In Figure 14.4, this process is shown in the design view of the O'GRAPE tool.

Any content-based retrieval system is commonly exposed to heavy load under two distinct circumstances:

1. Content-based queries, e.g., for similar images, are based on comparison of features of the object like color histograms or texture. Because these features form high-dimensional retrieval spaces, determining the similarity for ranking the results is computationally expensive. One approach to reduce the system load would use efficient data structures as indexes, e.g., as described in [Weber et al., 1998]. Another approach would replicate data on several nodes to serve more requests in parallel, employ load-balancing, or try to handle parts of the request on several nodes [Böhm et al., 2001b]. ISIS follows both approaches.

2. While replication helps to cope with query load, it increases complexity of modifying a collection by inserting, deleting, or updating objects since the updates of all indexes have to be coordinated to ensure consistency. In ISIS, this is done by appropriate system processes, i.e., processes that have been designed by system administrator and which run automatically to guarantee consistency over several replicas of the index. The extraction of features itself can be a time-consuming task, therefore monitoring constantly changing collections and providing access can be challenging as well. If the insertion of multimedia objects can be divided in several sub-tasks and those can be executed on different nodes while using an infrastructure ensuring correctness of the distributed execution, this can improve the performance significantly [Weber and Schek, 1999].

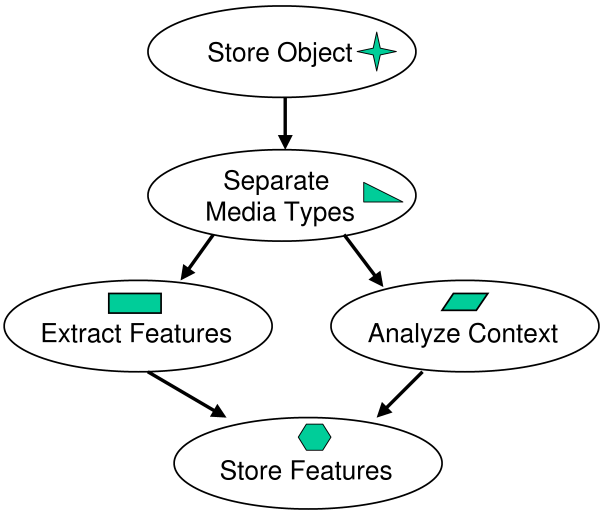


Figure 14.5: Sample Process *Insert Multimedia Object* (Illustration from [Agosti et al., 2007])

Figure 14.5 shows a simplified version of the process used in ISIS to insert new objects which follows this idea of distributing effort among different nodes in the system. The first service (each service is illustrated by a different shape) stores the multimedia object, i.e., the location and available meta information. Depending on the media type further information is extracted. In case of a web document, the object will not only contain an image, but also some text surrounding this image

on the page. Later on, this text is used to determine textual descriptions related to the image. Independent of the image context, the feature extraction service uses raw pixel information of the image. Finally, the store features service hands all derived object information over to a metadata service, which makes it available for indexing and search in a suitable way.

Content Management Services in ISIS

Every object contained in the digital library is assigned a unique identifer, the OID, within the collection. The *Meta Database service* maintains the association of OID with objects. Each object is associated with a object type (OType), which can be image, audio, video, video sequence, or text. Every object can be associated with several locations, e.g., the original URL from which the object has been downloaded and the filename of the local copy, or the address of a thumbnail image. It is also possible to store arbitrary additional properties like size in bytes of the file or whether the object is copyrighted. The Meta Database service also maintains the structure of a collection, that is, the objects can be assigned to a taxonomy of categories.

Storage service is able to deliver the file content of the object and monitor changes in a directory of the file system. In case a change is detected, a process is started in OSIRIS. For instance, if a new file is inserted, *Storage* triggers the execution by sending a message to the OSIRIS process system service. *Storage* heavily uses the underlying OSIRIS middleware also for other tasks, e.g., to transfer objects from one node to another. It can be configured using the configuration agent, e.g., which process should be started when an object is inserted, which process should be started if an object is deleted, which directories should be monitored. Therefore the only operation provided directly to the user is to retrieve a requested file from disk. This is, again, performed using a system service of OSIRIS, the *Web component*.

The *Web Crawler service* periodically monitors websites for modifications. For faster access, a local copy is stored and updated in case of modifications. If a new link is found and this link is qualified for inclusion, the crawler follows this link and inserts the new object. Similar to the *Storage service*, processes are triggered according to modification events. The only difference is that here it is also necessary to specify at what time and in which intervals the crawler should visit the websites, inclusion and exclusion patterns need to be defined, and so on. Hence, more administration options are provided to the user.

Query Formulation and Execution Services in ISIS

There are many different features supported in ISIS, which are offered by a couple of different services. To ease the handling, feature extractors that are implemented as tightly coupled services share a common interface for extraction and for registering to the ISIS system. In order to support several types of features, an administration tool exists in ISIS that automatically generates a sub-process within the activity "Extract Features" for the process in Figure 14.5, where the administrator simply has to select which features should be extracted and indexed. Several services are allowed to provide a feature with the same name. At registration time, it is also possible to specify the storage

format, dependencies on other features, and a preferred distance measure for similarity evaluation. Maintaining the registration information of the features is handled by the *Meta Database service*.

The *Meta Database* also provides information about objects, although this is not necessarily extracted by this component. It provides the image height and width, file size in bytes. It can also handle arbitrary additional properties. These values can be access efficiently through the support of an attribute index of the *Indexing component*.

The general *Feature Extractor component* provides the following features: i.) Color histogram in RGB color space, ii.) Color moments in HCL and Lab color space, and iii.) Gabor texture moments on luminance (cf. Chapter 5.2). All of these can be applied on the image as a whole, 3×3 overlapping rectangles, or "5 fuzzy regions", which basically is one region starting in each corner and a ellipse in the image center where a membership function defines for each pixel of the image its influence on the feature of the region (cf. Chapter 5.1.3). The image can be extracted from a movie, e.g., a key frame. All these features are stored and processed as high-dimensional feature vectors.

The *Face Detector* offers several algorithms to analyze images to identify regions, that contain a face. It returns the number of faces found and a bounding box for each.

The *Audio Feature Extractor* uses the MARSYAS [Tzanetakis and Cook, 1999] library to extract the features "beat" and "pitch", which are again, feature vectors. The *Hypertext Feature Extractor* analyzes HTML documents for embedded links. It also identifies images and assigns the surrounding text to it. The result is not directly used for retrieval. The *Term Frequency Extractor* is applied to plain text, in particular to the output of the *Hypertext Feature Extractor*. It returns a list of all found terms and the number of their occurrences. This feature is stored as a set index and used for boolean and vector space retrieval.

The *Indexing service* is used to answer queries efficiently. It is therefore invoked as the middle step between the relevance feedback related steps within the query process in Figure 14.4. Several types of indexes are supported. The first two, attribute and set indexes, use relational databases. The difference between the two is that attribute indexes are build on one particular value, e.g., the size of an object in number of bytes. This can be handled easily by the functionality that common RDBMS provide. Set indexes, in contrast, can contain multiple values, e.g., used to provide text retrieval in vector space model, for which term frequencies and document frequencies are needed to measure relevance. This can also be handled with the support of an RDBMS as described in [Grabs et al., 2001]. Query processing of these two indexes is optimized by the used database. For high-dimensional feature spaces, e.g., if color histograms of images are used, additional indexes are stored outside the RDBMS [Weber et al., 1998, Mlivoncic and Weber, 2003]. The *Indexing service* optimizes and executes queries on these index types. When complex queries are used, it is also necessary to aggregate the results delivered by the individual indexes. This task is also performed by this component, since this allows for further performance optimizations [Böhm et al., 2001a].

User Interaction in ISIS

ISIS provides a web interface to the user to issue searches. The design is stored in HTML template files extended by a tag library using XSLT. The OSIRIS *Web component* automatically applies those templates to visualize the results or error messages. This makes the customization of a digital library an easy task.

The *Session Management service* provides the basic handling of the user interactions. Each query issued by the user is executed as a new process as displayed in Figure 14.4. The result of the process is handed over to the *Web component*, which will layout the result based on the templates.

The *Relevance Feedback component* evaluates the feedback that a user can issue for previously executed queries. As identified in the search process, this may take place in two execution steps:

1. Before the query is executed, it can be reformulated, e.g., by adding more reference objects to the query or modifying their weights.

2. After the query has been executed, it can filter out some unwanted objects or rearrange the position within the result.

Major Extensions of ISIS in DelosDLMS

As stated already, DelosDLMS has been implemented as an extension of ISIS on top of OSIRIS [Agosti et al., 2007, Binding et al., 2007, Ioannidis et al., 2008, Schek and Schuldt, 2008]. These extensions have affected different aspects of the system:

- New media types like 3D shapes required completely new features to allow similarity search and therefore also new feature extraction services.

- Other new audio feature extraction services provide improved replacements for existing audio features. For video content, dedicated features and services have been added to not only search for keyframes using CBIR, but also be able to search for particular events or a certain action inside the video.

- An annotation service allows the user to add information to existing content, which is also available in the search process.

- New instances of digital libraries with individual collections have been supported through the deployment of dedicated service instances. This included the collection of 3D shapes, but also medical images, a collection of images of archeological artifacts, a specialized collection of historical paintings and sculptures in the dedicated “art gallery”, soccer videos,and cover illustrations of a movie database.

- New user interfaces have been allowed to access the existing service instances.

- New speech interface to retrieve art paintings or particular scenes of a video, both supported by domain-dependent natural language processing and thesauri.

In particular the user interfaces have already been used in this thesis in Chapter 6.1 and Chapter 6.2: Figure 6.7, Figure 6.8, Figure 6.9, Figure 6.10 show different result visualizations that were already present in the web interface of ISIS. Figure 6.5(c) shows the user interface for DelosDLMS using interactive paper, Figure 6.11 shows DelosDLMS search results in a self-organizing map (SOM), Figure 6.14 the MedioVis zoomable user interface for DelosDLMS, and Figure 6.15 accessing DelosDLMS through DARE.

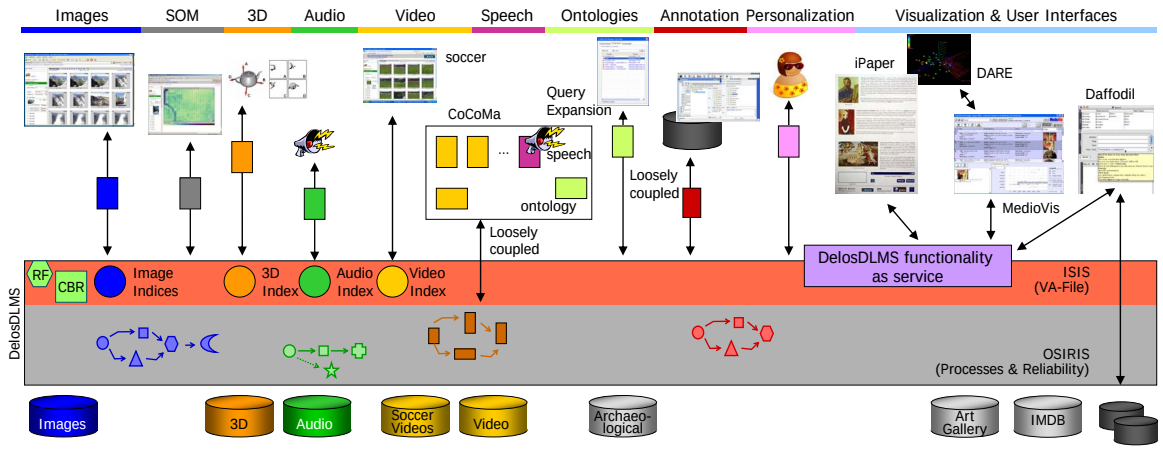


Figure 14.6: Overview of Extensions to ISIS in DelosDLMS (Illustration from [Schek and Schuldt, 2008])

Figure 14.6 presents an overview of the extensions that have been incorporated into DelosDLMS [Agosti et al., 2007, Binding et al., 2007, Ioannidis et al., 2008].

14.1.2 gCube

gCube [Candela et al., 2008b][1] is the system that has been developed in the context of the EU-FP6-funded DILIGENT project and its successors D4Science and D4Science-II and many institutions from European countries have been involved in the development of the system over the years. The emphasize in these projects was the exploitation of Grid infrastructures, including aspects of distributed information retrieval (DIR) over multiple nodes and digital libraries [Simeoni et al., 2006, Simeoni et al., 2007b, Simeoni et al., 2007a, Simeoni et al., 2009], re-use of services in multiple communities organized as virtual research environments (VRE) [Candela et al., 2009b, Candela et al., 2010, Assante et al., 2011], and dynamic deployment of services on top of Grid-environments that have been prepared by a set of core services (collective layer) [Assante et al., 2008].

Figure 14.7 shows an example how several user communities can share resources that are hosted by different providers through virtual organizations (VO). Grid-infrastructures provide security measures to restrict access to resources only to members of a VO.

[1] `http://www.gcube-system.org/`, released under the EUPL open-source license.

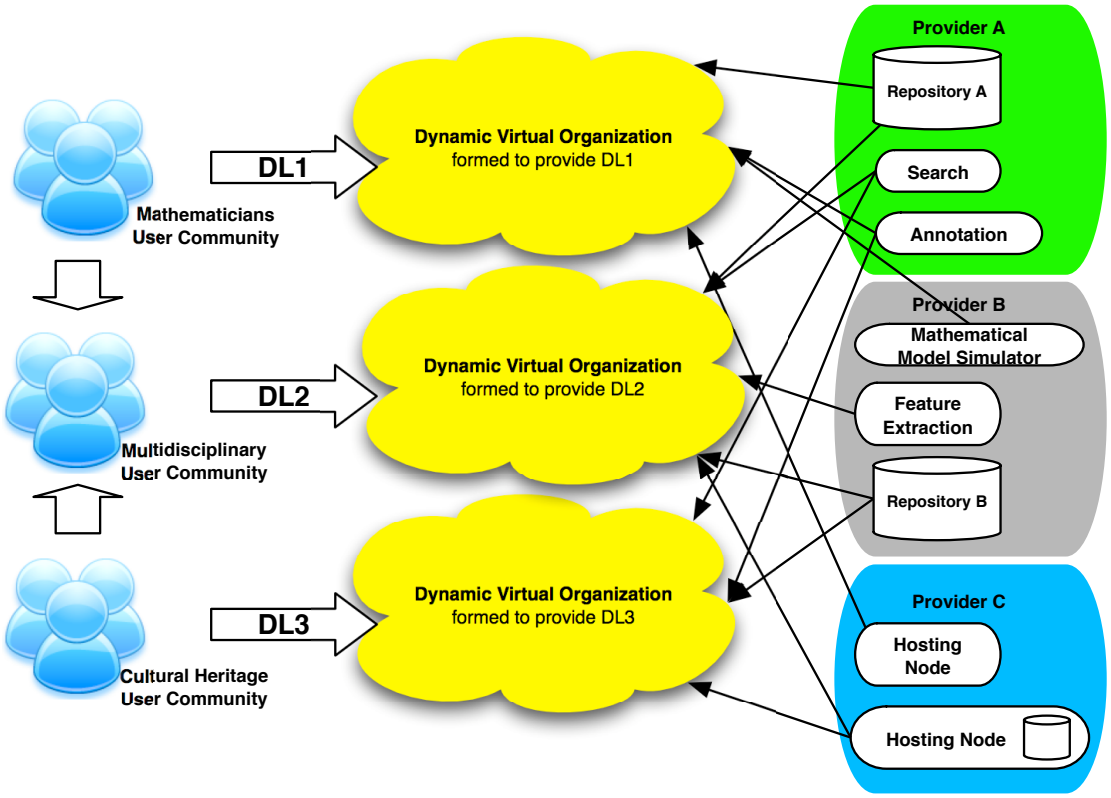


Figure 14.7: Resources shared between several user communities organized in virtual organizations (Illustration from [Candela et al., 2007])

gCube follows the paradigm of a Service-Oriented Architecture (SOA), therefore resources are made available through services. Figure 14.8 provides an overview on the logical architecture of gCube.

Content Management in gCube

The implementation of the content management functionality in a layered implementation with content management layer that provides high level operations for managing documents and collections, an intermediate storage layer that is based on the DILIGENT Storage Model, and a base layer that provides the backend implementations to configure and use backends for Grid- and local filesystems and relational databases. It has been explained in detail in Chapter 9.1.

As gCube has a strong focus on reusing resources and interoperability, importing existing content is of major importance. One particular protocol for this purpose supported in gCube is OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) [Simeoni et al., 2008, Van de Sompel et al., 2004]. Another aspect for this purpose is the transformation of metadata between different formats. For this, and to support queries for exact matches of metadata directly in content management, a dedicated Metadata Management service is added that replicates the metadata that is stored in

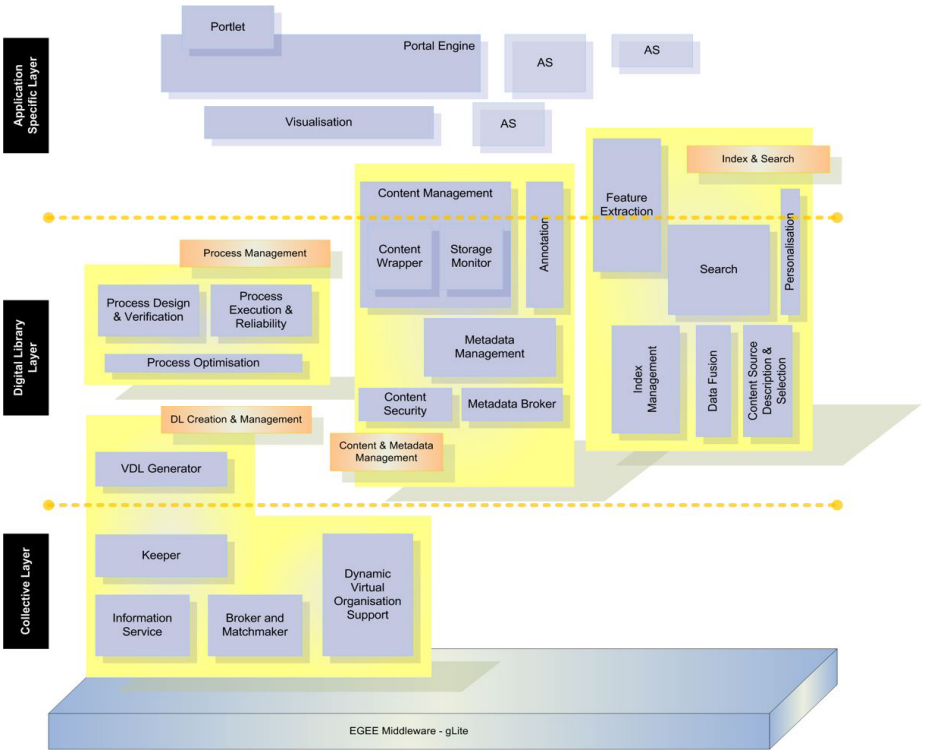


Figure 14.8: Logical Architecture of gCube (Illustration from [Candela et al., 2007])

storage management also into an XML database – in this case eXist [2] which allows rich queries and transformations.

Annotations are treated as particular kinds of metadata and managed by a dedicated service, that acts on top of the Metadata Management service. The Content Security service allows to embed digital watermarks into content.

Query Formulation and Execution in gCube

Feature extraction in gCube technically uses a wrapper to reuse functionality of ISIS, which has already been described in Section 14.1.1. For performing similarity search, it uses its own implementation of the VA-file [Weber et al., 1998].

For gCube, distributed information retrieval is of major interest and performing distributed queries in a dynamic system is challenging as it requires adaptation to resources that are currently available [Simeoni et al., 2007a, Simeoni et al., 2009]. The execution of searches is embedded in workflows, which are not predefined, but generated on the fly (ad-hoc processes) and optimized for the currently available Grid resources by a planning service [Simeoni et al., 2007a, pp. 165f]. The workflows are expressed in BPEL (Business Process Execution Language), which are executed by a process engine that is based on a Java-implementation of OSIRIS [Candela et al., 2007, pp. 69–74].

[2] http://exist.sourceforge.net/

Another important aspect in such a distributed retrieval is the aggregation of results and moving of potentially large datasets, as the common one-to-one request–response interaction pattern between SOAP-based web-services would create heavy load on the network and all participating nodes and latencies until all participating services have delivered their responses. For this reason, intermediate results are communicated using the ResultSet framework [Simeoni et al., 2007a, pp. 166f], which allows asynchronous invocations of services which deliver the results to a dedicated sink. This sink is a ResultSet, that can be the source for subsequent service invocations, thus allowing pipelined processing of the results. At the same time, the ResultSet provides a mean of indirection, therefore reducing the point-to-point communication between nodes as invocations can be performed in a *call-by-reference* manner instead of the *call-by-value* manner that is predominant for SOAP calls.

User Interaction in gCube

The users access gCube as a portal through a web browser. In order to allow flexible adaptation to the particular needs of the users inside a virtual research environment (VRE), the content of the portal itself is not predefined; it is generated by composing individual portlets to support individual functionality. A digital library administrator or dedicated UI designer can define a profile, that defines which and how portlets will be presented inside a VRE [Assante et al., 2011].

As the portlets themselves provide fairly generic functionality that is shared between different communities together with the services they interact with, they can be implemented and reused just like services. In gCube, portlets are implemented using the JSR-168 Portlet Specification [Java Community Process, 2003] and GridSphere[3] as a framework to host the portlets. Also the Google Web Toolkit (GWT)[4] is used for some components of the user interface and a Java applet allows the definition and monitoring of the execution of workflows (a.k.a. compound services).

14.2 Specialized Systems

Three specialized systems have been used in Part III of this thesis. Instead of repeating many details, we want to highlight some aspects of the systems. And one important observation is: All three systems are based on the same implementation of functional building blocks – even though individual each system targets very different tasks. This shared implementation is named IsisJ. It is a complete rewrite of the core functionality of a content-based image retrieval system in Java, but heavily inspired by ISIS and therefore keeps the reference in its name. IsisJ by itself is not a complete system, but rather a set of Java packages that provide elementary implementations of functional building blocks.

This allows for fine-grained reusability and extension of building blocks to serve the needs of the specialized systems: The clear separation into functional building blocks

[3] http://www.gridsphere.org/gridsphere/gridsphere

[4] http://code.google.com/webtoolkit/

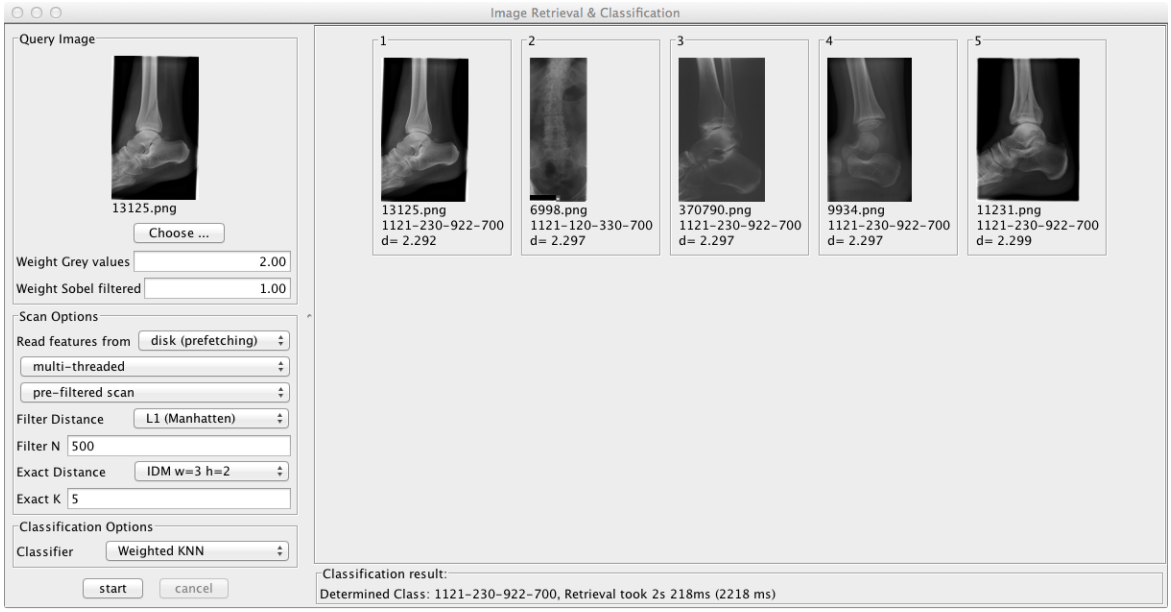


Figure 14.9: Experimentation GUI for Medical Image Classification

improves this reusability by providing logical bounds for any functionality that remains task specific. Therefore any development that is specific for a particular tasks will not interfere with unrelated aspects. Furthermore, as we have seen in Chapter 10.5, all systems based on these building blocks can benefit from generic approaches for optimizing the performance.

14.2.1 Medical Image Classification

As described in Chapter 10.2, the purpose of this small research prototype is to automatically classify medical images. This is performed by using ground-truth images with known class labels and estimating from them the most likely class label for the new image. In normal operation, no user would ever interact with the system directly: Once the reference image collection has been set and parameters adjusted, the only further input will be new images and the system will return the class label. The dataset will also not grow by itself: All new images will have to be reviewed by domain experts to assure that the reference dataset is accurate. As this is a proof-of-concept system, it is only used with a benchmark collection. Therefore, content management is restricted to support only this collection as described in Chapter 9.2.1. The system uses ImageJ[5], mainly for reading common medical image formats in Java.

Query formulation and execution was the key issue in designing the application: The exclusive focus was classification, implemented by an efficient *k*-nearest neighbor similarity search using Early Termination Strategy and multi-threading as described and evaluated in Chapter 10.5.5. Matching tolerance is configurable, which was in particular needed to fine-tune the system for the successful participation in

[5] `http://rsbweb.nih.gov/ij/`

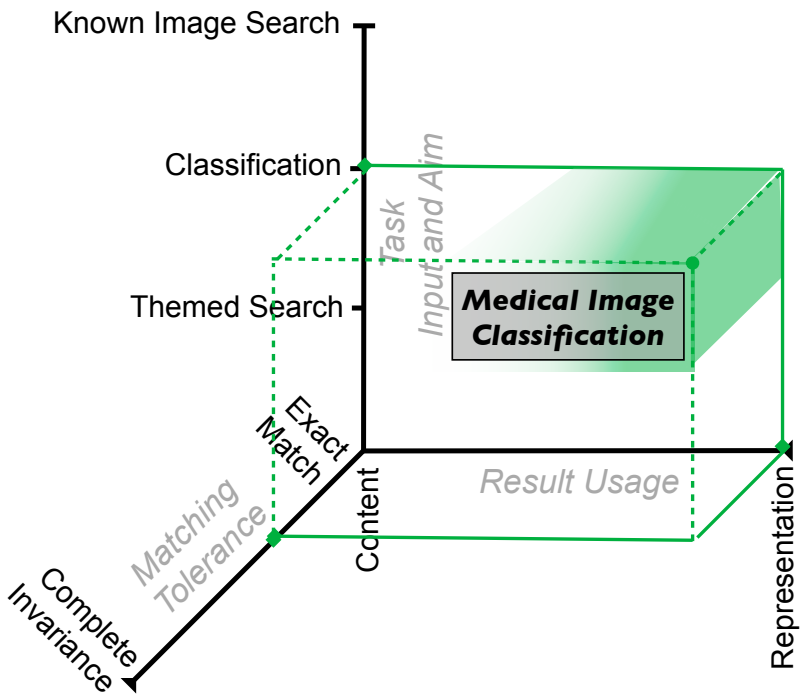


Figure 14.10: Aspects in ITM covered by the prototype system for medical image classification.

image retrieval benchmark of the Medical Automatic Annotation Task at ImageCLEF 2007 [Springmann and Schuldt, 2007, Müller et al., 2008].

In theory, no user interaction would be needed. However, for evaluation of the classification accuracy, some small programs / scripts do exist for batch runs as well as a small GUI application for investigating the behavior for individual images. The user interface is presented in Figure 14.9: As visible in the screenshot, on the top-left, the user can select an image – in particular those, for which the log files of a batch run on the training data did result in a false classification. Below the query image, the user can adjust the parameters for the *k*NN search and the subsequent classification. On the right, the determined nearest neighbors are shown and on the bottom, the determined class label and the time the query execution and classification took.

The intended use of the application is illustrated in Figure 14.10:

- The system is targeted at a single Task Input and Aim – Propagating class labels of existing images, which is an instance of image classification.
- The Matching Tolerance is adjustable and clearly not restricted to exact matches. However, only small deviations are allowed as the position of the content of the image –the patient as a whole or some part of the patient's body– is very limited for medical images of the same class: The class information as described in Chapter 10.2.1 includes a directional code, that keeps track of the viewing direction. Furthermore, the anatomical code denotes which anatomical region of the body is examined and together with the used image modality encoded in the technical

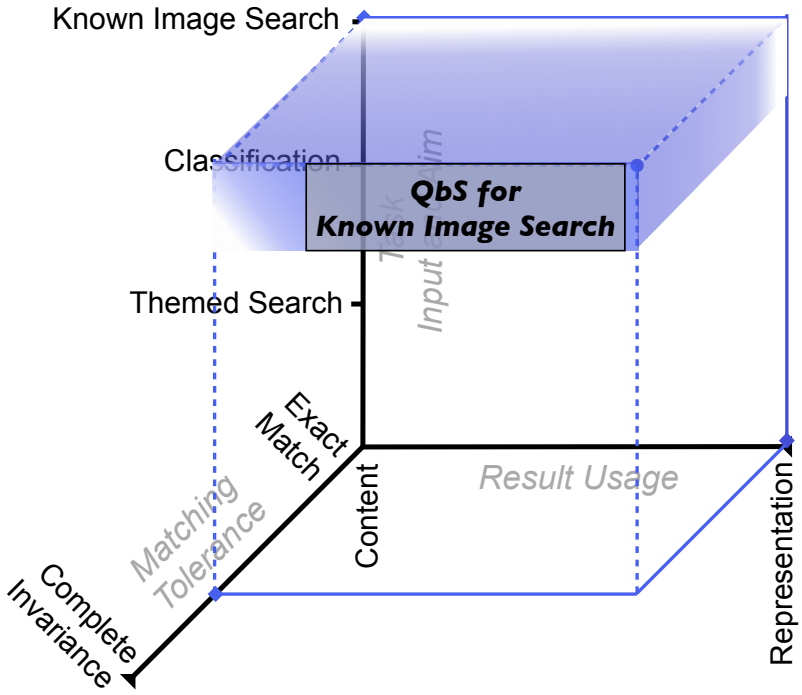


Figure 14.11: Aspects in ITM covered by the QbS prototype system.

code and the biological code, this leaves very little possibility for changes except cased by the subject and his or her medical condition – which is intentional, as such a very precise description of the image settings was the aim of the IRMA code [Lehmann et al., 2002]. Hence, the matching tolerance shall only cope with the intraclass variability of images.

- The result usage is very limited – only the class label associated to the image is relevant,which is an extreme case of representation-oriented usage. However, the GUI may also have the impact that the user learns something about the medical image classes and typical members of them.

14.2.2 QbS for Known Image Search

The Query by Sketching prototype QbS has been mentioned in much detail already within this thesis. It was implemented to target known image search, with a strong emphasize on novel user interfaces including Tablet PCs and Digital Pen and Paper to improve user experience [Springmann et al., 2007b, Springmann et al., 2010c, Springmann et al., 2010a, Springmann et al., 2010b], which was also the subject of Chapter 11.2 – 11.3.

Much effort was spent in expanding the possibilities to define the matching tolerance, for which the existing ARP features have been extended by partial scale and translation invariance. We also applied IDM to sketch-based retrieval, which provides already some limited invariance to scale and translation. For both features, we added

the possibility to define the level of detail of the sketch and unknown areas. These have been described and evaluated in depth in Chapter 10.3. Optimizations to achieve better response times have been subject of Chapter 10.5.6.

The aspects of content management have been mainly restricted to be able to perform evaluations as described in Chapter 9.2.2, however, some extensions have been added to also support the bigger MIRFLICKR-1M dataset [Huiskes et al., 2010] that contain one million images organized in 100 subdirectories.

The intended use is illustrated in Figure 14.11:

- The system is targeted at a single Task Input and Aim – Known Image Search.
- Matching Tolerance covers many invariances, which remain optional. However, there is little interest in retrieving exact matches and no support for very strong invariance as the spatial occurrence of edges or color is concerned: the main cues taken from the input sketch are edges and color, hence invariance to translation and shifts of individual objects inside the sketch will always remain limited.
- A broad range of result usage is supported – rich metadata is preserved and made available to the user. Purely content-oriented usage might not be needed: This would be the case if the representation was not an issue and all the user needed to have is to know the image content to perform the task. The general assumption is, that the user knowns the image; only in cases where part of the image has slipped the user's memory, retrieving the image to regain knowledge of the content might be a purely content-oriented usage. An example of such a situation was Scenario 3 in Chapter 1.4.3: The user Steve knew about the existence of the image of the car at the wedding and that the car would have an emblem of the car make, but Steve could not remember the emblem without looking again at the image.

14.2.3 Retrospective Geotagging

RetroGeotag is the application that has been developed to assists the user in geotagging images even if geocoordinates have not been recorded at the time when the image was taken. This scenario has been described in depth in Chapter 10.4. The intended use of the system is illustrated in Figure 14.12:

- The system is targeted at a single Task Input and Aim – Propagating geotags of existing images, which is an instance of image classification.
- Out of the three systems, retrospective geotagging requests for the broadest range of invariance: A high degree of translation of any object inside the image is tolerated, as long as the viewpoint didn't change too much and it is still the same object. Any change of color is accepted, any rotation may be as well – although, preferably, it's only the aspect ratio that changes, but this may get confused with a rotation by 90°.
- The result usage is very limited – only the geotag associated to the image is relevant.

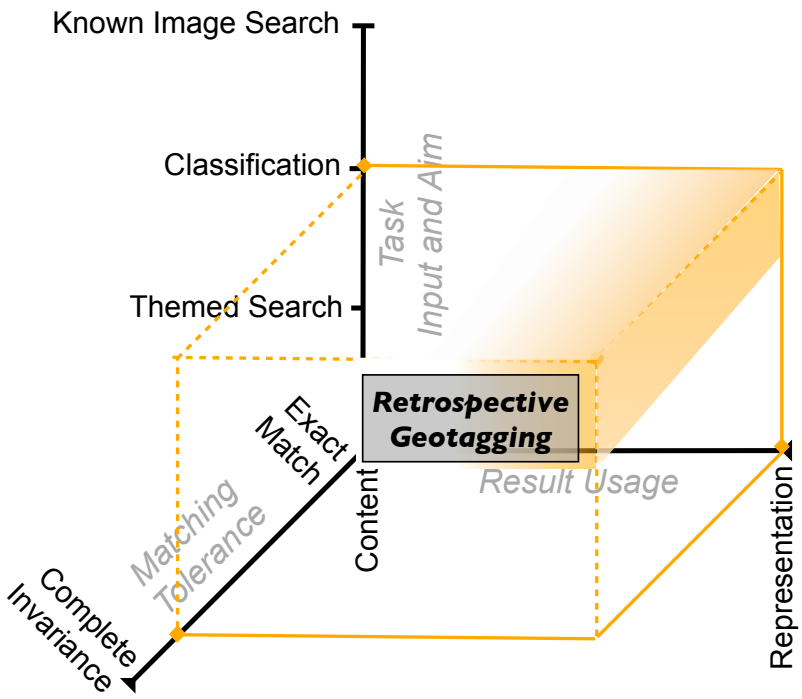


Figure 14.12: Aspects in ITM covered by the prototype system for retrospective geotagging.

The implementation is based on [Morel and Schurter, 2009] and reuses the work on keypoint-based features of [Studer, 2008].The latter itself uses some code of JavaSIFT[6].

Apart from the motivation for the system and how content-based searches are used to estimate the location of images with unknown coordinates, little information on the actual application and its implementation has been presented so far.

Basic User Interaction for Geotagging of Image Series

As mentioned in Chapter 10.4.3, the system is dedicated to tagging individual images in the context of an entire series of images. Therefore the user starts by loading all images that should be tagged in this session from a directory.

Figure 14.13 shows the application to which a series of images was loaded. The last image in the list has already a location associated with it, therefore it shows the coordinates. For any image that has been geotagged, if the user clicks on the save button to the lower right, the geotags will be saved in the image file as Exif metadata, but also the image will get added to the database as a new reference with known coordinates for content-based searches. By this, the user can incrementally increase the dataset to cover more places.

The users can also directly copy geotags of images of the series. This is in particular helpful for events with many pictures taken at the same place. Another function offered

[6]JavaSIFT is an GPL'ed open-source implementation of SIFT, available at `http://fly.mpi-cbg.de/~saalfeld/Projects/javasift.html`

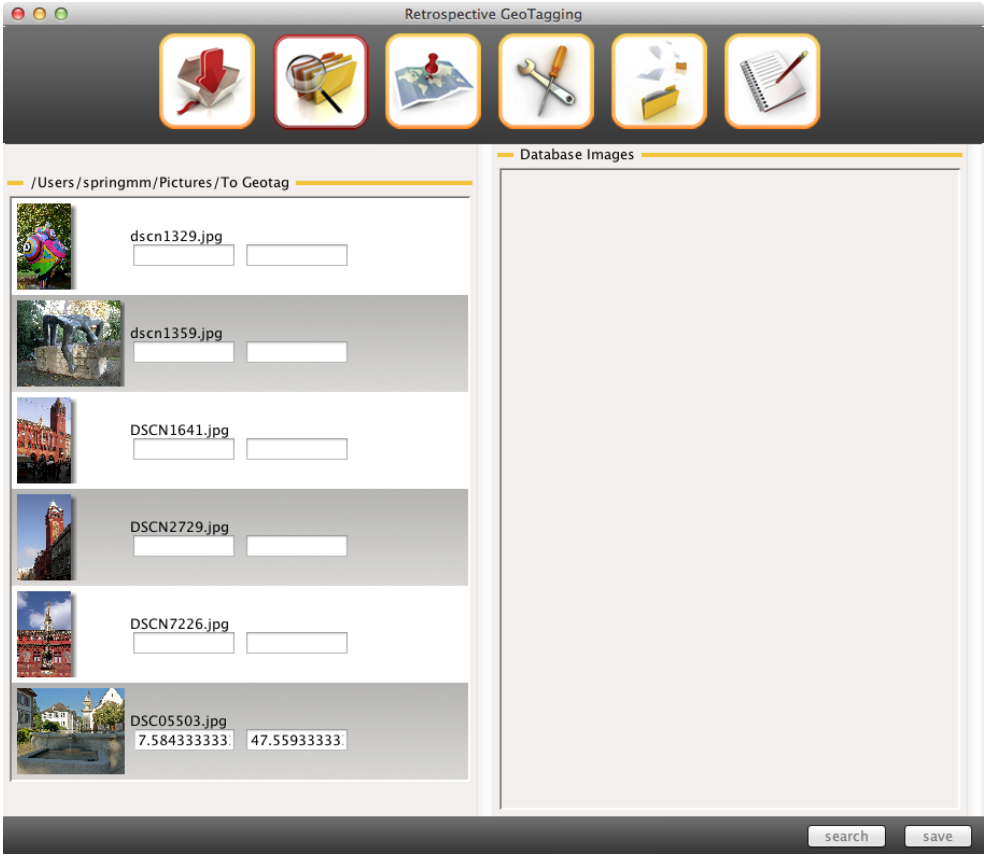


Figure 14.13: List of images for retrospective geotagging

to the user is to interpolate the position based on one geotagged image taken earlier than the current image and another geotagged image taken later: By taking into account the time that has passed between the three images and assuming a direct and constant moving from the different places, the heuristics estimates the coordinates between the two known reference images. These two heuristics are not technically challenging and certainly not reliable enough to be applied automatically. But from a user perspective, they can be very helpful to save time and since they are only applied when the user explicitly invokes them and the results only saved when the user is content with the identified geolocation, they are as accurate as manually assigned geolocations by the same user.

We also included the functionality to geocode the images by performing text-based search for the locations. *GeoNames*[7] provides a geographical database that covers all countries and over eight million placenames. We use the Java client library[8] for performing lookup searches on the GeoNames web service. It returns geocodes for famous places like Big Ben, Eiffel tower, Golden Gate bridge, Red Square in Russia, Mt. Rushmore, Taj Mahal, "Goldenes Dachl" (Golden Roof) in Innsbruck. Also places which are less famous like the Tinguely museum and Marktplatz in Basel are present in this

[7] `http://www.geonames.org/`
[8] `http://www.geonames.org/source-code/`

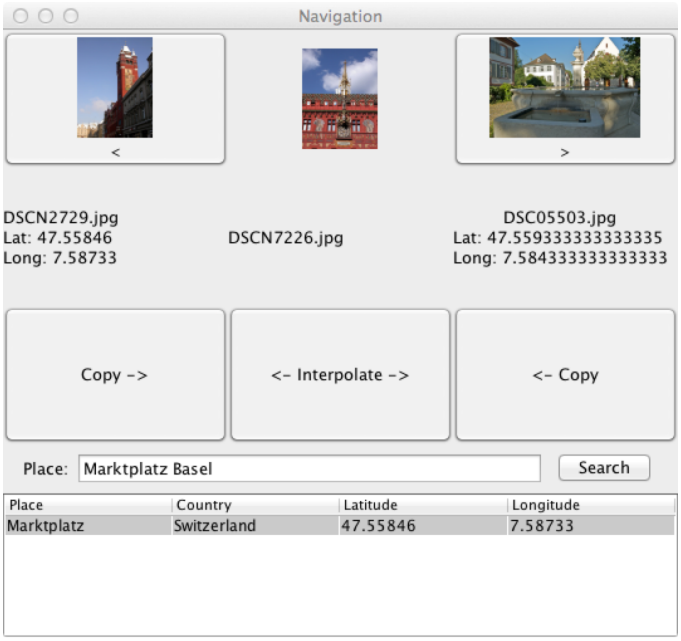


Figure 14.14: Navigation and manipulation in series of images: The image buttons allow to navigate to the previous and next image. The buttons below the images allow to copy the information associated with the images or interpolate the location. The table at the bottom shows the placenames found by the GeoNames lookup service.

database. Thus, all of them can get geotagged very quickly by the user. If the information provided by GeoNames is not sufficient or too imprecise, the GeoNames results can also be used just to approximate the area and manually tag the image with a more accurate location.This is in particular helpful if the object spreads over a wide area, like the Great Wall of China or pictures are taken with significant distance from the geocoded object as commonly the case for pictures of mountains. Figure 14.14 shows the separate controls for navigating in a series of images and the possibility to search by text.

To provide a starting point for content-based searches, there is also a bulk import operation to populate the system with images that can be used as references. This bulk-import functionally can be used to prepare the system for a session: In case the user wants to retrospectively geotag images of an event or even just refine geotags that were assigned on the event as a whole as, for instance, tools like Apple iPhoto and Aperture allow in their "Places" feature, additional geotagged images can get collected by crawling the web for images of this area and inserted to the database with all features extracted. Through this preparation step, the user can geotag her images with a rich set of reference images available.

For displaying the location of images and letting the user place "pins" for new locations or select areas of interest, using a map to visualize these coordinates and their surrounding is essential. The maps are provided by OpenStreetMap[9] via JXMapViewer [10].

[9] `http://www.openstreetmap.org/`
[10] Cf. `http://wiki.openstreetmap.org/wiki/JXMapViewer`

Alternatively, also other map providers can be selected, for instance the NASA Landsat scenes[11].

The main benefit for the user of RetroGeotag is achieved by being able to combine the provided tools and switch quickly between various modes. For instance, the user may look up a place with a textual search, get the result displayed on the map, select an area around this place and then start to perform content-based searches within the selected area.

Content Management in RetroGeotag

The content managed in in RetroGeotag does not require much flexibility as all relationships and properties are well-known at application design time:

1. Images in the reference dataset are without hierarchy; selection of particular images is based on the property "creation time" or "GPS location". The only further property of the image is the image name / ID.

2. The only relationship important for the images are the extracted features of various types.

But compared to the other two specialized prototype applications that can use very simple content management implementations described in Chapter 9.2, RetroGeotag has some slightly more challenging requirements:

- *Growing dataset:* Instead of benchmark collection of fixed size, the intended usage incorporates that the user constantly adds new images to the dataset. The size of this dataset is not expected to be critical, as it will be mainly interesting for the use with personal image collections – therefore far below what one could consider "web scale" of millions of images that common image sharing sites have to deal with.

  Critical is rather the time frame in which changes have to be propagated to any part of the system: The user is expected to geotag series of images and it is commonly most convenient to do this in chronologic order. Therefore the most relevant image with geotags with respect to the image that the user is currently tagging can be the image that the user has tagged just a second ago. It depends only on the content of the images and in case the user took several images at the same or a very close location, the previous image in the series is likely to be the best candidate. If the user is aware of such a situation, the copying tools in Figure 14.14 provide the ideal support. However, if the user knows that there is some small distance between the location of the two shots and the user hasn't been at this spot for the first time or downloaded some geotagged reference images of the area, there is still a chance that the database already contains an image taken at this location. The only way to find out whether such a better image exists, is to issue a search. And therefore the time to store and index a newly geotagged image is ideally just the time that it takes the user to press the save button and move to the next image.

[11] `http://onearth.jpl.nasa.gov/` – or a mirror as the map tile server is no longer in operation; see `http://wiki.openstreetmap.org/wiki/Landsat`.

- *Multiple, high-dimensional Features:* For each image in the database, two simple features (Color and Texture moments) with four different static regions are extracted, as described in Chapter 10.4.3. In addition, many keypoints are detected and each descriptor for a keypoint consumes at least 512 bytes. As shown in Chapter 10.4.3, with an average of 1'482 keypoints per image, up to one megabyte of feature data will be extracted per image. Still, with the storage space that is commonly available these days, this does not appear critical; however, one has to keep in mind that this amount of information also needs to get processed during searches and at query time, it will be not be the features of only a single image that have to be processed. Which directly leads to the next requirement...
- *Selective Access to Data:* The query may include a selected time range or area of interest on a map. If this is the case, only those features will be needed when computing the nearest neighbors that belong to geotagged images that fall inside the defined bounds.

All these requirements are not critical if occurring in isolation, however, when occurring jointly, they become prohibitive for oversimplifying solutions. In particular, sequentially scanning a single file containing all keypoint descriptors for all images –even if the user provided a highly selective date range or area on map– will ruin the performance: Referring again to Chapter 10.5.4, table Table 10.5 shows that even for just 1'500 images, the keypoint descryiptors will result in over 1 GB of data (if stored in an uncompressed format) and Table 10.6 shows that reading 1 GB from a traditional hard disk will take between 7 and 100 seconds; thus, with a growing collection of several thousand images neither reading features from disk nor caching them all into memory will be a reasonable strategy.

Using dedicated index structures to look up only the features for relevant images is advisable – but they need to stay synchronized with the incrementally growing dataset such that whenever the system tells the user that a geotagged has been assigned to an image and the image added to the dataset, the indexes are also up-to-date for searches with the next image of the series. This also prohibits mainly approaches where adding new content requires to perform again a step of discriminative learning and bag-of-visual-words approaches, at least if new content can significantly diverge from the content that was analyzed before.

Relational databases have been built and optimized to deal with many of the mentioned issues: Cost-based optimizers determine when query criteria make index access beneficial over full table scans, cache management optimizes the utilization of the available main memory, and the bounds of a transaction can guarantee that related data remains consistent. The layered implementation presented in Section 9.1 can be used with a relational database backend, however, as summarized in Table 9.2, not all of underlying functionality will then be transparently available for search functionality. To provide access to the underlying functionality, some additional code would be needed.

As the application in itself is fairly simple, it adds less complexity if needed content management functionality is reimplemented directly on top of a suitable relational database. In case of RetroGeotag, we use PostgreSQL[12] as the relational database

[12] `http://www.postgresql.org/`

for storing the image metadata. For restricting search on images within a range of dates, the system itself provides provides adequate index structures; for geospatial data, PostGIS[13] adds the appropriate datatypes and index structures as mentioned in Chapter 10.5.7.

[13] http://postgis.refractions.net/

Table 14.1: Overview on Organizational and Technical Aspects of Systems

System:	ISIS / OSIRIS	DelosDLMS	gCube	Med. Image Class.	QbS	RetroGeotag
Main Development Time	1998 - 2006	2004-2007	2004-2011	2006-2007	2007-2011	2009-2010
#Core Developers	4	4	> 20	1	2	1 (part time)
Institution	ETHZ	UMIT/UNIBAS	> 10	UMIT/UNIBAS	UNIBAS	UNIBAS
Temporary Developers	> 40 Students	> 20 @ Project Partners	> 20 @ Project Partners	1 Student	3 Students	3 Students
External Funding	EU, BBW	EU FP6, SER	EU FP6 / FP7	HITT	SNF	-
Architecture / Technology	SOA using OSIRIS	SOA using OSIRIS and SOAP	SOA using WSRF	Standalone Java Application	Standalone Java Application	Standalone Java Application
Programming Languages	C++, Tcl, JavaScript	+ Java	Java, Ant build scripts	Java	Java	Java
GUI Technology	HTML, CSS, JavaScript, Java Applet (Process Design)	+ dedicated Java Swing GUIs (MedioVis, DARE, Daffodil)	JSR-168 Portlets using GridSphere, Google Web Toolkit (GWT), Java Applet (Process Design)	Java Swing	Java Swing	Java Swing
Underlying Infrastructure Technology	OSIRIS	+ Web Services (SOAP)	gLite Grid Infrastructure	Plain Java (including commonly used building block implementation of IsisJ)		
Major 3rd Party Products Used	MS SQL Server, Apache HTTP Server	+ Apache Tomcat, MySQL, PostgreSQL	Globus Toolkit 4 Java WS Core, MySQL or Apache Derby, GridSphere	ImageJ	Apache Lucene	PostgreSQL, PostGIS
Perceptual Features	Color Moments, Texture Moments, Audio (MARSYAS)	+ Audio (TU Vienna), 3D Shapes	Color Moments, Texture Moments, Video Watermarking	IDM	ARP, IDM, ARCM, EHD	Color Moments, Texture Moments, SIFT
High-Dimensional Index Structures	VA-File, DynReg-File	VA-File, DynReg-File	VA-File	simple files containing all extracted features *(similar to VA-File ignoring approximations and DynReg-File)*		extracted features simply stored in RDBMS
Targeted Task Input and Aim	generic	generic	generic	Classification	Known Image Search	Classification

[1] ETHZ = ETH Zürich, UMIT = University for Health Sciences Tyrol, UNIBAS = University of Basel

[2] EU FPx = x.th framework program of the European Union, HITT = Health Information Technologies Tyrol, SNF = Swiss National Science Foundation, BBW = Schweizer Bundesamt für Bildung und Wissenschaft (now SER), SER = State Secretariat for Education and Research

14.3 Comparison of Systems

In this section, we will finally compare some aspects of the different systems in this chapter. Table 14.1 provides some numbers and other facts about the systems and related projects. DelosDLMS is an extension of ISIS; to represent this visually, the two corresponding columns have been separated with a dotted line in the table. The number of core developers as well as temporary developers shall give a rough impression on how many people have been involved rather than providing accurate numbers that could be translated into a measure like man-years.

This thesis is titled *Building Blocks for Adaptable Image Search in Digital Libraries* and the idea of building blocks includes the vision to be able to build systems reusing building blocks. On an conceptual level, this reusing of building blocks has already been shown. From a software engineering perspective, the reuse may not only face conceptual, but also technical challenges.

All the presented systems have been implemented with the aim of providing reusable software components. However, since the complexity of the systems itself as well as the underlying infrastructure technology differs heavily, this has also some impact on the implementations. We will discuss the different approaches and aspects of software engineering from bottom-up, starting with the source code level.

14.3.1 Modularity on Source Code Level

The source code of all systems uses the features in the used programming languages to separate independent pieces of software: namespaces in C++, packages in Java. This has been done one a fine-grained level, commonly at least one more level than what would correspond to the conceptual building blocks.

To allow reuse also in applications that do not have to be complete digital library systems, fairly independent parts like generic data structures or image processing libraries have been implemented in namespaces / packages of their own.

14.3.2 Modularity on Software Product Level

For the specialized systems based on IsisJ, the separation into different software products has been mainly achieved by forking from a common line of development. The build process for each fork is fairly monolithic, and this was intentional to get new people easily started in working on existing projects and starting projects of their own. This was in particular helpful for students working a limited time on the code for a Bachelor's or Master's thesis project. However, this forking has been assisted by a distributed version control system that allows merging. We use Mercurial[14] which made it easy to propagate changes from the main IsisJ development line into the separate projects and vice-versa allowed also to merge back reusable parts from the specialized systems as soon as their code had matured.

For a limited audience consisting mainly of the staff in our group and students, this model has been quite convenient as getting started has been very simple: Clone the

[14] `http://mercurial.selenic.com/`

IsisJ source code repository, run some of the existing examples that are contained in this repository, start modifying and build a new feature or system. This model, however, is already limited when parts of IsisJ shall be used in other, existing systems: As the build process does not generate independent libraries, one has to get back to the source code level to ship only parts of the software product, e.g., if only packages related from Query Formulation and Execution should be reused.

For the much more complex generic systems, such a model would certainly not have been feasible at all. In ISIS / OSIRIS, the build process generates one or more libraries per building block plus some libraries for the OSIRIS framework. Reusing such libraries is easily possible.

OSIRIS as a system is implemented as a service-oriented architecture and ISIS consists of a set of services and some processes running on to of OSIRIS. Therefore all executable applications are services. That means, that the generated executables have dependencies and need to be deployed to a certain runtime environment – a running OSIRIS infrastructure. To ease this deployment, all required libraries are statically linked into the executables. The reuse of these (tightly integrated) services is only possible within an OSIRIS infrastructure and they cannot be used in isolation.

DelosDLMS has been a strong integration effort, and all of the integrated systems have existed before independently and can still be used independently.

The build process of gCube also generates libraries and services; reuse of libraries in different frameworks is easier than for the executable services that always require a special runtime environment. This runtime environment consists of an extended Globus Toolkit 4 Java WS-Core container[15] named *gCore Container* and the *gCore Framework*. All services and some of the libraries will have dependencies on this environment, therefore may only be used as software products within such an environment.

14.3.3 Reusing Running Service Instances and Data

When the services are deployed to their runtime environment, they allow reuse over the network through the service API. In any service infrastructure that allows the execution of processes / workflows, this allows "programming in the large": Services will get invoked as activities inside the process and new services that enrich the functionality can be embedded inside the process without the need to alter the implementation of the existing service. When service APIs are defined well, it is also not necessary to write "glue code" to make the APIs and invocation parameters compatible – commonly only minor transformations of messages are needed, for instance, to bind one output variable of one service to the corresponding input variable of another service.

The search process in Figure 14.4 is an example where such programming in the large is used: Relevance feedback is added in this context without a need to change the existing Indexing service in ISIS. Instead, the query is reformulated by an additional activity in the search process that takes the query from the process and incorporates the feedback that the user provided in a previous iteration. The modified query is then sent as a new message to the Indexing service. The results returned by the Index service get

[15] http://www.globus.org/toolkit/docs/4.0/common/javawscore/

again processed by an additional activity to filter these results before they ultimately get presented to the user.

In general, such infrastructures allow to build new systems though composition of existing services. DelosDLMS is one example of a system, that has been built by reusing ISIS and combining it with services from different systems. What is particularly appealing is, that by granting access to running services instances, not only the functionality of the services can be used, but also the data that is hosted by the service. In general, this allows to build federated digital libraries without the need to exchange or harvest metadata and actual documents, but use the query functionality and the access to the documents of the individual digital libraries and generate combined results.[16]

In case of ISIS / DelosDLMS, this API is available via:

- OSIRIS Messages: OSIRIS provides its own messaging infrastructure. Tightly coupled services can use this to send messages, but also to execute processes which may span many activities executed at different nodes, as well as other infrastructure features that allow for instance transactional guarantees in execution, broadcasting messages to all instances of a service, querying the load of a service, or deploying starting new service instances.

- SOAP Messages: a dedicated OSIRIS service acts as a gateway to communicate with loosely coupled web services. These services do not need to be running inside the OSIRIS infrastructure, but can be hosted inside any other container as long as network communication is possible.

- REST-style messages: OSIRIS provides direct access via HTTP and OSIRIS messages can be rendered to XML, HTML with some template, or plain text. In cases of legacy systems, in particular client applications, this has been proven a very helpful possibility to access services and the data and functionality that they provide.

All OSIRIS services are stateless w.r.t. API invocation, which eases the calling of OSIRIS services from other systems. Service APIs that are clean and easy to understand also foster the use of services.

gCube is also implemented as a service-oriented architecture, therefore also allows the reuse of services. Most gCube services are stateful using WSRF, therefore continued communication between services can be implemented more efficiently. However, invoking stateful services from external component that are implemented using a different framework is not as simple as invoking stateless services.

gCube provides the capability to execute processes using a process execution engine that is based on a Java-port of OSIRIS and named *CSEngine*. However, instead of a proprietary protocol that has been used in the original C++ version of OSIRIS, CSEngine uses the BPEL standard. This offers great flexibility and this has been even been used for executing queries as ad-hoc processes (cf. Section 14.1.2).

[16]Of course, to achieve good results, in particular good source selection and consistent ranking, quite some additional work has to be performed. Cf. references to distributed information retrieval in Section 14.1.2. However, service-orientation provides a very good starting point for these approaches.

For reuse as a whole and in parts, gCube provides sophisticated graphical tools for dynamic deployment of services and configuration of the digital library together with established security features for authorization and authentication from the Grid-community, or more general: setting up and managing virtual research environment. This is one aspect of reuse, where ISIS and DelosDLMS provide no graphical tools and less comprehensive support for the digital library administrator.

15

Conclusions

15.1 Summary

In this thesis, we have thoroughly analyzed and categorized the potential user needs. We have introduced the Image Task Model (ITM) to characterize image-related search tasks which integrates and refines pre-existing models into one concise model for interaction intentions. This model considers the user's *Task Input and Aim*, *Matching Tolerance*, and intended *Result Usage*.

Based on this analysis, we have identified conceptual building blocks that provide the functionality digital libraries require to support CBIR and similarity search: *Content Management*, *Query Formulation and Execution*, and *User Interaction*. These conceptual building blocks and their interactions have been further investigated and we present a comprehensive survey that reviews state-of-the-art approaches to which extent they can support search tasks on the basis of ITM to identify strong and weak spots. In particular we:

- Introduced the Generic Storage Model that can be used to derive application-depending content models and compare existing approaches.
- Provide an overview of image regions, perceptual features, distance measures, and search primitives and how all these relate to particular classes of image-related search tasks.
- Presented the various possibilities to interact with the user and analyzed, which methods are appropriate depending on the user's task.

We provide a detailed discussion of selected building blocks together with our own implementation that extend and improve state-of-the-art approaches to better support similarity searches for images in digital libraries. This includes both, improved effectiveness and efficiency. In particular we show how user-controllable *Matching Tolerance* can

- improve the classification accuracy for medical images up to 12.56%,

- help users to find known images with sketches well within the top 100 results out of 25'000 images in a collection, and

- allow users to retrospectively assign geolocations to previously untagged images reusing the geotags of images found in a user-generated database.

Furthermore we present the *Early Termination Strategy* for distance computations that can speed up the execution of realistic searches by factors of 2 to 5 without any loss in retrieval quality. Combined with techniques like multi-threading and caching, this allows interactive response times for similarity searches and more expressive queries on commodity hardware. This thesis also shows how content can be stored and managed on Grid- or cloud-infrastructures to add scalability and how to include novel input devices like Tablet PCs and digital pens and interactive paper to improve user interaction for sketch-oriented tasks.

One goal of this thesis is to built entire systems that can adapt to the needs for particular tasks based on reusable building blocks. The three specialized systems for the different application domains *automatic classification of medical images*, *sketch-based search for known images*, *and retrospective geotagging of images* achieve this: They all share reusable building blocks.

In our opinion, complete systems are a necessity to identify how close state-of-the-art techniques for similarity search can match the requirements of digital library users to perform their image-related search tasks. Therefore we hope that our experiences in building complete systems out of functional building blocks will help in two directions:

1. Bringing results from research on individual topics to real end-users: So far, content-based image retrieval and similarity search has not yet found its way into the common tools which are available to users of digital libraries. As we have shown, users of digital libraries with visual content can benefit from similarity search – the major step missing right now is no longer necessarily the development of new tools and new approaches, but bringing them to the end users.

2. Enhancing research from experiences with real users: With research prototypes that focus on very specific aspects of functionality, already some remarkable progress has been achieved. However, only with systems that provide the full set of basic functionality, actual end-users will start to use the systems not only for realistic tasks, but *real* tasks. From this, research in this domain will get very valuable feedback that will certainly lead to new ideas how to improve current approaches.

15.2 Future Work

Of course, this thesis does not mark the end of research in the domain of similarity search in digital libraries. Many questions remain open and will lead to future work.

15.2.1 Users

The second definition of digital libraries by [Borgman, 1997] that has been quoted in Chapter 1.1 includes the users – and so does many of the future work.

More User Studies in General

So far, the systems that we have developed have been evaluated with small groups of users in Part III. The results should get cross-validated with bigger groups of users and several datasets, which are (a) bigger in size and (b) closer to the actual data the individual users work with.

The first aspect is critical to ensure the scalability of the approach, the second will let the user deliver much more sophisticated feedback.

Simplifying the Usage

As it is common in research prototypes, the systems that we developed expose many tuning parameters to the end users as the heaviest users of the systems have been the developers themselves and the possibility to tweak the system as needed is very convenient for them. Real end-users will usually be far less experienced with the system and will have a hard time in setting tuning parameters.

It will therefore be important to identify which control parameters are really needed and can be used correctly by end-users. One particular example for QbS is the slider to select an appropriate range of β-values to control the edge detection. It would ease the use of the system very strongly, if a heuristic could determine values automatically given the user sketch and the content of the collection or preprocess the collection, to get rid of artifacts of the edge detection from background clutter in images that hardly ever are used in real searches for known images. As with many other heuristics that are already used in QbS, the usage of this heuristic could remain optional and be just used in combination with the current approach – but for inexperienced users of the system, this would reduce the early frustration that unfortunately is caused easily when inappropriate β values are selected.

Investigate on Themed Search

Within this thesis, we have evaluated in depth on *Automatic Classification of Medical Images*, *Query by Sketching for Known Image Search*, and *Propagation of Geotags*, therefore two instances related to image classification and one instance of known image search.

Themed search is an equally important task, however, it is usually the task for which it is hardest to define and measure the success of an approach. As the task can be ended at any time as soon as the user is satisfied with the found results, it is essential to define clear tasks that are executed by real users and measure their satisfaction with the tools provided by the system. Evaluation of themed search tasks therefore need much more integration with user studies than other tasks, for which it is frequently sufficient to track the user queries and correct result (correct class label of query image in case of image classification, the ID of the known item in known image search).

15.2.2 Technology

Another line of future work related to the advancement of technology. This will not stop, but already during the writing of this thesis, certain trends have become very visible.

Mobile Devices and Natural User Interfaces

During the last two or three years, smartphones and tablets with touchscreen interfaces and built-in cameras have seen an extreme adoption by end-users. The number of such devices in active use will soon outpace the use of computers – if this hasn't happened already. These devices are certainly not good replacements for all activities that traditionally have been performed on regular desktop or laptop computers; but in certain settings they are already even better suited than more capable computers.

One of these settings is taking and sharing pictures: Smartphones are basically always available, have network connection, and provide cameras that have at least the quality of digital compact cameras which are 5-10 years old. Compared to the abilities when a computer is used in combination with a webcam, scanner, or digital camera that has to be synced to the computer, smartphones are just much more convenient.

Another setting is related to viewing images and navigating through big image collections: Tablets have a smaller screen size than regular computer monitors and do not reach the quality of high-end, color-calibrated screens, but they add significant advantages in user interactions as users may simply point on individual images, pinch to zoom in, and rotate the device to adjust the screen to the proper image orientation. The latter is extremely convenient for images taken in portrait orientation where a tablet with a 10" screen offers the same or more space for the image in proper orientation as a laptop with a 15" screen.

Being able to directly interact with the objects on the screen through (multi-)touch inputs created a new class of user interfaces, the so-called *Natural User Interfaces (NUI)*. In surface computing [Wobbrock et al., 2009], large touch screens built into a table (a.k.a. digital interactive tabletops) are instances that are not mobile, but also support the touch interaction. Other NUIs may display to traditional screens, but receive input without the explicit need of touching a screen, such as Microsoft Kinect that observes the movement of the user in 3D through a camera. NUIs can be particularly appealing for tasks where a high quantity of visual information has to be presented and interacted with. This is already the case for some tasks of media consumption, but in general much more prevalent in media creation.

Both settings, being equipped with cameras and touch-sensitive interfaces, are relevant for many image-related search tasks. Even though not the entire task may be performed ideally on such a device, it may enrich the user experienced when included as optional means for user interaction whenever appropriate. As mobile devices like smartphones and tablets are now usually connected to a network, it is possible to integrate them easily into an existing system setup.

Distributed Index Structures and Solid State Disks

While mobile devices represent a shift of technology on the lower end considering the device size and storage capacity, the high end has also seen some significant shifts. Probably the IT buzzword of the years 2009 to 2011 has been "the cloud". In contrast to "the Grid" that usually was associated with sharing heterogenous resources over very distributed locations and institutions, cloud computing has a much stronger emphasize on almost unlimited scalability to whatever a single client needs.

There has always been the vision to provide content-based image retrieval on true web scale, that is, a single search engine may index all visual content available on the web. This has never been possible as the overhead of providing an infrastructure to handle even a small scale of image content of the web was clearly out of reach for a single research institution. Cloud computing brings this topic back on the agenda as it may allow a single institution to scale up to whatever is required to search over the entire net. [Weber et al., 1999, Weber et al., 2000a, Mlivoncic et al., 2004b], [Falchi et al., 2007, Bolettieri et al., 2009b, Batko et al., 2010], [Torralba et al., 2008], and [Jing and Baluja, 2008] are examples of the regained interest in tackling the problem on web-scale.

Of course, it will still be very expensive for a single research institution to crawl, index, and provide search over any substantial subset of the entire web. Nevertheless, cloud infrastructures remove at least the burden of providing the infrastructure at the institution itself (or one of its research partners). Experiments can start with fairly small subsets and with increased maturity of the system, resources can be added as needed and finally, scalability on much bigger scale can temporarily be evaluated with (synthetic –if needed–) datasets.

Another shift in technology is the availability of non-volatile storage without rotating platters: Solid state drives (SSD) based on flash memory are still significantly more expensive than traditional hard-disk drives (HDD), but they provide much better random access performance and modern SSDs that include a RAID-controller in their controller logic are already sometimes limited by the SATA interface bandwidth for sequential reads. That means that existing disk-based approaches may get faster just by using an SSD, but the full potential is only unveiled when an SSD is not only seen as a replacement for existing storage, but as a means to enhance is as proposed in [Leventhal, 2008]. For similarity search, all the observations on searches in high-dimensional spaces in [Weber et al., 1998] remain valid, therefore sequential scan will remain the baseline to compete with. However, as SSDs have much smaller tradeoff between random and sequential access of entire blocks, the parameters for concrete implementations may have shifted. Furthermore, with increased speed of storage while complexity in features and distance measures has risen, similarity search in 2012 and beyond may be much more CPU-bound than it was in 1998 and ideal index structures must be prepared not only for high internode parallelism –using multiple nodes in a network for search– but also for high intranode parallelism –using multiple threads, single instruction-multiple data (SIMD), and General-Purpose computation on Graphics Processing Units (GPGPU).

Source Code Packaging

Another technical challenge is more an engineering than a research challenge: Easing reuse of the software as such.

The development of software in the context of this thesis after the shift from C++ as the main programming language in ISIS / DelosDLMS towards Java in the context of DILIGENT / gCube and the specialized systems in Chapter 14.2 based on common building blocks has also led to a shift in the development model: All code has been explicitly developed under a open-source license – EUPL in case of gCube and for IsisJ mostly the LGPL, some parts also dual-licensed allowing even less restricted reuse under the terms of a BSD license, Apache License, or GPL with Classpath exception.

The code itself has been structured into logical packages which separate independent aspects. However, the build process is still fairly monolithic. Technically, a much more appealing solution would separate the aspects not only into different packages in terms of Java, but also modularize the repository and build process into software packages that are built and maintained in isolation. Tools like Apache Maven[1] and Hudson[2] / Jenkins[3] have already been used in our group for the development of the OSIRIS[4] and would certainly also be beneficial for IsisJ. So far, it was just the lack of time that prevented us from performing this step of modularization.

Once this engineering task of setting up individual projects, the next step will not only be a technical one: Grant access to a wider range of developers and establish a community to foster the reuse and enhancement of IsisJ. This will hopefully lower the burden to build systems out of existing implementations of functional building blocks for other people interested in similarity search ...and through this also lead to new research in this domain.

More Automatism in Usage and Reuse of Building Blocks

With the availability of many building blocks as software components and services, the work required to build systems for particular application domains will shift from the implementation of the elementary building blocks towards the integration and reuse. Development models and tools can assist in this regard.

The service-oriented architectures used in ISIS/DelosDLMS and gCube already allowed "programming in the large" where much of the application logic is encapsulated in workflows that are defined on top of services. The development of a new application may therefore only consist of deploying all required instances of services and adapting the workflows. In case of ISIS, workflows for feature extraction were even generated by the system and in gCube, ad-hoc workflows are defined for executing searches.

However, the development of new applications still requires a significant amount of in depth knowledge of all services involved in a process and their APIs – even if programing is limited to the design of workflows. New methods should assist the de-

[1] http://maven.apache.org/
[2] http://hudson-ci.org/
[3] http://jenkins-ci.org/
[4] http://on.cs.unibas.ch/

veloper by taking the API definition and probably some templates to reduce the manual work required to integrate new services and build new applications.

15.2.3 Media Types: Audio, Video, 3D, Compound Documents

This thesis, support for similarity search in digital libraries has been analyzed in depth and implemented for the the use with images. There are other media types also held in digital libraries, for which similarity search is of great interest as well.

Examples: Plagiarism detection in text documents. Query by Humming and by recorded samples for music retrieval. Video retrieval for particular scenes.

Switching to different media types is not only a question of technology, but it also affects the way people use the data, what information they can provide as search input, which kind of presentation and user interaction is adequate.

Granularity of Result

The Image Task Model (ITM) introduced in Chapter 2.5 on page 69 can be reused and adapted to these media types as task input and aims remain similar (search for known items, classification, themes), matching tolerance is certainly still a requirement, and the result usage can switch between two poles (content vs. representation). However, the details might differ significantly and novel problems may arise.

For instance, while in image retrieval it is usually of great help to present thumbnails to give a rough overview over the entire content. When a user sees search result image in full resolution (or for very high resolutions: adequately sized to the used screen), the user does usually not need any further assistance – even when the region of interest is just a part of the overall image; the user is able to identify the region quickly in the image and the system may assist with pointers, bounding boxes or other means of highlighting the region. In ITM, the encoding whether the user searches an image as a whole or is only interested in some region can therefore sufficiently expressed in the matching tolerance.

For other content that has a time component, it might be more challenging to express precisely what the user is looking for and present the search result in a way that the user can quickly identify the desired results: Audio and video material can usually not be summarized in a single snapshot. The common approach for video would be to perform automatic scene detection and –at least– generate a thumbnail per scene. But this creates the question, what should be the granularity of what shall get retrieved: Entire movies, a scene within a movie, or probably even individual frames. ITM has to deal with such questions w.r.t. retrieval of images based on parts of the image, e.g., object detection – but there it is limited to spatial segments, not also segments in time.[5]

Similarly, in audio retrieval search must also respect whether users look for entire audio tracks or individual parts of it. In 3D, most datasets that are available so far

[5] In practice, for instance in YouTube `http://www.youtube.com` as a side-effect from the 15 minutes length limit on movie clips to reduce copyright infringement issues, this problem of retrieval of scenes or entire movies is also decreased: Video clips on YouTube correspond mostly to scenes – if entire, longer sequences or entire movies are made available, this will be mostly done as a playlist.

consist of individual objects – therefore the granularity of the result will be the object. However, to judge whether the object is the desired one, the result presentation needs to determine in which position the result has to be shown as from many views of the 3D object, the similarity that the system identified might be not easily detectable by the user. For compound documents, of course, the document as a whole or any of its parts could be the desired result – or even a region or sequence of a single part and any combination.

Works on such media types therefore will have to carefully analyze tasks what granule the user is seeking in order to measure well the quality of results.

Development of Objects inside the Content over Time

For images, spatial arrangement is the main relationship that can be identified between several objects inside a single image. In video, the spatial arrangement is time dependent and movement of objects or people might be an important aspect to the user. For instance, in video recordings of sports events or surveillance cameras, a sequence can be of interest only because of the movement of a person at a certain time.

Furthermore, an object may change its shape and appearance over time with extreme cases might be a time-lapse recording of nature changing its appearance over seasons, a caterpillar turning into a butterfly, or a normally immobile object breaking into pieces – the latter may even change the number of individual objects.

All these aspects can be very relevant for the user. In case they determine the relevance of search results, they have to modeled such that they can get queried.

Sources of Additional, Textual Information for Media Types

In images, textual information either has to be extracted from the image by means of optical character recognition (OCR) or provided as annotations or descriptions. For video and audio, similar methods exist: OCR of still images in movies and speech recognition from audio tracks of movies and audio files in general and any kind of annotation. What is different: In the vast majority of cases for images, high-quality annotations or descriptions do not exist ("A picture is worth a thousand words") – for audio and video content, the transcript of the language does frequently exist, e.g., lyrics in case of (popular) music and closed captions for films and TV shows. For such content, also more textual summaries in form of reviews may exist compared to pictures, where such information is usually only available for highly important pieces of art – for commercial film and music productions, these are much more likely to exist – at least in the form of advertising material. For audio and video recordings of events, information related to particular timeframes may exist, e.g., a schedule of speakers over time or for sports events, in which minute a team scored.

In comparison to images, the availability and reliability of (textual) descriptions at present has to be considered much better than human descriptions of images or generic tools for automatically analyzing image content. The existence of such additional information is usually not coincidental, but caused by the fact that this information is very relevant to the users of the content. Due to this fact, purely content-based methods may not be able to compete with methods that focus just on such information associated with

the content. That does not mean that content-based methods cannot provide additional insights –they can– but any successful approach will need to make sure that it does already exploit all available sources of information.

Building Blocks

The major building blocks described in Part II *Content Management*, *Query Formulation and Execution*, and *User Interaction* will still remain valid although some adaptation of the implementation is likely to be required.

The *Generic Storage Model* can still be used to model these kinds of content as it separates the content model from the storage primitives and therefore may, for instance, a time sequence of parts can be expressed through relationships just as any other relationship. However, the emphasize on particular aspects may change, for instance, individual audio and video files require usually more storage than image files and the transmission of content is frequently more time-critical, in particular when content is streamed and delay in transmission may cause drop-outs in playback.

The individual building blocks for features extraction will also need adaptation to the media type and application domain, in particular speech-to-text conversion is likely to become a key component for audio and video recordings of conversations without existing transcripts. User interaction will require significant changes to present well the search results of different media types to the user. Also some new methods and devices may be required to interact, in particular navigate through the content and issue queries.

New systems built from building blocks to solve particular user-tasks and detailed evaluations including user-studies will therefore still be needed for any new media type. ISIS and DelosDLMS already did support more media types than just images in particular, audio, video, and 3D shapes which showed that many of the building blocks of ISIS were –in fact– reusable. However, the evaluation of the quality of retrieved results has never reached the same level of sophistication. This may have been caused by a much lesser analysis of the tasks that real users want to perform. Adapting the Image Task Model to such tasks on different media types may be a helpful tool.

Bibliography

[Abadi et al., 2006] Abadi, D., Madden, S., and Ferreira, M. (2006). Integrating compression and execution in column-oriented database systems. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*, pages 671–682, New York, NY, USA. ACM.

[Agarwal et al., 2004] Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(11):1475–1490.

[Aggarwal et al., 2001] Aggarwal, C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT 2001)*, pages 420–434, Berlin / Heidelberg. Springer.

[Aggarwal et al., 1999] Aggarwal, C. C., Wolf, J. L., and Yu, P. S. (1999). A new method for similarity indexing of market basket data. *ACM SIGMOD Record*, 28:407–418.

[Agosti et al., 2007] Agosti, M., Berretti, S., Brettlecker, G., Del Bimbo, A., Ferro, N., Fuhr, N., Keim, D. A., Klas, C.-P., Lidy, T., Milano, D., Norrie, M. C., Ranaldi, P., Rauber, A., Schek, H.-J., Schreck, T., Schuldt, H., Signer, B., and Springmann, M. (2007). DelosDLMS - the integrated DELOS Digital Library Management System. In *Digital Libraries: Research and Development, Proceedings of the First International DELOS Conference – Revised Selected Papers*, pages 36–45, Berlin / Heidelberg. Springer.

[Akal et al., 2005] Akal, F., Türker, C., Schek, H.-J., Breitbart, Y., Grabs, T., and Veen, L. (2005). Fine-grained replication and scheduling with freshness and correctness guarantees. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, pages 565–576, New York, NY, USA. ACM.

[Al-Maskari and Sanderson, 2010] Al-Maskari, A. and Sanderson, M. (2010). A review of factors influencing user satisfaction in information retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 61(5):859–868.

[Al-Maskari et al., 2008] Al-Maskari, A., Sanderson, M., Clough, P., and Airio, E. (2008). The good and the bad system: does the test collection predict users' effectiveness? In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 59–66, New York, NY, USA. ACM.

[Alexe et al., 2010] Alexe, B., Deselaers, T., and Ferrari, V. (2010). ClassCut for unsupervised class segmentation. In *Proceedings of the 11th European Conference on Computer Vision (ECCV 2010)*, volume 5, pages 380–393, Berlin / Heidelberg. Springer.

[Amato et al., 2008] Amato, G., Debole, F., Peters, C., and Savino, P. (2008). The MultiMatch Prototype: Multilingual/multimedia search for cultural heritage objects. In *Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2008)*, pages 385–387, Berlin / Heidelberg. Springer.

[Amato et al., 2009] Amato, G., Debole, F., Peters, C., and Savino, P. (2009). MultiMatch: multilingual/multimedia access to cultural heritage. In *Post-proceedings of the Fifth Italian Research Conference on Digital Libraries (IRCDL 2009)*, pages 162–165. DELOS Association.

[Anh et al., 2001] Anh, V. N., de Kretser, O., and Moffat, A. (2001). Vector-space ranking with effective early termination. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '01)*, pages 35–42, New York, NY, USA. ACM.

[Ankerst et al., 1999] Ankerst, M., Kastenmüller, G., Kriegel, H.-P., and Seidl, T. (1999). Nearest neighbor classification in 3d protein databases. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 34–43. AAAI.

[Ardizzoni et al., 1999] Ardizzoni, S., Bartolini, I., and Patella, M. (1999). Windsurf: region-based image retrieval using wavelets. In *Proceedings of the 1st Workshop on Similarity Search (IWOSS'99), held in conjunction with the 10th International Workshop on Database and Expert Systems Applications (DEXA '99)*, pages 167–173.

[Armitage and Enser, 1997] Armitage, L. H. and Enser, P. G. (1997). Analysis of user need in image archives. *Journal of Information Science*, 23(4):287–299.

[Assante et al., 2008] Assante, M., Candela, L., Castelli, D., Frosini, L., Lelii, L., Manghi, P., Manzi, A., Pagano, P., and Simi, M. (2008). An extensible virtual digital libraries generator. In *Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2008)*, pages 122–134, Berlin / Heidelberg. Springer.

[Assante et al., 2011] Assante, M., Pagano, P., Candela, L., De Faveri, F., and Lelii, L. (2011). An approach to virtual research environment user interfaces dynamic construction. In *International Conference on Theory and Practice of Digital Libraries (TPDL 2011)*, pages 101–109, Berlin / Heidelberg. Springer.

[Bally et al., 1999] Bally, P., Angleraud, C., and Somer, Y. (1999). The spot image coherence product and dimap: A new format for a new product. In *Proceedings of the SAR Workshop: CEOS Committee on Earth Observation Satellites; Working Group on Calibration and Validation*, volume 450 of *ESA-SP*, pages 671–678, Paris, France. European Space Agency (ESA).

[Barnes et al., 2009] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28:24:1–24:11.

[Barni et al., 1998] Barni, M., Bartolini, F., Cappellini, V., and Piva, A. (1998). Copyright protection of digital images by embedded unperceivable marks. *Image and Vision Computing*, 16(12-13):897–906.

[Bartolini et al., 2001] Bartolini, I., Ciaccia, P., and Waas, F. (2001). FeedbackBypass: A new approach to interactive similarity query processing. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 201–210, San Francisco, CA, USA. Morgan Kaufmann.

[Batko et al., 2010] Batko, M., Falchi, F., Lucchese, C., Novak, D., Perego, R., Rabitti, F., Sedmidubsky, J., and Zezula, P. (2010). Building a web-scale image similarity search system. *Multimedia Tools Appl.*, 47:599–629.

[Batko et al., 2007] Batko, M., Novak, D., and Zezula, P. (2007). MESSIF: Metric similarity search implementation framework. In *Digital Libraries: Research and Development, Proceedings of the First International DELOS Conference – Revised Selected Papers*, pages 1–10, Berlin / Heidelberg. Springer.

[Batley, 1988] Batley, S. (1988). Visual information retrieval: Browsing strategies in pictorial databases. In *Proceedings of 12th International Online Information Meetings (Online Information 1988)*, volume 1, pages 373–381, Oxford, UK. Learned Information.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision (ECCV 2006)*, pages 404–417.

[Beaver, 2007] Beaver, D. (2007). Facebook photos infrastructure. WWW page. `http://blog.facebook.com/blog.php?post=2406207130`, last accessed: February 2nd, 2010.

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD '90)*, pages 322–331, New York, NY, USA. ACM Press.

[Bei and Gray, 1985] Bei, C.-D. and Gray, R. M. (1985). An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, 33(10):1132–1133.

[Bekaert et al., 2003] Bekaert, J., Hochstenbach, P., and Van de Sompel, H. (2003). Using MPEG-21 DIDL to represent complex digital objects in the Los Alamos National Laboratory Digital Library. *D-Lib Magazine*, 9(11).

[Berchtold and Keim, 1998] Berchtold, S. and Keim, D. A. (1998). Section Coding: A similarity search technique for the car manufacturing industry. In *Proceedings of the International Workshop on Issues and Applications of Database Technology (IADT'98)*, pages 256–263.

[Berchtold et al., 1996] Berchtold, S., Keim, D. A., and Kriegel, H.-P. (1996). The x-tree: An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96)*, pages 28–39, San Francisco, CA, USA. Morgan Kaufmann.

[Berchtold et al., 1997] Berchtold, S., Keim, D. A., and Kriegel, H.-P. (1997). Section Coding: Ein verfahren zur ähnlichkeitssuche in cad-datenbanken. In *GI-Fachtagung: Datenbanksysteme in Büro, Technik und Wissenschaft (BTW'97)*, pages 152–171.

[Berretti et al., 2000] Berretti, S., Del Bimbo, A., and Pala, P. (2000). Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Transactions on Multimedia*, 2(4):225–239.

[Bertini et al., 2011] Bertini, M., Del Bimbo, A., Ioannidis, G., Stan, A., and Bijk, E. (2011). A flexible environment for multimedia management and publishing. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval (ICMR '11)*, pages 76:1–76:2, New York, NY, USA. ACM.

[Bidgood et al., 1997] Bidgood, W. D., Horii, S. C., Prior, F. W., and Van Syckle, D. E. (1997). Understanding and using DICOM, the data interchange standard for biomedical imaging. *Journal of the American Medical Informatics Association (JAMIA)*, 4(3):199–212.

[Binding et al., 2007] Binding, C., Catarci, T., Christodoulakis, S., Crecelius, T., Gioldasis, N., Jetter, H.-C., Kacimi, M., Milano, D., Ranaldi, P., Reiterer, H., Santucci, G., Schek, H.-J., Schuldt, H., Tudhope, D., and Weikum, G. (2007). DelosDLMS: Infrastructure and services for future digital library systems. In *Proceedings of the Second International DELOS Conference*.

[Bischoff et al., 2008] Bischoff, K., Firan, C. S., Nejdl, W., and Paiu, R. (2008). Can all tags be used for search? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008)*, pages 193–202. ACM.

[Bloch, 2003] Bloch, J. (2003). *Effective Java*. Addison-Wesley, Boston, MA, USA.

[Bober, 2001] Bober, M. (2001). MPEG-7 visual shape descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716–719.

[Böhm et al., 2001a] Böhm, K., Mlivoncic, M., Schek, H.-J., and Weber, R. (2001a). Fast evaluation techniques for complex similarity queries. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 211–220, San Francisco, CA, USA. Morgan Kaufmann.

[Böhm et al., 2001b] Böhm, K., Mlivoncic, M., and Weber, R. (2001b). Quality-aware and load-sensitive planning of image similarity queries. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 401–410, Washington, DC, USA. IEEE Computer Society.

[Bolettieri et al., 2009a] Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Piccioli, T., and Rabitti, F. (2009a). CoPhIR: a test collection for content-based image retrieval. *Computing Research Repository (CoRR)*, abs/0905.4627v2.

[Bolettieri et al., 2009b] Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., and Rabitti, F. (2009b). Enabling content-based image retrieval in very large digital libraries. In *Proceedings of the Second Workshop on Very Large Digital Libraries (VLDL 2009)*. DELOS Association.

[Borenstein et al., 1996] Borenstein, J., Everett, H. R., and Feng, L. (1996). *Where am I? Sensors and Methods for Mobile Robot Positioning*. A. K. Peters, Ltd., Wellesley, MA, 1996 edition. original 1995 Edition out of print, new edition available online at `http://www-personal.umich.edu/~johannb/position.htm`.

[Borgman, 1997] Borgman, C. L. (1997). Now that we have digital collections, why do we need libraries? In *Proceedings of the 60th ASIS Annual Meeting*, volume 34, pages 27–33.

[Bosch et al., 2007] Bosch, A., Zisserman, A., and Munoz, X. (2007). Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval (CIVR '07)*, pages 401–408, New York, NY, USA. ACM.

[Boujemaa and Ferecatu, 2002] Boujemaa, N. and Ferecatu, M. (2002). Approximate search vs. precise search by visual content in cultural heritage image databases. In *Proceedings of the 4. International Workshop on Multimedia Information Retrieval (MIR'02)*.

[Brettlecker et al., 2007] Brettlecker, G., Ranaldi, P., Schek, H.-J., Schuldt, H., and Springmann, M. (2007). ISIS & OSIRIS: a process-based digital library application on top of a distributed process support middleware. In *Digital Libraries: Research and Development, Proceedings of the First International DELOS Conference – Revised Selected Papers*, pages 45–55, Berlin / Heidelberg. Springer.

[Browne et al., 2006] Browne, P., Rüger, S. M., Xu, L.-Q., and Heesch, D. (2006). iBase: Navigating digital library collections. In *Proceedings of the 5th International Conference on Image and Video Retrieval (CIVR 2006)*, pages 510–513, Berlin / Heidelberg. Springer.

[Burnett et al., 2003] Burnett, I., Van de Walle, R., Hill, K., Bormans, J., and Pereira, F. (2003). MPEG-21: goals and achievements. *IEEE MultiMedia*, 10:60–70.

[Büttcher et al., 2010] Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press, Cambridge, MA, USA.

[Callahan et al., 1988] Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. (1988). An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '88)*, pages 95–100, New York, NY, USA. ACM.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg. Springer-Verlag.

[Camp, 2009] Camp, H. (2009). Flickr! It's made of people! WWW page. `http://blog.flickr.net/en/2009/10/21/people-in-photos/`, last accessed: February 2nd, 2010.

[Candela et al., 2007] Candela, L., Akal, F., Avancini, H., Castelli, D., Fusco, L., Guidetti, V., Langguth, C., Manzi, A., Pagano, P., Schuldt, H., Simi, M., Springmann, M., and Voicu, L. (2007). DILIGENT: integrating digital library and Grid technologies for a new earth observation research infrastructure. *International Journal on Digital Libraries*, 7(1-2):59–80.

[Candela et al., 2008a] Candela, L., Castelli, D., Ferro, N., Ioannidis, Y., Koutrika, G., Meghini, C., Pagano, P., Ross, S., Soergel, D., Agosti, M., Dobreva, M., Katifori, V., and Schuldt, H. (2008a). The DELOS digital library reference model: Foundations for digital libraries, version 0.98. `http://www.delos.info/files/pdf/ReferenceModel/DELOS_DLReferenceModel_0.98.pdf`.

[Candela et al., 2009a] Candela, L., Castelli, D., Manghi, P., Mikulicic, M., and Pagano, P. (2009a). On foundations of typed data models for digital. In *Post-proceedings of the Fifth Italian Research Conference on Digital Libraries (IRCDL 2009)*, pages 1–11. DELOS Association.

[Candela et al., 2008b] Candela, L., Castelli, D., and Pagano, P. (2008b). gCube: A service-oriented application framework on the grid. *ERCIM News*, 2008(72).

[Candela et al., 2009b] Candela, L., Castelli, D., and Pagano, P. (2009b). D4Science: an e-infrastructure for supporting virtual research environments. In *Post-proceedings of the Fifth Italian Research Conference on Digital Libraries (IRCDL 2009)*, pages 166–169. DELOS Association.

[Candela et al., 2010] Candela, L., Castelli, D., and Pagano, P. (2010). Making virtual research environments in the cloud a reality: the gCube approach. *ERCIM News*, 2010(83):32.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(6):679 –698.

[Caputo et al., 2009] Caputo, B., Pronobis, A., and Jensfelt, P. (2009). Overview of the clef 2009 robot vision track. In *Working Notes of the 2009 Cross Language Evaluation Forum Workshop (CLEF 2009)*.

[Carmel et al., 2002] Carmel, D., Farchi, E., Petruschka, Y., and Soffer, A. (2002). Automatic query refinement using lexical affinities with maximal information gain. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 283–290, New York, NY, USA. ACM.

[Carson et al., 2002] Carson, C., Belongie, S., Greenspan, H., and Malik, J. (2002). Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(8):1026–1038.

[Carson et al., 1999] Carson, C., Thomas, M., Belongie, S., Hellerstein, J. M., and Malik, J. (1999). Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the 3rd International Conference on Visual Information and Information Systems (VISUAL '99)*, pages 509–516.

[Castelli and Bergman, 2002] Castelli, V. and Bergman, L. D., editors (2002). *Image Databases: Search and Retrieval of Digital Imagery*. Wiley, New York, NY, USA.

[Catarci and Santucci, 2001] Catarci, T. and Santucci, G. (2001). The prototype of the dare system. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*, page 609, New York, NY, USA. ACM.

[Cattell and Barry, 2000] Cattell, R. G. G. and Barry, D. K., editors (2000). *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann, San Francisco, California.

[Chalechale et al., 2004] Chalechale, A., Mertins, A., and Naghdy, G. (2004). Edge image description using angular radial partitioning. *IEE Proceedings on Vision, Image and Signal Processing*, 151(2):93–101.

[Chalechale et al., 2005] Chalechale, A., Naghdy, G., and Mertins, A. (2005). Sketch-based image matching using angular partitioning. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(1):28–41.

[Champ, 2008] Champ, H. (2008). 3 billion! WWW page. `http://blog.flickr.net/en/2008/11/03/3-billion/`, last accessed: February 2nd, 2010.

[Champ, 2009] Champ, H. (2009). 4,000,000,000. WWW page. `http://blog.flickr.net/en/2009/10/12/4000000000/`, last accessed: February 2nd, 2010.

[Chang and Fu, 1979] Chang, N.-S. and Fu, K.-s. (1979). Query-by-pictorial-example. In *Proceedings of the Third IEEE International Computer Software and Applications Conference (COMPSAC 79)*, pages 325–330, Los Alamitos, CA, USA. IEEE Computer Society.

[Chang et al., 2001] Chang, S.-F., Sikora, T., and Purl, A. (2001). Overview of the MPEG-7 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):688–695.

[Chatzichristofis and Boutalis, 2008] Chatzichristofis, S. A. and Boutalis, Y. S. (2008). CEDD: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *Proceedings of the 6th International Conference in advanced research on Computer Vision Systems (ICVS 2008)*, pages 312–322, Berlin / Heidelberg. Springer.

[Chatzichristofis and Boutalis, 2010] Chatzichristofis, S. A. and Boutalis, Y. S. (2010). Accurate image retrieval based on compact composite descriptors. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 24(2):207–244.

[Chatzichristofis et al., 2010] Chatzichristofis, S. A., Boutalis, Y. S., and Lux, M. (2010). Combining color and spatial color distribution information in a fuzzy rule based compact composite descriptor. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART 2010) – Revised Selected Papers*, pages 49–60, Berlin / Heidelberg. Springer.

[Chen, 1976] Chen, P. P.-S. (1976). The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1:9–36.

[Chen et al., 2009] Chen, T., Cheng, M.-M., Tan, P., Shamir, A., and Hu, S.-M. (2009). Sketch2Photo: Internet image montage. *ACM Transactions on Graphics (TOG)*, 28:124:1–124:10.

[Cheng et al., 1984] Cheng, D.-Y., Gersho, A., Ramamurthi, B., and Shoham, Y. (1984). Fast search algorithms for vector quantization and pattern matching. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '84)*, volume 9, pages 372–375.

[Chou and DeWitt, 1985] Chou, H.-T. and DeWitt, D. J. (1985). An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th international conference on Very Large Data Bases (VLDB '85)*, pages 127–141. VLDB Endowment.

[Ciaccia et al., 1997] Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB '97)*, pages 426–435, San Francisco, CA, USA. Morgan Kaufmann.

[Ciaccia et al., 1998] Ciaccia, P., Patella, M., and Zezula, P. (1998). Processing complex similarity queries with distance-based access methods. In *Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '98)*, pages 9–23, Berlin / Heidelberg. Springer.

[Ciocca et al., 2011] Ciocca, G., Cusano, C., Santini, S., and Schettini, R. (2011). Prosemantic Features for Content-Based Image Retrieval. In *Understanding Media and Adapting to the User: Proceedings of the 7th International Workshop on Adaptive Multimedia Retrieval (AMR 2009) – Revised Selected Papers*, pages 87–100, Berlin / Heidelberg. Springer.

[Clough et al., 2005] Clough, P., Müller, H., Deselaers, T., Grubinger, M., Lehmann, T. M., Jensen, J., and Hersh, W. (2005). The CLEF 2005 cross-language image retrieval track. In *Proceedings of the 6th Workshop of the Cross-Language Evalution Forum (CLEF 2005) – Revised Selected Papers*, Berlin / Heidelberg. Springer.

[Cootes, 2010] Cootes, T. F. (2010). Deformable object modelling and matching. In *Proceedings of the 10th Asian Conference on Computer Vision (ACCV 2010) – Selected Papers, Part I*, pages 1–10, Berlin / Heidelberg. Springer.

[Cootes et al., 1998] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1998). Active appearance models. In *Proceedings of the 5th European Conference on Computer Vision (ECCV'98)*, volume 3, pages 484–498, Berlin / Heidelberg. Springer.

[Cootes et al., 1995] Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models – their training and application. *Computer Vision and Image Understanding*, 61:38–59.

[Cormen et al., 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, 2nd edition.

[Cox et al., 2000] Cox, I. J., Miller, M. L., Minka, T. P., Papathomas, T. V., and Yianilos, P. N. (2000). The bayesian image retrieval system, PicHunter: theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, 9(1):20–37.

[Cox et al., 1998] Cox, I. J., Miller, M. L., Minka, T. P., and Yianilos, P. N. (1998). An optimized interaction strategy for bayesian relevance feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR' 98)*, pages 553–558, Los Alamitos, CA, USA. IEEE Computer Society.

[Cox et al., 1996] Cox, I. J., Miller, M. L., Omohundro, S. M., and Yianilos, P. N. (1996). PicHunter: Bayesian relevance feedback for image retrieval. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR 1996)*, volume 3, pages 361–369, Los Alamitos, CA, USA. IEEE Computer Society.

[Cox et al., 1997] Cox, I. J., Papathomas, T. V., Ghosn, J., Yianilos, P. N., and Miller, M. L. (1997). Hidden annotation in content based image retrieval. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL '97)*, pages 76–81, Los Alamitos, CA, USA. IEEE Computer Society.

[Crandall et al., 2009] Crandall, D. J., Backstrom, L., Huttenlocher, D., and Kleinberg, J. (2009). Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*, pages 761–770, New York, NY, USA. ACM.

[Cutrell and Guan, 2007] Cutrell, E. and Guan, Z. (2007). What are you looking for?: an eye-tracking study of information usage in web search. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '07)*, pages 407–416, New York, NY, USA. ACM.

[Dander, 2006] Dander, A. (2006). Bildähnlichkeitssuche mit medizinischen Bildern. BSc Thesis supervised by Michael Springmann and Heiko Schuldt at UMIT, Hall i.T., Austria.

[Datar et al., 2004] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry (SCG '04)*, pages 253–262, New York, NY, USA. ACM.

[Datta et al., 2008] Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Transactions on Computing Surveys (CSUR)*, 40(2):5:1–5:60.

[Davis and Wilper, 2011] Davis, D. and Wilper, C. (2011). Fedora 3.5 documentation: Fedora digital object model. `https://wiki.duraspace.org/display/FEDORA35/Fedora+Digital+Object+Model`, last accessed: December 30th, 2011.

[de Silva and Tenenbaum, 2002] de Silva, V. and Tenenbaum, J. B. (2002). Global versus local methods in nonlinear dimensionality reduction. In *Proceedings of the Neural Information Processing Systems Conference (NIPS 2002)*, pages 705–712.

[Del Bimbo, 1999] Del Bimbo, A. (1999). *Visual information retrieval*. Morgan Kaufmann, San Francisco, CA, USA.

[Del Bimbo et al., 2004] Del Bimbo, A., Gradmann, S., and Ioannidis, Y., editors (2004). *Future Research Directions*. 3rd DELOS Brainstorming Workshop Report. `http://www.delos.info/files/pdf/events/2004_Jul_8_10/D8.pdf`.

[Del Bimbo et al., 1998] Del Bimbo, A., Mugnaini, M., Pala, P., and Turco, F. (1998). Visual querying by color perceptive regions. *Pattern Recognition*, 31(9):1241–1253.

[Del Bimbo et al., 2009] Del Bimbo, A., Nunziati, W., and Pala, P. (2009). David: Discriminant analysis for verification of monuments in image data. In *IEEE International Conference on Multimedia and Expo 2009 (ICME 2009)*, pages 334–337, Washington, DC, USA. IEEE Computer Society.

[Del Bimbo and Pala, 1997] Del Bimbo, A. and Pala, P. (1997). Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(2):121–132.

[Deselaers et al., 2008a] Deselaers, T., Deserno, T. M., and Müller, H. (2008a). Automatic medical image annotation in ImageCLEF 2007: Overview, results, and discussion. *Pattern Recognition Letters, Special Issue on Medical Image Annotation in ImageCLEF 2007*, 29(15):1988 – 1995.

[Deselaers et al., 2005] Deselaers, T., Keysers, D., and Ney, H. (2005). Discriminative training for object recognition using image patches. In *Proceedings of the IEEE Com-*

puter on Computer Vision and Pattern Recognition (CVPR 2005), volume 2, pages 157–162.

[Deselaers et al., 2008b] Deselaers, T., Pimenidis, L., and Ney, H. (2008b). Bag-of-Visual-Words Models for Adult Image Classification and Filtering. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)*, pages 1–4.

[Di Sciascio et al., 2002] Di Sciascio, E., M. Donini, F., and Mongiello, M. (2002). Spatial layout representation for query-by-sketch content-based image retrieval. *Pattern Recognition Letters*, 23(13):1599–1612.

[Digital Imaging Group, 2001] Digital Imaging Group (2001). DIG35 specification. metadata for digital images version 1.1.

[Dimai, 1999a] Dimai, A. (1999a). Assessment of effectiveness of content based image retrieval systems. In *Proceedings of the Third International Conference on Visual Information and Information Systems (VISUAL '99)*, pages 525–532.

[Dimai, 1999b] Dimai, A. (1999b). *Invariant scene representations for preattentive similarity assessment: Content based image retrieval exploring low and intermediate level image information*. PhD thesis, ETH Zürich, Switzerland. Diss. Techn. Wiss. ETH Zürich, Nr. 13164 1999.

[Dimai, 1999c] Dimai, A. (1999c). Rotation invariant texture description using general moment invariants and Gabor filters. In *Proceedings of the 11th Scandinavian Conference on Image Analysis*, volume 1, pages 391–398.

[Dublin Core Metadata Initiative, 2011] Dublin Core Metadata Initiative (2011). Dcmi specifications. `http://www.dublincore.org/specifications/`, last accessed on February 6th, 2011.

[Duke Today, 2011] Duke Today (2011). New smartphone app automatically tags photos. WWW page. `http://today.duke.edu/2011/06/tagsense`, last accessed: July 7, 2011.

[Egenhofer, 1997] Egenhofer, M. J. (1997). Query processing in spatial-query-by-sketch. *Journal of Visual Languages & Computing*, 8(4):403–424.

[Eitz and Hays, 2011] Eitz, M. and Hays, J. (2011). Learning to classify human object sketches. In *Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2011 Talks)*, pages 30:1–30:1, New York, NY, USA. ACM.

[Eitz et al., 2009a] Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2009a). A descriptor for large scale image retrieval based on sketched feature lines. In *Sixth Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling*.

[Eitz et al., 2009b] Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2009b). PhotoSketch: A sketch based image query and compositing system. In *ACM SIGGRAPH 2009 Talk Program*.

[Eitz et al., 2010] Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2010). An evaluation of descriptors for large-scale image retrieval from sketched feature lines. *Computers & Graphics*, 34(5):482–498.

[Eitz et al., 2011a] Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2011a). Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624 –1636.

[Eitz et al., 2011b] Eitz, M., Richter, R., Hildebrand, K., Boubekeur, T., and Alexa, M. (2011b). Photosketcher: Interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66.

[Elsweiler et al., 2011] Elsweiler, D., Wilson, M. L., and Kirkegaard-Lunn, B. (2011). Understanding casual-leisure information behaviour. In Spink, A. and Heinström, J., editors, *New Directions in Information Behaviour*. Emerald Group Publishing Limited, Bringley, UK.

[Enser, 2008] Enser, P. (2008). The evolution of visual information retrieval. *Journal of Information Science*, 34(4):531–546.

[Esters and Sander, 2000] Esters, M. and Sander, J. (2000). *Knowledge Discovery in Databases*. Springer, Berlin / Heidelberg.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. `http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html`.

[Falchi et al., 2007] Falchi, F., Kacimi, M., Mass, Y., Rabitti, F., and Zezula, P. (2007). SAPIR: Scalable and distributed image searching. In *Proceedings of the 2nd International Conference on Semantic and Digital Media Technologies (SAMT 2007) – Posters and Demos*, volume 300 of *CEUR Workshop Proceedings*, pages 11–12. CEUR-WS.org.

[Faloutsos and Lin, 1995] Faloutsos, C. and Lin, K.-I. (1995). Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD '95)*, pages 163–174, New York, NY, USA. ACM.

[Ferrari et al., 2010] Ferrari, V., Jurie, F., and Schmid, C. (2010). From images to shape models for object detection. *International Journal of Computer Vision*, 87:284–303.

[Fidel, 1997] Fidel, R. (1997). The image retrieval task: implications for the design and evaluation of image databases. *New Review of Hypermedia and Multimedia*, 3(1):181–199.

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). Updated by RFCs 2817, 5785, 6266.

[Fisher et al., 2004] Fisher, R., Perkins, S., Walker, A., and Wolfart, E. (2004). Hypermedia image processing reference 2 (hipr2) – image processing learning ressources. WWW page. Available online at `http://homepages.inf.ed.ac.uk/rbf/HIPR2/`, last accessed: August 10, 2011.

[Flickner et al., 1995] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. (1995). Query by image and video content: The QBIC system. *Computer*, 28(9):23–32.

[Foster et al., 2001] Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal on High Performance Computing Applications*, 15:200–222.

[Funt and Finlayson, 1995] Funt, B. V. and Finlayson, G. D. (1995). Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(5):522–529.

[Furrie, 2009] Furrie, B. (2009). Understanding MARC bibliographic: Machine-readable cataloging. `http://www.loc.gov/marc/umb/`, last accessed on February 6th, 2011.

[Galleguillos et al., 2008] Galleguillos, C., Babenko, B., Rabinovich, A., and Belongie, S. J. (2008). Weakly supervised object localization with stable segmentations. In *Proceedings of the 10th European Conference on Computer Vision (ECCV 2008)*, volume 1, pages 193–207, Berlin / Heidelberg. Springer.

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Ganea and Brezovan, 2010] Ganea, E. and Brezovan, M. (2010). Graph object oriented database for semantic image retrieval. In *Proceedings of the 14th east European conference on Advances in databases and information systems (ADBIS'10)*, pages 563–566, Berlin/Heidelberg. Springer.

[Gavves and Snoek, 2010] Gavves, E. and Snoek, C. G. (2010). Landmark image retrieval using visual synonyms. In *Proceedings of the 18. ACM International Conference on Multimedia (MM 2010)*, New York, NY, USA. ACM.

[Gerken et al., 2008] Gerken, J., Demarmels, M., Dierdorf, S., and Reiterer, H. (2008). HyperScatter – modellierungs- und zoomtechniken für punktdiagramme. In *Mensch & Computer 2008: Viel mehr Interaktion, 8. Konferenz für interaktive und kooperative Medien*, Munich, Germany. Oldenbourg Verlag.

[Ghemawat et al., 2003] Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03*, pages 29–43, New York, NY, USA. ACM.

[Giangreco, 2010] Giangreco, I. (2010). Inhaltsbasierte Suche in Bildkollektionen mittels Farbskizzen. BSc Thesis supervised by Michael Springmann and Heiko Schuldt at University of Basel, Switzerland.

[Giangreco et al., 2012] Giangreco, I., Springmann, M., Al Kabary, I., and Schuldt, H. (2012). A user interface for query-by-sketch based image retrieval with color sketches. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*, pages 571–572, Heidelberg / Berlin, Germany. Springer.

[Gonçalves et al., 2004] Gonçalves, M. A., Fox, E. A., Watson, L. T., and Kipp, N. A. (2004). Streams, structures, spaces, scenarios, societies (5S): A formal model for digital libraries. *ACM Transactions on Information Systems (TOIS)*, 22:270–312.

[Google Inc., 2010] Google Inc. (2010). Google maps with street view: Privacy. Online Product Information / FAQ. `http://www.google.com/intl/en_us/help/maps/streetview/privacy.html`, last accessed: November 30th, 2010.

[Google Inc., 2011a] Google Inc. (2011a). Google images with sorting. WWW page. http://www.google.com/landing/imagesorting/, last accessed: May 11, 2011.

[Google Inc., 2011b] Google Inc. (2011b). Instant previews. WWW page. http://www.google.com/landing/instantpreviews/, last accessed: May 11, 2011.

[Google Inc., 2011c] Google Inc. (2011c). Search by image. WWW page. http://www.google.com/insidesearch/searchbyimage.html, last accessed: June 15, 2011.

[Google Picasa Help, 2011a] Google Picasa Help (2011a). Name tags: Add name tags in picasa. WWW page. http://picasa.google.com/support/bin/answer.py?answer=156272, last accessed: March 29th, 2011.

[Google Picasa Help, 2011b] Google Picasa Help (2011b). Name tags: Add name tags in picasa web albums. WWW page. http://picasa.google.com/support/bin/answer.py?hl=en&answer=93973, last accessed: March 29th, 2011.

[Gouet and Boujemaa, 2001] Gouet, V. and Boujemaa, N. (2001). Object-based queries using color points of interest. In *IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL 2001)*, pages 30–36, Los Alamitos, CA, USA. IEEE Computer Society.

[Grabs et al., 2001] Grabs, T., Böhm, K., and Schek, H.-J. (2001). PowerDB-IR: Information retrieval on top of a database cluster. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM '01)*, pages 411–418, New York, NY, USA. ACM Press.

[Graham et al., 2002] Graham, A., Garcia-Molina, H., Paepcke, A., and Winograd, T. (2002). Time as essence for photo browsing through personal digital libraries. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 326–335, New York, NY, USA. ACM.

[Grauman and Darrell, 2007] Grauman, K. and Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760.

[Grefenstette, 2010] Grefenstette, G. (2010). Use of semantics in real life applications. Keynote Speech at the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010).

[Grün et al., 2005] Grün, C., Gerken, J., Jetter, H.-C., König, W. A., and Reiterer, H. (2005). MedioVis - a user-centred library metadata browser. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2005)*, pages 174–185, Berlin / Heidelberg. Springer.

[Guan and Cutrell, 2007] Guan, Z. and Cutrell, E. (2007). An eye tracking study of the effect of target rank on web search. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '07)*, pages 417–420, New York, NY, USA. ACM.

[Guenther, 2003] Guenther, R. S. (2003). MODS: the metadata object description schema. *portal: Libraries and the Academy*, 3(1).

[Guimbretière, 2003] Guimbretière, F. (2003). Paper augmented digital documents. In *Proceedings of the 16th annual ACM symposium on User interface software and technology (UIST '03)*, pages 51–60, New York, NY, USA. ACM.

[Güld and Deserno, 2007] Güld, M. O. and Deserno, T. M. (2007). Baseline results for the CLEF 2007 medical automatic annotation task. In *Working Notes of the CLEF Workshop 2007*.

[Güld et al., 2002] Güld, M. O., Kohnen, M., Keysers, D., Schubert, H., Wein, B. B., Bredno, J., and Lehmann, T. M. (2002). Quality of DICOM header information for image categorization. In *Proceedings of SPIE Vol. 4685 – Medical Imaging 2002: PACS and Integrated Medical Information Systems: Design and Evaluation*, volume 4685, pages 280–287.

[Güld et al., 2005] Güld, M. O., Thies, C., Fischer, B., and Lehmann, T. M. (2005). Combining global features for content-based retrieval of medical images. In *Working Notes of the CLEF Workshop 2005*.

[Guttman, 1984] Guttman, A. (1984). R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD '84)*, pages 47–57, New York, NY, USA. ACM Press.

[Harman, 1992] Harman, D. (1992). Relevance feedback revisited. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92)*, pages 1–10, New York, NY, USA. ACM.

[Hays and Efros, 2008] Hays, J. and Efros, A. A. (2008). IM2GPS: estimating geographic information from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Los Alamitos, California, USA. IEEE Computer Society.

[He et al., 2008] He, L., Peng, Z., Everding, B., Wang, X., Han, C. Y., Weiss, K. L., and Wee, W. G. (2008). A comparative study of deformable contour methods on medical image segmentation. *Image and Vision Computing*, 26(2):141–163.

[Hearst, 2006] Hearst, M. A. (2006). Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49:59–61.

[Heesch and Rüger, 2002] Heesch, D. and Rüger, S. (2002). Combining features for content-based sketch retrieval - a comparative evaluation of retrieval performance. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research (ECIR 2002)*, pages 41–52, Berlin / Heidelberg. Springer.

[Heesch and Rüger, 2004] Heesch, D. and Rüger, S. (2004). NNk networks for content-based image retrieval. In *Proceedings of the 26th European Conference on IR Research (ECIR 2004)*, pages 253–266, Berlin / Heidelberg. Springer.

[Heesch et al., 2006] Heesch, D., Yavlinsky, A., and Rüger, S. (2006). NNk networks and automated annotation for browsing large image collections from the world wide web. In *Proceedings of the 14th annual ACM International Conference on Multimedia (MULTIMEDIA '06)*, pages 493–494, New York, NY, USA. ACM.

[Heimann and Meinzer, 2009] Heimann, T. and Meinzer, H.-P. (2009). Statistical shape models for 3d medical image segmentation: A review. *Medical Image Analysis*, 13(4):543–563.

[Hellerstein et al., 1995] Hellerstein, J. M., Naughton, J. F., and Pfeffer, A. (1995). Generalized search trees for database systems. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)*, pages 562–573, San Francisco, CA, USA. Morgan Kaufmann.

[Hersh et al., 2006] Hersh, W. R., Müller, H., Jensen, J. R., Yang, J., Gorman, P. N., and Ruch, P. (2006). Advancing biomedical image retrieval: Development and analysis of a test collection. *Journal of the American Medical Informatics Association (JAMIA)*, 13(5):488–496.

[Hirata and Kato, 1992] Hirata, K. and Kato, T. (1992). Query by visual example. In *Proceedings of the 3rd International Conference on Extending Database Technology (EDBT '92)*, pages 56–71, Berlin / Heidelberg. Springer.

[Hjelmås and Low, 2001] Hjelmås, E. and Low, B. K. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274.

[Huiskes and Lew, 2008] Huiskes, M. J. and Lew, M. S. (2008). The MIR Flickr retrieval evaluation. In *Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval (MIR 2008)*, New York, NY, USA. ACM.

[Huiskes et al., 2010] Huiskes, M. J., Thomee, B., and Lew, M. S. (2010). New Trends and Ideas in Visual Concept Detection: The MIR Flickr Retrieval Evaluation Initiative. In *Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval (MIR 2010)*, pages 527–536, New York, NY, USA. ACM.

[Ioannidis et al., 2005] Ioannidis, Y. E., Maier, D., Abiteboul, S., Buneman, P., Davidson, S. B., Fox, E. A., Halevy, A. Y., Knoblock, C. A., Rabitti, F., Schek, H.-J., and Weikum, G. (2005). Digital library information-technology infrastructures. *International Journal on Digital Libraries*, 5(4):266–274.

[Ioannidis et al., 2008] Ioannidis, Y. E., Milano, D., Schek, H.-J., and Schuldt, H. (2008). DelosDLMS - from the DELOS vision to the implementation of a future digital library management system. *International Journal on Digital Libraries, Special Issue on Very Large Digital Libraries*, 9(2):101–114.

[Iordanov, 2010] Iordanov, B. (2010). HyperGraphDB: A generalized graph database. In *Proceedings of the 1st International Workshop on Graph Database (IWGD 2010), in Web-Age Information Management (WAIM 2010) – Revised Selected Papers*, pages 25–36, Berlin/Heidelberg. Springer.

[Ishikawa et al., 1998] Ishikawa, Y., Subramanya, R., and Faloutsos, C. (1998). MindReader: Querying databases through multiple examples. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*, pages 218–227, San Francisco, CA, USA. Morgan Kaufmann.

[J. Paul Getty Museum, 2011] J. Paul Getty Museum (2011). Offline meets online at the getty: Getty is first museum to provide expanded google goggles experi-

ence to visitors. WWW page. http://www.getty.edu/news/press/center/googlegoggles.html, last accessed: June 27, 2011.

[Jackson, 1975] Jackson, M. A. (1975). *Principles of Program Design*. Academic Press, Inc., Orlando, FL, USA.

[Jacobs et al., 1995] Jacobs, C. E., Finkelstein, A., and Salesin, D. H. (1995). Fast multiresolution image querying. In *Proceedings of the 22nd Annual Conference on Computer Graphics (SIGGRAPH '95)*, pages 277–286.

[Jain and Vailaya, 1996] Jain, A. K. and Vailaya, A. (1996). Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244.

[Jain and Vailaya, 1998] Jain, A. K. and Vailaya, A. (1998). Shape-based retrieval: A case study with trademark image databases. *Pattern Recognition*, 31(9):1369–1390.

[Janiszewski, 1998] Janiszewski, C. (1998). The influence of display characteristics on visual exploratory search behavior. *The Journal of Consumer Research*, 25(3):290–301.

[Järvelin and Ingwersen, 2004] Järvelin, K. and Ingwersen, P. (2004). Information seeking research needs extension towards tasks and technology. *Information Research*, 10(1). http://InformationR.net/ir/10-1/paper212.html, last accessed: October 8th, 2010.

[Java Community Process, 2003] Java Community Process (2003). JSR-000168 Portlet Specification (Final Release). http://www.jcp.org/en/jsr/detail?id=168.

[Java Community Process, 2006] Java Community Process (2006). JSR-000170 Content Repository for Java technology API (Final Release). http://www.jcp.org/en/jsr/detail?id=170.

[Java Community Process, 2009] Java Community Process (2009). JSR-000283 Content Repository for Java Technology API 2.0 (Final Release). http://www.jcp.org/en/jsr/detail?id=283.

[Jegou et al., 2008] Jegou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision: Part I (ECCV '08)*, pages 304–317, Berlin, Heidelberg. Springer-Verlag.

[JETIA, 2010] JETIA (2010). Exchangeable image file format for digital still cameras: Exif Unified Version 2.3. Standard published by the AV&IT Standardization Committee of the Japan Electronics and Information Technology Industries Association (JETIA) and Camera & Imaging Products Association (CIPA). JEITA CP-3451B / DC-008-2010, established in April 2002, revised in April 2010, http://www.jeita.or.jp/cgi-bin/standard_e/list.cgi?cateid=1&subcateid=4, last accessed: November 12th, 2010.

[Jetter et al., 2005] Jetter, H.-C., Gerken, J., König, W. A., Grün, C., and Reiterer, H. (2005). Hypergrid - accessing complex information spaces. In *Proceedings of the 19th British HCI Group Annual Conference 2005 (HCI UK 2005): People and Computers XIX - The Bigger Picture*, pages 349–364, Berlin / Heidelberg. Springer.

[Jing et al., 2004] Jing, F., Li, M., Zhang, H., and Zhang, B. (2004). Relevance feedback in region-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):672–681.

[Jing and Baluja, 2008] Jing, Y. and Baluja, S. (2008). VisualRank: Applying PageRank to large-scale image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30:1877–1890.

[Jörgensen, 1998] Jörgensen, C. (1998). Attributes of images in describing tasks. *Information Processing and Management*, 34:161–174.

[Jörgensen, 2003] Jörgensen, C. (2003). *Image retrieval: theory and research*. Scarecrow Press, Inc., Lanham, MD, USA.

[Jörgensen and Jörgensen, 2005] Jörgensen, C. and Jörgensen, P. (2005). Image querying by image professionals. *Journal of the American Society for Information Science and Technology (JASIST)*, 56:1346–1359.

[Kadir and Brady, 2001] Kadir, T. and Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105.

[Kalogerakis et al., 2009] Kalogerakis, E., Vesselova, O., Hays, J., Efros, A. A., and Hertzmann, A. (2009). Image sequence geolocation with human travel priors. In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV 2009)*, pages 253–260, Los Alamitos, California, USA. IEEE Computer Society.

[Karanastasi et al., 2006] Karanastasi, A., Zotos, A., and Christodoulakis, S. (2006). User interactions with multimedia repositories using natural language interfaces – OntoNL: an architectural framework and its implementation. *Journal of Digital Information Management*, 4(4):220–226.

[Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331. 10.1007/BF00133570.

[Kay, 1972] Kay, A. C. (1972). A personal computer for children of all ages. In *Proceedings of the ACM National Conference*.

[Kazi, 2004] Kazi, S. A. (2004). A conceptual framework for web-based intelligent learning environments using SCORM-2004. In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT'04)*, pages 12–15, Los Alamitos, CA, USA. IEEE Computer Society.

[Ke and Sukthankar, 2004] Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, 2:506–513.

[Keogh, 2011] Keogh, E. (2011). Dangers of dimensionality reduction. Presentation Slides. Available online at `http://www.cs.ucr.edu/~eamonn/DangersOfDimensionalityReduction.pptx`, PDF version at `http://www.cs.ucr.edu/~eamonn/DangersOfDimensionalityReduction.pdf`.

[Keogh and Kasetty, 2002] Keogh, E. and Kasetty, S. (2002). On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pages 102–111, New York, NY, USA. ACM.

[Keogh et al., 2006] Keogh, E., Wei, L., Xi, X., Lee, S.-H., and Vlachos, M. (2006). LB_-Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006)*, pages 882–893. VLDB Endowment.

[Keogh et al., 2009] Keogh, E., Wei, L., Xi, X., Vlachos, M., Lee, S.-H., and Protopapas, P. (2009). Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *VLDB Journal*, 18(3):45–66.

[Keysers et al., 2003] Keysers, D., Dahmen, J., Ney, H., Wein, B. B., and Lehmann, T. M. (2003). Statistical framework for model-based image retrieval in medical applications. *Journal of Electronic Imaging*, 12:59–68.

[Keysers et al., 2007] Keysers, D., Deselaers, T., Gollan, C., and Hermann, N. (2007). Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(8):1422–1435.

[Keysers et al., 2004] Keysers, D., Gollan, C., and Ney, H. (2004). Local context in non-linear deformation models for handwritten character recognition. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04) Volume 4*, pages 511–514, Los Alamitos, CA, USA. IEEE Computer Society.

[Khan et al., 2009] Khan, R., Mease, D., and Patel, R. (2009). The impact of result abstracts on task completion time. In *Proceedings of the Workshop on Web Search Result Summarization and Presentation (WSSP 2009)*.

[Kimia, 2002] Kimia, B. B. (2002). Shape representation for image retrieval. In [Castelli and Bergman, 2002], pages 345–372.

[Kogler and Lux, 2010] Kogler, M. and Lux, M. (2010). Bag of visual words revisited: An exploratory study on robust image retrieval exploiting fuzzy codebooks. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining (MDMKDD '10)*, pages 3:1–3:6, New York, NY, USA. ACM.

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

[Kopp, 2009] Kopp, D. (2009). Implementation and evaluation of query by sketch. MSc Thesis supervised by Michael Springmann and Heiko Schuldt at University of Basel, Switzerland.

[Koutamanis, 2005] Koutamanis, A. (2005). Sketching with digital pen and paper. In *Proceedings of the 11th International Computer Aided Architectural Design (CAAD Futures 2005)*, pages 321 – 330, Berlin / Heidelberg. Springer.

[Kreuzer, 2010] Kreuzer, R. (2010). Query by sketching using digital pen and interactive paper. MSc Thesis supervised by Michael Springmann and Heiko Schuldt at University of Basel, Switzerland.

[Kreuzer et al., 2012] Kreuzer, R., Springmann, M., Al Kabary, I., and Schuldt, H. (2012). An interactive paper and digital pen interface for query-by-sketch image retrieval. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*, pages 317–328, Heidelberg / Berlin, Germany. Springer.

[Krishnan, 2009] Krishnan, R. (2009). Explore images with google image swirl. WWW page. http://googleresearch.blogspot.com/2009/11/explore-images-with-google-image-swirl.html, last accessed: May 11, 2011.

[Krishnan, 2010] Krishnan, R. (2010). Beyond instant results: Instant previews. WWW page. http://googleblog.blogspot.com/2010/11/beyond-instant-results-instant-previews.html, last accessed: May 11, 2011.

[Kurita and Kato, 1993] Kurita, T. and Kato, T. (1993). Learning of personal visual impression for image database systems. In *Proceedings of the Second International Conference on Document Analysis and Recognition (ICDAR '93)*, pages 547–552, Los Alamitos, CA, USA. IEEE Computer Society.

[Kurtenbach, 2004] Kurtenbach, G. (2004). Notes on the history of radial menus, pie menus and marking menus. Available online at http://www.autodeskresearch.com/pdf/unpublished/NotesonHistoryofRadialMenus.pdf, last accessed: December 2nd, 2011.

[Kurtenbach and Buxton, 1994] Kurtenbach, G. and Buxton, W. (1994). User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing system (CHI '94)*, pages 258–264, New York, NY, USA. ACM.

[Lagoze et al., 2006] Lagoze, C., Payette, S., Shin, E., and Wilper, C. (2006). Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6:124–138.

[Lassila and Swick, 1999] Lassila, O. and Swick, R., editors (1999). *Resource Description Framework (RDF) Model and Syntax Specification*. The World Wide Web Consortium. W3C Recommendation, available at http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, last accessed: February 3rd, 2011.

[Lee and Hu, 2003] Lee, K. P. and Hu, J. (2003). XML Schema Representation of DICOM Structured Reporting. *Journal of the American Medical Informatics Association*, 10(2):213–223.

[Lehmann et al., 2005] Lehmann, T. M., Güld, M. O., Deselaers, T., Keysers, D., Schubert, H., Spitzer, K., Ney, H., and Wein, B. B. (2005). Automatic categorization of medical images for content-based retrieval and data mining. *Computerized Medical Imaging and Graphics*, 29(2):143–155.

[Lehmann et al., 2002] Lehmann, T. M., Schubert, H., Keysers, D., Kohnen, M., and Wein, B. B. (2002). A monohierarchical multiaxial classification code for medical images in content-based retrieval. In *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI 2002)*, pages 313–316, Washington, DC, USA. IEEE Computer Society.

[Lehmann et al., 2003a] Lehmann, T. M., Schubert, H., Keysers, D., Kohnen, M., and Wein, B. B. (2003a). The irma code for unique classification of medical images. In

Proceedings of SPIE Vol. 5033 – Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation, volume 5033, pages 440–451.

[Lehmann et al., 2003b] Lehmann, T. M., Schubert, H., Keysers, D., Kohnen, M., and Wein, B. B. (2003b). A novel keycode for mono-hierarchical and multi-axial categorization of radiological images. *Informatik, Biometrie und Epidemiologie in Medizin und Biologie*, 34(3):229–230.

[Lepetit and Fua, 2006] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(9):1465–1479.

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2011)*, Los Alamitos, CA, USA. IEEE Computer Society.

[Leventhal, 2008] Leventhal, A. (2008). Flash storage memory. *Communications of the ACM (CACM)*, 51:47–51.

[Li and Wang, 2008] Li, J. and Wang, J. Z. (2008). Real-time computerized annotation of pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30:985–1002.

[Li et al., 2009a] Li, S., Lee, M.-C., and Pun, C.-M. (2009a). Complex zernike moments features for shape-based image retrieval. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(1):227–237.

[Li et al., 2009b] Li, Y., Crandall, D. J., and Huttenlocher, D. P. (2009b). Landmark classification in large-scale image collections. In *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV 2009)*, pages 1957–1964, Los Alamitos, California, USA. IEEE Computer Society.

[Li et al., 2002] Li, Y., Kuo, C.-C. J. K., and Wan, X. (2002). Introduction to content-based image retrieval – overview of key techniques. In [Castelli and Bergman, 2002], pages 261–284.

[Liang et al., 2005] Liang, S., Sun, Z., and Li, B. (2005). Sketch retrieval based on spatial relations. In *International Conference on Computer Graphics, Imaging and Visualization (CGIV 2005)*, pages 24–29, Los Alamitos, CA, USA. IEEE Computer Society.

[Lin and Keogh, 2003] Lin, J. and Keogh, E. (2003). Dangers of dimensionality reduction. Demo Code / Video. Available online at `http://www.cs.gmu.edu/~jessica/DimReducDanger.htm`.

[Loupias et al., 2000] Loupias, E., Sebe, N., Bres, S., and Jolion, J.-M. (2000). Wavelet-based salient points for image retrieval. In *Proceedings of the 7th IEEE International Conference on Image Processing (ICIP 2000)*, volume II, pages 518–521, Los Alamitos, CA, USA. IEEE Computer Society.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of 7th International Conference on Computer Vision (ICCV'99)*, volume 2, pages 1150–1158.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110.

[Lux and Chatzichristofis, 2008] Lux, M. and Chatzichristofis, S. A. (2008). LIRe: Lucene Image Retrieval: an extensible Java CBIR library. In *Proceeding of the 16th ACM International Conference on Multimedia (MM '08)*, pages 1085–1088, New York, NY, USA. ACM.

[Lux et al., 2010] Lux, M., Kofler, C., and Marques, O. (2010). A classification scheme for user intentions in image search. In *Proceedings of the 28th of the International Conference on Human factors in Computing Systems (CHI '10) – Extended Abstracts*, pages 3913–3918, New York, NY, USA. ACM.

[Ma and Manjunath, 1998] Ma, W.-Y. and Manjunath, B. S. (1998). A texture thesaurus for browsing large aerial photographs. *Journal of the American Society for Information Science*, 49(7):633–648.

[Ma and Manjunath, 2000] Ma, W.-Y. and Manjunath, B. S. (2000). EdgeFlow: a technique for boundary detection and image segmentation. *IEEE Transactions on Image Processing*, 9(8):1375–1388.

[Malisiewicz et al., 2011] Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the 13th International Conference on Computer Vision (ICCV 2011)*.

[Malladi et al., 1995] Malladi, R., Sethian, J. A., and Vemuri, B. C. (1995). Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(2):158–175.

[Manjunath and Ma, 1996] Manjunath, B. and Ma, W. (1996). Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(8):837–842.

[Manjunath et al., 2001] Manjunath, B. S., Ohm, J.-R., Vasudevan, V. V., and Yamada, A. (2001). Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715.

[Manning et al., 2009] Manning, C. D., Raghavan, P., and Schütze, H. (2009). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England, online edition, last updated 2009.04.01 edition.

[Marchionini, 2006] Marchionini, G. (2006). Exploratory search: from finding to understanding. *Communications of the ACM*, 49:41–46.

[Markkula and Sormunen, 2000] Markkula, M. and Sormunen, E. (2000). End-user searching challenges indexing practices in the digital newspaper photo archive. *Information Retrieval*, 1:259–285.

[Martin et al., 2004] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(5):530–549.

[Matas et al., 2004] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Computing*, 22(10):761–767.

[May, 2004] May, A. (2004). Web-based image and video navigation. Final project report, Imperial College London, London, UK. Available at `http://technologies.kmi.open.ac.uk/ubase/papers.html`.

[McDonald and Tait, 2003] McDonald, S. and Tait, J. (2003). Search strategies in content-based image retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and Development in Informaion Retrieval (SIGIR '03)*, pages 80–87, New York, NY, USA. ACM.

[Mehta et al., 2008] Mehta, B., Nangia, S., Gupta, M., and Nejdl, W. (2008). Detecting image spam using visual features and near duplicate detection. In *Proceeding of the 17th international conference on World Wide Web (WWW '08)*, pages 497–506, New York, NY, USA. ACM.

[Mikolajczyk and Schmid, 2001] Mikolajczyk, K. and Schmid, C. (2001). Indexing based on scale invariant interest points. In *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV 2001)*, volume 1, pages 525–531, Los Alamitos, CA, USA. IEEE Computer Society.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27:1615–1630.

[Mildenberger et al., 2002] Mildenberger, P., Eichelberg, M., and Martin, E. (2002). Introduction to the DICOM standard. *European Radiology*, 12:920–927.

[Mitchell, 2010] Mitchell, J. (2010). Making photo tagging easier. WWW page. `http://www.facebook.com/blog.php?post=467145887130`, last accessed: March 29th, 2011.

[Mlivoncic, 2006] Mlivoncic, M. (2006). *Efficient evaluation techniques for complex similarity queries on large media collections*. PhD thesis, ETH Zürich, Switzerland. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 16600, 2006.

[Mlivoncic et al., 2004a] Mlivoncic, M., Schuler, C., and Türker, C. (2004a). Hyperdatabase infrastructure for management and search of multimedia collections. In *Pre-proceedings of the Sixth Thematic Workshop of the EU Network of Excellence DELOS on Digital Library Architectures: Peer-to-Peer, Grid, and Service-Orientation*, pages 25–36. Edizioni Libreria Progetto, Padova.

[Mlivoncic et al., 2004b] Mlivoncic, M., Schuler, C., Türker, C., and Balko, S. (2004b). A service-oriented grid infrastructure for multimedia management and search. In *Proceedings of the 6th Thematic Workshop of the EU Network of Excellence DELOS on Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures, Revised Selected Papers*, pages 167–187, Berlin / Heidelberg. Springer.

[Mlivoncic and Weber, 2003] Mlivoncic, M. and Weber, R. (2003). A filter and refinement approach to RBIR. In *Proceedings of the DELOS Workshop on Multimedia Contents in Digital Libraries*, Chania, Crete, Greece.

[Morel and Schurter, 2009] Morel, E. and Schurter, T. (2009). Nachträgliches Geotagging von Bildern mittels inhaltsbasierter Suche. BSc Thesis supervised by Michael Springmann and Heiko Schuldt at University of Basel, Switzerland.

[Morrison et al., 2009] Morrison, D., Marchand-Maillet, S., and Bruno, E. (2009). TagCaptcha: Annotating images with CAPTCHAs. In *Proceedings of the Human Computation Workshop (HCOMP2009) - co-located with ACM-KDD 2009*, Paris, France.

[Morrison et al., 2010] Morrison, D., Marchand-Maillet, S., and Bruno, E. (2010). TagCaptcha: Annotating images with CAPTCHAs. In *Proceedings of the International Conference on Multimedia (MM '10) – Demo Program*, pages 1557–1558, New York, NY, USA. ACM.

[Mulhem and Lim, 2003] Mulhem, P. and Lim, J.-H. (2003). Home photo retrieval: Time matters. In *Proceedings of the 2nd International Conference on Image and Video Retrieval (CIVR'03)*, pages 321–330, Berlin / Heidelberg. Springer.

[Müller et al., 2008] Müller, H., Deselaers, T., Kim, E., Kalpathy-Cramer, J., Deserno, T. M., and Hersh, W. (2008). Overview of the ImageCLEF 2007 medical retrieval and annotation tasks. In *Proceedings of the Eigth Workshop of the Cross-Language Evaluation Forum (CLEF 2007)*, Berlin / Heidelberg. Springer.

[Müller et al., 2000] Müller, H., Müller, W., Marchand-Maillet, S., Pun, T., and Squire, D. M. (2000). Strategies for positive and negative relevance feedback in image retrieval. In *15th International Conference on Pattern Recognition (ICPR '00)*, volume 1, pages 5043–5046, Los Alamitos, CA, USA. IEEE Computer Society.

[Murphy-Chutorian and Rosenberg, 2009] Murphy-Chutorian, E. and Rosenberg, C. (2009). Similar Images graduates from Google Labs. WWW page. `http://googleblog.blogspot.com/2009/10/similar-images-graduates-from-google.html`, last accessed: May 13, 2011.

[Myers, 1985] Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '85)*, pages 11–17, New York, NY, USA. ACM.

[Niblack et al., 1993] Niblack, W., Barber, R., Equitz, W., Flickner, M. D., Glasman, E. H., Petkovic, D., Yanker, P., Faloutsos, C., and Taubin, G. (1993). The QBIC Project: querying images by content, using color, texture, and shape. In Niblack, W., editor, *Proceedings of SPIE Vol. 1908 Storage and Retrieval for Image and Video Databases*, pages 173–187.

[Nielsen, 1993] Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann, San Francisco, CA, USA.

[Nielsen and Phillips, 1993] Nielsen, J. and Phillips, V. L. (1993). Estimating the relative usability of two interfaces: heuristic, formal, and empirical methods compared. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems (CHI '93)*, pages 214–221, New York, NY, USA. ACM.

[Norrie et al., 2006] Norrie, M. C., Signer, B., and Weibel, N. (2006). General framework for the rapid development of interactive paper applications. In *Proceedings of the 1st International Workshop on Collaborating over Paper and Digital Documents (CoPADD 2006)*, pages 9–12.

[Nowak et al., 2010] Nowak, S., Hanbury, A., and Deselaers, T. (2010). Object and concept recognition for image retrieval. In Müller, H., Clough, P., Deselaers, T., and

Caputo, B., editors, *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*, volume 32 of *The Information Retrieval Series*, chapter 11, pages 199–220. Springer, Berlin / Heidelberg, 1 edition.

[Ntousias et al., 2008] Ntousias, A., Gioldasis, N., Tsinaraki, C., and Christodoulakis, S. (2008). Rich metadata and context capturing through CIDOC/CRM and MPEG-7 interoperability. In *Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval (CIVR '08)*, pages 151–160, New York, NY, USA. ACM.

[Nussbaumer and Haslhofer, 2007] Nussbaumer, P. and Haslhofer, B. (2007). Putting the CIDOC CRM into practice - experiences and challenges. Technical Report TR-200, University of Vienna. available at `http://eprints.cs.univie.ac.at/404/`.

[OASIS, 2006a] OASIS (2006a). Web Services Resource 1.2 (WS-Resource). `http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf`.

[OASIS, 2006b] OASIS (2006b). Web Services Resource Framework (WSRF) v.1.2. `http://www.oasis-open.org/committees/wsrf`.

[OASIS, 2010] OASIS (2010). Content Management Interoperability Services (CMIS) Version 1.0. `http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html`.

[Oates, 2008] Oates, G. (2008). Many hands make light work. WWW page. `http://blog.flickr.net/en/2008/01/16/many-hands-make-light-work/`, last accessed: February 2nd, 2010.

[O'Brien and Toms, 2008] O'Brien, H. L. and Toms, E. G. (2008). What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology (JASIST)*, 59:938–955.

[Odio, 2010] Odio, S. (2010). Making facebook photos better. WWW page. `http://blog.facebook.com/blog.php?post=403838582130`, last accessed: March 29th, 2011.

[Odio, 2011] Odio, S. (2011). More beautiful photos. WWW page. `http://blog.facebook.com/blog.php?post=432670242130`, last accessed: March 29th, 2011.

[Ojala et al., 2002] Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24:971–987.

[Oliva and Torralba, 2006] Oliva, A. and Torralba, A. (2006). Building the gist of a scene: the role of global image features in recognition. In *Visual Perception Fundamentals of Awareness: Multi-Sensory Integration and High-Order Perception*, volume 155, Part B of *Progress in Brain Research*, pages 23–36. Elsevier.

[Oliver et al., 2007] Oliver, A., Lladó, X., Martí, J., Marti, R., and Freixenet, J. (2007). False positive reduction in breast mass detection using two-dimensional PCA. In *Proceedings of the Third Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2007) Part II*, pages 154–161, Berlin / Heidelberg. Springer.

[Opelt et al., 2006] Opelt, A., Pinz, A., and Zisserman, A. (2006). A boundary-fragment-model for object detection. In *Proceedings of the 9th European Conference on Computer Vision (ECCV 2006)*, volume 2, pages 575–588, Berlin / Heidelberg. Springer.

[Opelt et al., 2008] Opelt, A., Pinz, A., and Zisserman, A. (2008). Learning an alphabet of shape and appearance for multi-class object detection. *International Journal of Computer Vision*, 80(1):16–44.

[Open Archives Initiative, 2008] Open Archives Initiative (2008). ORE specification version 1.0 - abstract data model. `http://www.openarchives.org/ore/1.0/datamodel`.

[Open Grid Forum, 2008] Open Grid Forum (2008). The storage resource manager interface specification version 2.2. `http://www.ggf.org/documents/GFD.129.pdf`.

[Ornager, 1995] Ornager, S. (1995). The newspaper image database: empirical supported analysis of users' typology and word association clusters. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95)*, pages 212–218, New York, NY, USA. ACM.

[Ortega-Binderberger and Mehrotra, 2004] Ortega-Binderberger, M. and Mehrotra, S. (2004). Relevance feedback techniques in the MARS image retrieval system. *Multimedia Systems*, 9(6):535–547.

[Ozuysal et al., 2010] Ozuysal, M., Calonder, M., Lepetit, V., and Fua, P. (2010). Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(3):448–461.

[Pala and Santini, 1999] Pala, P. and Santini, S. (1999). Image retrieval by shape and texture. *Pattern Recognition*, 32(3):517–527.

[Park et al., 2000] Park, S. J., Park, D. K., and Won, C. S. (2000). Core experiments on MPEG-7 edge histogram descriptor. *MPEG document M5984*.

[Pass et al., 1996] Pass, G., Zabih, R., and Miller, J. (1996). Comparing images using color coherence vectors. In *Proceedings of the fourth ACM International Conference on Multimedia (MM '96)*, pages 65–73, New York, NY, USA. ACM.

[Patella and Ciaccia, 2008] Patella, M. and Ciaccia, P. (2008). The many facets of approximate similarity search. In *1st International Workshop on Similarity Search and Applications (SISAP 2008)*, pages 308–319.

[Pavlidis, 2008] Pavlidis, T. (2008). Limitations of content-based image retrieval. invited plenary talk at the 19th International Conference on Pattern Recognition (ICPR 2008). Slides can be found in `http://www.theopavlidis.com/technology/CBIR/PaperB/icpr08.htm`, with an an extended version at `http://www.theopavlidis.com/technology/CBIR/PaperB/vers3.htm`.

[Pavlidis, 2009a] Pavlidis, T. (2009a). Why general artificial intelligence (ai) is so hard. Seminar given at Stony Brook University on October 29, 2008. Slides can be found in `http://theopavlidis.com/technology/MachineIntel/Oct09_2009slides.htm`.

[Pavlidis, 2009b] Pavlidis, T. (2009b). Why meaningful automatic tagging of images is very hard. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2009)*, pages 1432–1435, Los Alamitos, CA, USA. IEEE Computer Society.

[Phillips et al., 2000] Phillips, P. J., Moon, H., Rizvi, S. A., and Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(10):1090 – 1104.

[Picard, 1995] Picard, R. W. (1995). Computer learning of subjectivity. *ACM Computing Surveys (CSUR)*, 27:621–623.

[Pinheiro, 2007] Pinheiro, A. M. (2007). Image description using scale-space edge pixel directions histogram. In *Proceedings of the 2nd International Workshop on Semantic Media Adaptation and Personalization (SMAP '07)*, pages 211–218, Los Alamitos, CA, USA. IEEE Computer Society.

[Pinheiro, 2010] Pinheiro, A. M. (2010). The angular orientation partition edge descriptor. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010)*, pages 1250–1253, Los Alamitos, CA, USA. IEEE Computer Society.

[Platt, 2005] Platt, J. C. (2005). FastMap, MetricMap, and Landmark MDS are all Nyström Algorithms. In *Proceedings of 10th International Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 261–268. Society for Artificial Intelligence and Statistics.

[Pollefeys and Van Gool, 2002] Pollefeys, M. and Van Gool, L. (2002). From images to 3d models. *Communications of the ACM*, 45:50–55.

[Pollefeys et al., 2004] Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59:207–232.

[Popper, 1959] Popper, K. R. (1959). *The logic of scientific discovery*. Hutchinson/Routledge/Taylor & Francis Group Ltd, London/Oxford, UK.

[Popper, 2002] Popper, K. R. (2002). *Logik der Forschung*. Mohr Siebeck, Tübingen, Germany, reprint of 10th revised and extended edition.

[Porkaew et al., 1999a] Porkaew, K., Mehrota, S., Ortega, M., and Chakrabarti, K. (1999a). Similarity search using multiple examples in MARS. In *Proceedings of the 3rd International Conference on Visual Information and Information Systems (VISUAL '99)*, pages 68–75, Berlin / Heidelberg. Springer.

[Porkaew et al., 1999b] Porkaew, K., Ortega, M., and Mehrota, S. (1999b). Query reformulation for content based multimedia retrieval in MARS. In *IEEE International Conference on Multimedia Computing and Systems (ICMCS '99)*, volume 2, pages 747–751, Los Alamitos, CA, USA. IEEE Computer Society.

[Press et al., 2007] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3 edition.

[Pronobis et al., 2010] Pronobis, A., Fornoni, M., Christensen, H. I., and Caputo, B. (2010). The robot vision track at imageclef 2010. In *Working Notes of the 2010 Conference on Multilingual and Multimodal Information Access Evaluation (CLEF 2010)*.

[Qin et al., 2011] Qin, C., Bao, X., Choudhury, R. R., and Nelakuditi, S. (2011). TagSense: A smartphone-based approach to automatic image tagging. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys 2011)*, pages 1–14, New York, NY, USA. ACM.

[Rabinovich et al., 2007] Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Proceedings of the 11th IEEE International Conference on Computer Vision(ICCV 2007)*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.

[Reiterer and Büring, 2009] Reiterer, H. and Büring, T. (2009). Zooming techniques. In Liu, L. and Özsu, T. M., editors, *Encyclopedia of Database Systems*, pages 3684–3689. Springer, Berlin / Heidelberg.

[Robertson, 2006] Robertson, S. (2006). On GMAP and other transformations. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM'06)*, pages 78–83, New York, NY, USA. ACM.

[Rocchio, 1971] Rocchio, J. J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ.

[Rodden et al., 2001] Rodden, K., Basalaj, W., Sinclair, D., and Wood, K. (2001). Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '01)*, pages 190–197, New York, NY, USA. ACM.

[Roddy, 1991] Roddy, K. (1991). Subject access to visual resources: What the 90s might portend. *Libr. Hi Tech*, 9(1):45–49.

[Romdhani et al., 2006] Romdhani, S., Ho, J., Vetter, T., and Kriegman, D. J. (2006). Face recognition using 3-d models: Pose and illumination. *Proceedings of the IEEE*, 94(11).

[Rorissa, 2007] Rorissa, A. (2007). Relationships between perceived features and similarity of images: A test of tversky's contrast model. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(10):1401–1418.

[Rorissa et al., 2008] Rorissa, A., Clough, P., and Deselaers, T. (2008). Exploring the relationship between feature and perceptual visual spaces. *Journal of the American Society for Information Science and Technology (JASIST)*, 59(5):770–784.

[Rosenberg et al., 2009] Rosenberg, C., Hertzfeld, A., and Cohen, M. (2009). Hard at play in Google Labs with Similar Images and Google News Timeline. WWW page. `http://googleblog.blogspot.com/2009/04/hard-at-play-in-google-labs-with.html`, last accessed: May 13, 2011.

[Roth and Ommer, 2006] Roth, V. and Ommer, B. (2006). Exploiting low-level image segmentation for object recognition. In *Proceedings of the 28th Annual Symposium of*

the German Association for Pattern Recognition (DAGM 2006), pages 11–20, Berlin / Heidelberg. Springer.

[Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314.

[Rowley et al., 1998] Rowley, H., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(1):23–38.

[Rui et al., 1998] Rui, Y., Huang, T., Ortega, M., and Mehrotra, S. (1998). Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655.

[Rui et al., 1997] Rui, Y., Huang, T. S., and Mehrotra, S. (1997). Content-based image retrieval with relevance feedback in MARS. In *Proceedings of the 1997 International Conference on Image Processing (ICIP '97)*, volume 2, pages 815–818.

[Ruske, 2008] Ruske, S. (2008). Distanzverteilung und Distanznormalisierung für inhaltsbasierte Bildsuche. external Diploma Thesis supervised by Georg Paul at Otto-von-Guericke-University Magdeburg, Germany, performed at University of Basel supervised by Michael Springmann and Heiko Schuldt.

[Russell et al., 2008] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173.

[Safar and Shahabi, 2002] Safar, M. H. and Shahabi, C. (2002). *Shape Analysis and Retrieval of Multimedia Objects*. Kluwer Academic Publishers, Norwell, MA, USA.

[Sajda et al., 2010] Sajda, P., Pohlmeyer, E., Wang, J., Parra, L. C., Christoforou, C., Dmochowski, J., Hanna, B., Bahlmann, C., Singh, M. K., and Chang, S.-F. (2010). In a blink of an eye and a switch of a transistor: Cortically coupled computer vision. *Proceedings of the IEEE*, 98(3):462–478.

[Salton and Buckley, 1990] Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4).

[Santini, 2008] Santini, S. (2008). Ontology: Use and Abuse. In *Proceedings of the 7th International Workshop on Adaptive Multimedia Retrieval: Retrieval, User, and Semantics (AMR 2007) – Revised Selected Papers*, pages 17–31, Berlin / Heidelberg. Springer.

[Santini and Gupta, 2010] Santini, S. and Gupta, A. (2010). "Disputatio" on the Use of Ontologies in Multimedia. In *Proceedings of the 18. ACM International Conference on Multimedia (MM 2010)*, pages 1723–1728, New York, NY, USA. ACM.

[Santini and Jain, 1995] Santini, S. and Jain, R. (1995). Similarity Matching. In *Recent Developments in Computer Vision: Proceedings of the Second Asian Conference on Computer Vision (ACCV '95) – Invited Session Papers*, pages 571–580, Berlin / Heidelberg. Springer.

[Santini and Jain, 1999] Santini, S. and Jain, R. (1999). Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21:871–883.

[Schek and Schuldt, 2008] Schek, H.-J. and Schuldt, H. (2008). The hyperdatabase project – from the vision to realizations. In *Proceedings of the 25th British National Conference on Databases (BNCOD)*, pages 207–226.

[Schindler and Suter, 2008] Schindler, K. and Suter, D. (2008). Object detection by global contour shape. *Pattern Recognition*, 41(12):3736–3748.

[Schmidt, 2006] Schmidt, I. (2006). *Ähnlichkeitssuche in Multimedia-Datenbanken: Retrieval, Suchalgorithmen and Anfragebehandlung*. Oldenbourg, Munich, Germany.

[Schonfeld, 2009] Schonfeld, E. (2009). Who has the most photos of them all? hint: It is not facebook. WWW page. `http://techcrunch.com/2009/04/07/who-has-the-most-photos-of-them-all-hint-it-is-not-facebook/`, last accessed: October 11th, 2010.

[Schreck et al., 2009] Schreck, T., Bernard, J., von Landesberger, T., and Kohlhammer, J. (2009). Visual cluster analysis of trajectory data with interactive Kohonen maps. *Information Visualization*, 8(1):14–29.

[Schuldt et al., 2002] Schuldt, H., Alonso, G., Beeri, C., and Schek, H.-J. (2002). Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems (TODS)*, 27(1):63–116.

[Schuler et al., 2003] Schuler, C., Weber, R., Schuldt, H., and Schek, H.-J. (2003). Peer–to–Peer Process Execution with OSIRIS. In *Proceedings of the First International Conference on Service-Oriented Computing (ICSOC 2003)*, pages 483–498, Berlin / Heidelberg. Springer.

[Schuler et al., 2004] Schuler, C., Weber, R., Schuldt, H., and Schek, H.-J. (2004). Scalable Peer–to–Peer Process Management – The OSIRIS Approach. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 26–34, Los Alamitos, California, USA. IEEE Computer Society.

[Sharples and Beale, 2003] Sharples, M. and Beale, R. (2003). A technical review of mobile computational devices. *Journal of Computer Assisted Learning*, 19(3):392–395.

[Shrivastava et al., 2011] Shrivastava, A., Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Data-driven visual similarity for cross-domain image matching. *ACM Transactions on Graphics (TOG)*, 30:154:1–154:10.

[Siggelkow, 2002] Siggelkow, S. (2002). *Feature histograms for content-based image retrieval*. PhD thesis, University of Freiburg, Germany.

[Siggelkow and Burkhardt, 2002] Siggelkow, S. and Burkhardt, H. (2002). Improvement of histogram-based image retrieval and classification. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, volume 3, pages 367–370, Los Alamitos, CA, USA. IEEE Computer Society.

[Siggelkow et al., 2001] Siggelkow, S., Schael, M., and Burkhardt, H. (2001). SIMBA - Search IMages By Appearance. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 9–16, Berlin / Heidelberg. Springer.

[Sikora, 2001] Sikora, T. (2001). The MPEG-7 visual standard for content description – an overview. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):696–702.

[Simard, 1993] Simard, P. (1993). Efficient computation of complex distance metrics using hierarchical filtering. In *Proceedings of the 7th Annual Conference on Neural Information Processing Systems (NIPS 1993)*, pages 168–175, San Francisco, CA, USA. Morgan Kaufmann.

[Simeoni et al., 2006] Simeoni, F., Azzopardi, L., and Crestani, F. (2006). An application framework for distributed information retrieval. In *Proceedings of the 9th International Conference on Asian Digital Libraries (ICADL 2006)*, pages 192–201, Berlin / Heidelberg. Springer.

[Simeoni et al., 2007a] Simeoni, F., Candela, L., Kakaletris, G., Sibeko, M., Pagano, P., Papanikos, G., Polydoras, P., Ioannidis, Y., Aarvaag, D., and Crestani, F. (2007a). A grid-based infrastructure for distributed retrieval. In *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2007)*, pages 161–173, Berlin / Heidelberg. Springer.

[Simeoni et al., 2009] Simeoni, F., Candela, L., Lievens, D., Pagano, P., and Simi, M. (2009). Functional adaptivity for digital library services in e-infrastructures: The gcube approach. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2009)*, pages 51–62, Berlin / Heidelberg. Springer.

[Simeoni et al., 2007b] Simeoni, F., Crestani, F., and Bierig, R. (2007b). The DILIGENT framework for distributed information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 781–782, New York, NY, USA. ACM.

[Simeoni et al., 2008] Simeoni, F., Yakici, M., Neely, S., and Crestani, F. (2008). Metadata harvesting for content-based distributed information retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 59(1):12–24.

[Singhal, 2011] Singhal, A. (2011). Knocking down barriers to knowledge. WWW page. `http://googleblog.blogspot.com/2011/06/knocking-down-barriers-to-knowledge.html`, last accessed: June 15, 2011.

[Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV 2003)*, pages 1470–1477, Los Alamitos, CA, USA. IEEE Computer Society.

[Smeulders et al., 2000] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(12):1349–1380.

[Smith and Kantor, 2008] Smith, C. L. and Kantor, P. B. (2008). User adaptation: good results from poor systems. In *Proceedings of the 31st Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval (SIGIR '08), pages 147–154, New York, NY, USA. ACM.

[Smith and Chang, 1996] Smith, J. R. and Chang, S.-F. (1996). VisualSEEk: a fully automated content-based image query system. In *Proceedings of the fourth ACM international conference on Multimedia (MM '96)*, pages 87–98, New York, NY, USA. ACM.

[Smucker and Jethani, 2010] Smucker, M. D. and Jethani, C. P. (2010). Human performance and retrieval precision revisited. In *Proceeding of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pages 595–602, New York, NY, USA. ACM.

[Snavely et al., 2006] Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: exploring photo collections in 3d. *ACM Transactions on Graphics (TOG)*, 25:835–846.

[Snavely et al., 2008a] Snavely, N., Seitz, S. M., and Szeliski, R. (2008a). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

[Snavely et al., 2008b] Snavely, N., Seitz, S. M., and Szeliski, R. (2008b). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.

[Snavely et al., 2010] Snavely, N., Simon, I., Goesele, M., and Seitz, S. M. (2010). Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, 98(8):1370 –1390.

[Sobel, 2010] Sobel, J. (2010). Developing facebook's new photo viewer. WWW page. `http://www.facebook.com/note.php?note_id=307069903919`, last accessed: March 29th, 2011.

[Springmann et al., 2010a] Springmann, M., Al Kabary, I., and Schuldt, H. (2010a). Experiences with QbS: Challenges and evaluation of known image search based on user-drawn sketches. CS Technical Report CS-2010-001, University of Basel, Switzerland. Available at `http://cs-wwwarchiv.cs.unibas.ch/research/publications_tec_report.html`.

[Springmann et al., 2010b] Springmann, M., Al Kabary, I., and Schuldt, H. (2010b). Image retrieval at memory's edge: Known image search based on user-drawn sketches. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, pages 1465–1468, New York, NY, USA. ACM.

[Springmann et al., 2007a] Springmann, M., Dander, A., and Schuldt, H. (2007a). Speeding up idm without degradation of retrieval quality. In *Working Notes of the CLEF Workshop 2007*.

[Springmann et al., 2008] Springmann, M., Dander, A., and Schuldt, H. (2008). Improving efficiency and effectiveness of the image distortion model. *Pattern Recognition Letters, Special Issue on Medical Image Annotation in ImageCLEF 2007*, 29(15):2018–2024.

[Springmann et al., 2007b] Springmann, M., Ispas, A., Schuldt, H., Norrie, M. C., and Signer, B. (2007b). Towards query by sketch. In *Proceedings of the Second International DELOS Conference*.

[Springmann et al., 2010c] Springmann, M., Kopp, D., and Schuldt, H. (2010c). QbS - searching for known images using user-drawn sketches. In *Proceedings of the 11th ACM SIG Multimedia International Conference on Multimedia Information Retrieval (MIR 2010)*, pages 417–420, New York, NY, USA. ACM.

[Springmann and Schuldt, 2007] Springmann, M. and Schuldt, H. (2007). Speeding up IDM without degradation of retrieval quality. In *Proceedings of the 8th Workshop of the Cross-Language Evaluation Forum (CLEF 2007)*, pages 607–614, Heidelberg / Berlin, Germany. Springer.

[Springmann and Schuldt, 2008] Springmann, M. and Schuldt, H. (2008). Using regions of interest for adaptive image retrieval. In *Proceedings of the Second International Workshop on Adaptive Information Retrieval (AIR 2008)*.

[Stonebraker et al., 2007] Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., and Helland, P. (2007). The end of an architectural era: (it's time for a complete rewrite). In *Proceedings of the 33rd international conference on Very large data bases (VLDB '07)*, pages 1150–1160. VLDB Endowment.

[Stricker and Dimai, 1996] Stricker, M. and Dimai, A. (1996). Color indexing with weak spatial constraints. In *Proceedings of the 4th SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2670, pages 29–40.

[Stricker and Dimai, 1997] Stricker, M. and Dimai, A. (1997). Spectral covariance and fuzzy regions for image indexing. *Machine Vision and Applications*, 10(2):66–73.

[Stricker and Orengo, 1995] Stricker, M. A. and Orengo, M. (1995). Similarity of color images. In *Proceedings of the 3rd SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2420, pages 381–392.

[Stricker and Swain, 1994] Stricker, M. A. and Swain, M. J. (1994). The capacity of color histogram indexing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pages 704–708, Los Alamitos, CA, USA. IEEE Computer Society.

[Studer, 2008] Studer, M. (2008). Fast scale and orientation matching of images using scale-invariant features. External MSc Thesis supervised by Marino Widmer at University of Fribourg, Switzerland, performed at University of Basel supervised by Michael Springmann and Heiko Schuldt.

[Sun Microsystems, Inc., 2003] Sun Microsystems, Inc. (2003). Java object serialization specification, revision 1.5.0. `http://java.sun.com/j2se/1.5/pdf/serial-1.5.0.pdf`.

[Sutherland, 1964] Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop (DAC '64)*, pages 6.329–6.346, New York, NY, USA. ACM.

[Swain and Ballard, 1991] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32.

[Szekely et al., 2009] Szekely, E., Bruno, E., and Marchand-Maillet, S. (2009). Collection guiding: review of the main strategies for multimedia collection browsing. Technical Report 09.01, Viper group - CVMLab, Dept of Computer Science, University of

Geneva, Route de Drize, 7, CH-1227, Carouge, Switzerland. Available at http://viper.unige.ch/doku.php/research:publi_type#technical_reports.

[Tamura et al., 1978] Tamura, H., Mori, S., and Yamawaki, T. (1978). Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6):460–473.

[Tanguay, 2011] Tanguay, D. (2011). Sort by subject in google images. WWW page. http://googleblog.blogspot.com/2011/05/sort-by-subject-in-google-images.html, last accessed: May 11, 2011.

[Tarrant et al., 2009] Tarrant, D., O'Steen, B., Brody, T., Hitchcock, S., Jefferies, N., and Carr, L. (2009). Using OAI-ORE to transform digital repositories into interoperable storage and services applications. *Code4Lib*, 6.

[Teague, 1980] Teague, M. R. (1980). Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930.

[Teevan et al., 2007] Teevan, J., Adar, E., Jones, R., and Potts, M. A. S. (2007). Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 151–158, New York, NY, USA. ACM.

[Teevan et al., 2009] Teevan, J., Cutrell, E., Fisher, D., Drucker, S. M., Ramos, G., André, P., and Hu, C. (2009). Visual snippets: summarizing web pages for search and revisitation. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '09)*, pages 2023–2032, New York, NY, USA. ACM.

[Terboven et al., 2006] Terboven, C., Deselaers, T., Bischof, C., and Ney, H. (2006). Shared-memory parallelization for content-based image retrieval. In *ECCV Workshop on Computation Intensive Methods for Computer Vision*, Graz, Austria.

[Tesic, 2005] Tesic, J. (2005). Metadata practices for consumer photos. *IEEE Multimedia*, 12:86–92.

[Teynor et al., 2006] Teynor, A., Rahtu, E., Setia, L., and Burkhardt, H. (2006). Properties of patch based approaches for the recognition of visual object classes. In *Proceedings of the 28rd DAGM-Symposium on Pattern Recognition*, pages 284–293, Berlin / Heidelberg. Springer.

[Thies et al., 2005] Thies, C., Güld, M. O., Fischer, B., and Lehmann, T. M. (2005). Content-based queries on the CasImage database within the IRMA Framework: A field report. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491/2005 of *LNCS*, pages 781–792, Berlin / Heidelberg. Springer.

[Tian et al., 2011] Tian, Q., Zhang, S., Zhou, W., Ji, R., Ni, B., and Sebe, N. (2011). Building descriptive and discriminative visual codebook for large-scale image applications. *Multimedia Tools and Applications (MTAP)*, 51(2):441–477.

[Tirilly et al., 2010] Tirilly, P., Claveau, V., and Gros, P. (2010). Distances and weighting schemes for bag of visual words image retrieval. In *Proceedings of the 11th ACM SIG Multimedia International Conference on Multimedia Information Retrieval (MIR 2010)*, pages 323–332.

[Tola et al., 2010] Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(5):815–830.

[Tommasi et al., 2008a] Tommasi, T., Orabona, F., and Caputo, B. (2008a). Discriminative cue integration for medical image annotation. *Pattern Recognition Letters, Special Issue on Medical Image Annotation in ImageCLEF 2007*, 29(15):1996 – 2002.

[Tommasi et al., 2008b] Tommasi, T., Orabona, F., and Caputo, B. (2008b). An SVM confidence-based approach to medical image annotation. In *Proceedings of the 9th Workshop of the Cross-Language Evaluation Forum (CLEF 2008), Revised Selected Papers*, pages 696–703, Berlin / Heidelberg. Springer.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1958–1970.

[Trachler, 1998] Trachler, B. (1998). *Basler Brunnen aus alter und neuer Zeit*. GS-Verlag Basel, Basel, Switzerland.

[Troncy, 2008] Troncy, R. (2008). Bringing the IPTC news architecture into the semantic web. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, pages 483–498, Berlin / Heidelberg. Springer.

[Tuite et al., 2011] Tuite, K., Snavely, N., Hsiao, D.-y., Tabing, N., and Popovic, Z. (2011). PhotoCity: training experts at large-scale image acquisition through a competitive game. In *Proceedings of the 2011 International Conference on Human Factors in Computing Systems (CHI '11)*, pages 1383–1392, New York, NY, USA. ACM.

[Turpin and Scholer, 2006] Turpin, A. and Scholer, F. (2006). User performance versus precision measures for simple search tasks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 11–18, New York, NY, USA. ACM.

[Tversky, 1977] Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

[Tyler and Teevan, 2010] Tyler, S. K. and Teevan, J. (2010). Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining (WSDM '10)*, pages 191–200, New York, NY, USA. ACM.

[Tzanetakis and Cook, 1999] Tzanetakis, G. and Cook, P. (1999). MARSYAS: a framework for audio analysis. *Organised Sound*, 4(3):169–175.

[Unay et al., 2007] Unay, D., Ekin, A., Cetin, M., Jasinschi, R., and Ercil, A. (2007). Robustness of Local Binary Patterns in brain MR image analysis. In *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2007)*, pages 2098–2101, Los Alamitos, CA, USA. IEEE Computer Society.

[Vailaya et al., 2001] Vailaya, A., Figueiredo, M. A., Jain, A. K., and Zhang, H.-J. (2001). Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1):117 –130.

[Vajgel, 2009] Vajgel, P. (2009). Needle in a haystack: efficient storage of billions of photos. WWW page. http://www.facebook.com/note.php?note_id=76191543919, last accessed: February 2nd, 2010.

[van de Sande et al., 2008] van de Sande, K. E., Gevers, T., and Snoek, C. G. (2008). A comparison of color features for visual concept classification. In *Proceedings of the 7th ACM International Conference on Image and Video Retrieval (CIVR '08)*, pages 141–150, New York, NY, USA. ACM.

[van de Sande et al., 2010] van de Sande, K. E., Gevers, T., and Snoek, C. G. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1582–1596.

[Van de Sompel et al., 2004] Van de Sompel, H., Nelson, M. L., Lagoze, C., and Warner, S. (2004). Resource harvesting within the OAI-PMH framework. *D-Lib Magazine*, 10(12).

[van den Broek et al., 2004] van den Broek, E. L., Kisters, P. M. F., and Vuurpijl, L. G. (2004). Design guidelines for a content-based image retrieval color-selection interface. In *Proceedings of the conference on Dutch directions in HCI (Dutch HCI '04)*, pages 14–17, New York, NY, USA. ACM.

[Van Gool et al., 2001] Van Gool, L., Tuytelaars, T., and Turina, A. (2001). Local features for image retrieval. In Veltkamp, R. C., Burkhardt, H., and Kriegel, H.-P., editors, *State-of-the-Art in Content-Based Image and Video Retrieval*, chapter 2, pages 21–42. Kluwer Academic Publishing, Dordrecht, The Netherlands.

[Veltkamp and Tanase, 2000] Veltkamp, R. C. and Tanase, M. (2000). Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Utrecht University. revised and extended version available at http://give-lab.cs.uu.nl/cbirsurvey/.

[Veltkamp et al., 2001] Veltkamp, R. C., Tanase, M., and Sent, D. (2001). Features in content-based image retrieval systems: A survey. In Veltkamp, R. C., Burkhardt, H., and Kriegel, H.-P., editors, *State-of-the-Art in Content-Based Image and Video Retrieval*, chapter 5, pages 97–124. Kluwer Academic Publishing, Dordrecht, The Netherlands.

[Vilain, 2006] Vilain, S. (2006). http://utsl.gen.nz/talks/git-svn/intro.html#yay-simple, last accessed: February 3rd, 2011.

[Voicu and Schuldt, 2008] Voicu, L. C. and Schuldt, H. (2008). The Re:GRIDiT protocol: Correctness of distributed concurrency control in the data grid in the presence of replication. CS Technical Report CS-2008-002, University of Basel, Switzerland. Available at http://cs-wwwarchiv.cs.unibas.ch/research/publications_tec_report.html.

[Voicu and Schuldt, 2009a] Voicu, L. C. and Schuldt, H. (2009a). How replicated data management in the cloud can benefit from a data grid protocol: the Re:GRIDiT approach. In *Proceedings of the First International CIKM Workshop on Cloud Data Management (CloudDB 2009)*, pages 45–48, New York, NY, USA. ACM.

[Voicu and Schuldt, 2009b] Voicu, L. C. and Schuldt, H. (2009b). Load-aware dynamic replication management in a data grid. In *Proceedings of the 17th International Confer-*

ence on Cooperative Information Systems (CoopIS 2009), pages 201–218, Berlin / Heidelberg. Springer.

[Voicu et al., 2009a] Voicu, L. C., Schuldt, H., Akal, F., Breitbart, Y., and Schek, H.-J. (2009a). Re:GRIDiT - coordinating distributed update transactions on replicated data in the grid. In *Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (GRID 2009)*, pages 120–129, Los Alamitos, California, USA. IEEE Computer Society.

[Voicu et al., 2009b] Voicu, L. C., Schuldt, H., Breitbart, Y., and Schek, H.-J. (2009b). Replicated data management in the grid: The Re:GRIDiT approach. In *Proceedings of the 1st ACM International Workshop on Data Grids for e-Science (DaGreS 2009)*, pages 7–16, New York, NY, USA. ACM.

[Voicu et al., 2010] Voicu, L. C., Schuldt, H., Breitbart, Y., and Schek, H.-J. (2010). Flexible data access in a cloud based on freshness requirements. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD 2010)*, pages 180–187, Los Alamitos, California, USA. IEEE Computer Society.

[von Ahn and Dabbish, 2004] von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human factors in computing systems (CHI '04)*, pages 319–326, New York, NY, USA. ACM.

[Vu et al., 2003] Vu, K., Hua, K. A., and Tavanapong, W. (2003). Image retrieval based on regions of interest. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15:1045–1049.

[Vuurpijl et al., 2002] Vuurpijl, L., Schomaker, L., and van den Broek, E. (2002). Vind(x): using the user through cooperative annotation. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR '02)*, pages 221–226.

[W3C, 2000] W3C (2000). Simple Object Access Protocol (SOAP) 1.1. `http://www.w3.org/TR/2000/NOTE-SOAP-20000508`.

[W3C, 2001] W3C (2001). Web Services Description Language (WSDL) 1.1. `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`.

[Wan and Liu, 2008] Wan, G. and Liu, Z. (2008). Content-based information retrieval and digital libraries. *Information Technology and Libraries (ITAL)*, 27(1).

[Wang et al., 2009] Wang, J., Pohlmeyer, E., Hanna, B., Jiang, Y.-G., Sajda, P., and Chang, S.-F. (2009). Brain state decoding for rapid image retrieval. In *Proceedings of the 17th ACM International Conference on Multimedia (MM'09)*, pages 945–954, New York, NY, USA. ACM.

[Wang et al., 1999] Wang, J. T.-L., Wang, X., Lin, K.-I., Shasha, D., Shapiro, B. A., and Zhang, K. (1999). Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pages 307–311, New York, NY, USA. ACM.

[Wang et al., 2001] Wang, J. Z., Li, J., and Wiederhold, G. (2001). SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23:947–963.

[Ward and Blesser, 1986] Ward, J. R. and Blesser, B. (1986). Interactive recognition of handprinted characters for computer input. *ACM SIGCHI Bulletin*, 18(1):44–57.

[Web Gallery of Art, 1996] Web Gallery of Art (1996). Web gallery of art. WWW. `http://www.wga.hu/`, last accessed: May 12th, 2011.

[Weber, 2001] Weber, R. (2001). *Similarity search in high-dimensional vector spaces*. PhD thesis, ETH Zürich, Switzerland. Diss. Techn. Wiss. ETH Zürich, Nr. 13974, 2000, DISDBIS 74.

[Weber and Böhm, 2000] Weber, R. and Böhm, K. (2000). Trading quality for time with nearest neighbor search. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology (EDBT 2000)*, pages 21–35, London, UK. Springer-Verlag.

[Weber et al., 2000a] Weber, R., Böhm, K., and Schek, H.-J. (2000a). Interactive-time similarity search for large image collections using parallel VA-Files. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2000)*, pages 83–92, Berlin / Heidelberg. Springer.

[Weber et al., 2000b] Weber, R., Böhm, K., and Schek, H.-J. (2000b). Interactive-time similarity search for large image collections using parallel VA-Files. In *Proceedings of the16th International Conference on Data Engineering (ICDE'00)*, page 197, Los Alamitos, CA, USA. IEEE Computer Society.

[Weber et al., 1999] Weber, R., Bolliger, J., Gross, T. R., and Schek, H.-J. (1999). Architecture of a networked image search and retrieval system. In *Proceedings of the 8th ACM International Conference on Information and Knowledge Management (CIKM '99)*, pages 430–441, New York, NY, USA. ACM.

[Weber and Mlivoncic, 2003] Weber, R. and Mlivoncic, M. (2003). Efficient region-based image retrieval. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM '03)*, pages 69–76. ACM Press.

[Weber and Schek, 1999] Weber, R. and Schek, H.-J. (1999). A distributed image-database architecture for efficient insertion and retrieval. In *Proceedings of the Fifth International Workshop on Multimedia Information Systems (MIS'99)*, pages 48–55.

[Weber et al., 1998] Weber, R., Schek, H.-J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of 24rd International Conference on Very Large Data Bases (VLDB '98)*, pages 194–205, San Francisco, CA, USA. Morgan Kaufmann.

[Weber et al., 2003] Weber, R., Schuler, C., Neukomm, P., Schuldt, H., and Schek, H.-J. (2003). WebService Composition with O'GRAPE and OSIRIS. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, pages 1081–1084, San Francisco, CA, USA. Morgan Kaufmann.

[Westman et al., 2008] Westman, S., Lustila, A., and Oittinen, P. (2008). Search strategies in multimodal image retrieval. In *Proceedings of the Second International Symposium on Information Interaction in Context (IIiX '08)*, pages 13–20, New York, NY, USA. ACM.

[White et al., 2007] White, S. M., Marino, D., and Feiner, S. (2007). Designing a mobile user interface for automated species identification. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI '07)*, pages 291–294, New York, NY, USA. ACM.

[Wigger, 2007] Wigger, L. (2007). Analysis and implementation of feature extractors for content-based similarity search. BSc Thesis supervised by Michael Springmann and Heiko Schuldt at University of Basel, Switzerland.

[Wikimedia Foundation, 2010] Wikimedia Foundation (2010). Statistics - wikimedia foundation. WWW page. `http://commons.wikimedia.org/wiki/Special:Statistics`, last accessed: February 2nd, 2010.

[Wilson and Elsweiler, 2010] Wilson, M. L. and Elsweiler, D. (2010). Casual-leisure searching: the exploratory search scenarios that break our current models. In *Proceedings of the 4th International Workshop on Human-Computer Interaction and Information Retrieval (HCIR 2010)*.

[Wilson, 1981] Wilson, T. D. (1981). On user studies and information needs. *Journal of Documentation*, 37(1):3–15.

[Wilson, 1997] Wilson, T. D. (1997). Information behaviour: An inter-disciplinary perspective. *Information Processing & Management*, 33(4):551–572.

[Witten and Bainbridge, 2010] Witten, I. H. and Bainbridge, D. (2010). *How to Build a Digital Library*. Elsevier Science Inc., New York, NY, USA, second edition.

[Wobbrock et al., 2009] Wobbrock, J. O., Morris, M. R., and Wilson, A. D. (2009). User-defined gestures for surface computing. In *Proceedings of the 27th International Conference on Human factors in computing systems (CHI '09)*, pages 1083–1092, New York, NY, USA. ACM.

[Woodruff et al., 2002] Woodruff, A., Rosenholtz, R., Morrison, J. B., Faulring, A., and Pirolli, P. (2002). A comparison of the use of text summaries, plain thumbnails, and enhanced thumbnails for web search tasks. *Journal of the American Society for Information Science*, 53:172–185.

[Wright, 2011] Wright, J. (2011). Search by text, voice, or image. WWW page. `http://insidesearch.blogspot.com/2011/06/search-by-text-voice-or-image.html`, last accessed: June 15, 2011.

[Wu and Nevatia, 2005] Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of Edgelet part detectors. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV 2005)*, volume 1.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8, Washington, DC, USA. IEEE Computer Society.

[Xie, 1997] Xie, H. I. (1997). Planned and situated aspects in interactive IR: Patterns of user interactive intentions and information seeking strategies. In *Proceedings of the 60th ASIS Annual Meeting*, volume 34, pages 101–110.

[Xie, 2000] Xie, H. I. (2000). Shifts of interactive intentions and information-seeking strategies in interactive information retrieval. *Journal of the American Society for Information Science*, 51:841–857.

[Xue et al., 2008] Xue, X.-B., Zhou, Z.-H., and Zhang, Z. M. (2008). Improving web search using image snippets. *ACM Transactions on Internet Technology (TOIT)*, 8:21:1–21:28.

[Yamada et al., 2000] Yamada, A., Pickering, M., Jeannin, S., Cieplinski, L., and Ohm, J. (2000). MPEG-7 Visual Part of Experimentation Model Version 8.0 NOTE: 9.0. *ISO/IEC JTC1/SC29/WG11/N3673*, pages 1–82.

[Yang et al., 2007] Yang, J., Jiang, Y.-G., Hauptmann, A. G., and Ngo, C.-W. (2007). Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the 9th ACM SIG Multimedia International Workshop on Multimedia Information Retrieval (MIR '07)*, pages 197–206, New York, NY, USA. ACM.

[Yang et al., 2008] Yang, L., Jin, R., Sukthankar, R., and Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*.

[Yee et al., 2003] Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. (2003). Faceted metadata for image search and browsing. In *Proceedings of the ACM SIGCHI Conference on Human factors in Computing Systems (CHI '03)*, pages 401–408, New York, NY, USA. ACM.

[Yeh et al., 2004] Yeh, T., Tollmar, K., and Darrell, T. (2004). Searching the web with mobile images for location recognition. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 76–81, Los Alamitos, California, USA. IEEE Computer Society.

[Yi and Faloutsos, 2000] Yi, B.-K. and Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pages 385–394, San Francisco, CA, USA. Morgan Kaufmann.

[Zadeh, 1965] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338 – 353.

[Zhang and Lu, 2002] Zhang, D. and Lu, G. (2002). Shape-based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848.

[Zhang, 2008] Zhang, J. (2008). *Visualization for information Retrieval*. Springer, Berlin / Heidelberg.

[Zhang et al., 2007] Zhang, L., Li, S. Z., Yuan, X., and Xiang, S. (2007). Real-time object classification in video surveillance based on appearance learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.

[Zhang and Kosecka, 2006] Zhang, W. and Kosecka, J. (2006). Image based localization in urban environments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 33–40.

[Zhou and Huang, 2003] Zhou, X. S. and Huang, T. S. (2003). Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8:536–544. 10.1007/s00530-002-0070-3.

Acknowledgment of Prior Work

"Standing on the shoulders of giants" is a frequently used proverb when it comes to how research is performed. This thesis is no exception and probably even an extreme case as much of its contribution is achieved by integrating findings from previous works into combined models and implementations. All literature that was used has been mentioned in the bibliography and this list has grown very, very long over the years. Therefore I want to highlight some works and people that have been particular influential for this thesis.

For the Image Task Model (ITM) introduced in Chapter 2 the most important works have been [Xie, 1997, Xie, 2000] that opened the eyes to the difference between the *interaction intentions* that a user has and the *information seeking strategies* that the user applies to achieve those intentions. This provided the possibility to view in a new light to the frequently used (or –unfortunately– as frequently ignored) classification of image search tasks in [Cox et al., 2000] and [Smeulders et al., 2000] (which is based a lot on the former) and revealed, that some of their definitions were not ideal exactly because they lacked the separation between *interaction strategies* and *seeking strategies*. Other important works to base the ITM on have also been [Armitage and Enser, 1997, Enser, 2008] and [Fidel, 1997].

The idea to come up with reusable building blocks has always been around in my research group at UMIT, Austria under the guidance of Prof. Hans-Jörg Schek and in Basel, Switzerland under the guidance of Prof. Dr. Heiko Schuldt. Part II and Part III can therefore be seen as a fairly natural extension of the idea to separate the purely conceptual view from any particular implementation to improve reusability beyond the borders of issues of compatibility at the level of bits and bytes – although fortunately these issues have been significantly reduced thanks to a growing interest in interoperability on web-scale. The development of the Generic Storage Model presented in Chapter 4.1.1 is strongly related to the evolution of the DILIGENT Storage Model, for which [Candela et al., 2007, pp. 67f] was certainly not the starting point. More important were certainly the discussions in the earlier phases of the DILIGENT project, in which core contributions were given by Prof. Hans-Jörg Schek and Sören Balko, but also fruitful discussions with Fuat Akal w.r.t. replication and Peter Fankhauser and Bhaskar Metha of Fraunhofer Gesellschaft-IPSI w.r.t. integration of metadata.

For Chapter 5, [Smeulders et al., 2000] and [Jörgensen, 2003] were very helpful in not getting completely lost in the details of the perceptual features. For both, the overview on query execution in Chapter 5 as well as the implementation in Chapter 10, the possibility to benefit of the knowledge of Roger Weber through his lectures, personal discussions, and reading his thesis [Weber, 2001] as well as his source code were invaluable. As mentioned in Chapter 10.5.2, the Early Termination Strategy can already be found in his source code dating back to the late 90's. What was new, however, was the impact of the strategy to a very expensive distance measure such as the Image Distortion Model (IDM) which led to the extensive study of the Early Termination Strategy in [Springmann et al., 2008].

The implementation of applications described in Chapter 10, Chapter 11, and Chapter 14 would not have been possible without the starting point of ISIS with main contributions from Roger Weber [Weber, 2001] and Michael Mlivoncic [Mlivoncic, 2006] and the underlying OSIRIS infrastructure [Brettlecker et al., 2007], but also the novel contributions from the students I had the honor to work with: [Dander, 2006, Wigger, 2007, Studer, 2008, Ruske, 2008, Morel and Schurter, 2009, Kopp, 2009, Giangreco, 2010, Kreuzer, 2010]. Another very important contribution was Ihab Al Kabary's thorough assessment of the quality of QbS, of which his dedication and patience in generating and digging through log files led to the results that materialized in Chapter 10.3.5.

Photo Credits

Chapter 1

Figure 1.1: DSC01269 taken on March, 3 2009 at Petersplatz, Basel by Michael Springmann

Figure 1.2: Screenshot of `http://images.google.com`, taken on October 13, 2010

Figure 1.3: Screenshot of `http://www.flickr.com/photos/9655482@N08/2312129124/`, taken on October 13, 2010

Figure 1.4 and 1.5: Screenshots of `http://brunnenfuehrer.ch/`, taken on October 13, 2010

Figure 1.6: Screenshot of generated result page with code that was part of [Wigger, 2007] and extended by Michael Springmann, taken on June, 19 2009 with images crawled from `http://brunnenfuehrer.ch/`

Figure 1.7: DSCN3089 and DSCN3085 taken February, 15 2007 in Tirrenia/Pisa, Italy by Michael Springmann; Person in picture: P. Ranaldi

Figure 1.8: DSC08146 and DSC08380 on October 9, 2009 at Alte Kirche, Fautenbach/Achern, Germany by Michael Springmann

Figure 1.9: Screenshots of `http://en.wikipedia.org/w/index.php?title=Opel_Kapit.C3.A4n&oldid=361525224`, taken on January, 27 2011. *Figure 1.9(a):* Contains "Opel Kapitän 1959 03.jpg" by Späth Chr./ChiemseeMan, available as Public Domain at `http://en.wikipedia.org/wiki/File:Opel_Kapit%E4n_1959_03.jpg` ⓟⓓ *Figure 1.9(b):* Contains "Opel Kapitän B BW 1.JPG" by Berthold Werner, available as Public Domain at `http://en.wikipedia.org/wiki/File:Opel_Kapit%E4n_B_BW_1.JPG` ⓟⓓ

Chapter 2

Figure 2.1: "Mini Collection" consisting of DSC03682, DSCN1728, DSCN0010, DSCN7797, DSCN1067, DSC04879, DSC08579, DSC01932, DSC00912, DSCN8121 taken 2003–2011 by Michael Springmann

Figure 2.2: Illustration done by Michael Springmann, contains images of Figure 2.1 and DSC01931 as query image, taken by Michael Springmann

Figure 2.3: DSC06861and DSC09687 taken by Michael Springmann, see Figure 2.9 for details; Figure 2.3 illustration Copyright © Anoto AB (`http://www.anoto.com`), licensed under Creative Commons-Attribution Non-Commercial No-Derivative Works available at `http://www.flickr.com/photos/anotogroup/3465589846/` ⓒⓘⓢⓔ.

Figure 2.4 – 2.7: DSC08910 taken on December, 16 2009 near Mittlere Rheinbrücke in Basel, Switzerland by Michael Springmann; image editing performed using Pixelmator

Figure 2.8: DSC08910 and DSC08908 taken on December, 16 2009 near Mittlere Rheinbrücke in Basel, Switzerland by Michael Springmann; DSCN2536 and DSCN2540 taken on October, 25 2003 in Innsbruck, Austria by Michael Springmann

Figure 2.9: DSC06861 taken on December, 14 2010 and DSC09700 taken on March, 23 2010 by Michael Springmann; Hand in picture belongs to I. Al Kabary, application shown on Tablet PC screen has been developed in the context of [Kopp, 2009, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary, interactive paper designed in context of [Kreuzer, 2010] and [Agosti et al., 2007].

Figure 2.10: DSC08146 see Figure 1.8, DSC01269 see Figure 1.1

Figure 2.11: See Figure 1.6 and Figure 1.5

Figure 2.12: See Figure 1.8 and see Figure 1.9

Figure 2.13 – 2.16: Illustration done by Michael Springmann

Chapter 3

Figure 3.1: Illustration done by Michael Springmann, contains images of Figure 2.1 and DSC01931 as query image, taken by Michael Springmann

Chapter 4

Figure 4.1: Illustration done by Michael Springmann

Figure 4.2 – 4.5: UML Diagrams done by Michael Springmann using TopCoder UML Tool

Chapter 5

Figure 5.1: Illustration done by Michael Springmann

Figure 5.2: DSC01932 see Figure 2.1, segments generated using the "Smart Lasso" tool in Apple Preview on Mac OS X 10.6

Figure 5.3: Keypoint illustrations generated with code that was part of [Studer, 2008] and extended by Michael Springmann, source images DCS01931 and DSC01932 taken on April, 13 2009 near Achern, Germany by Michael Springmann

Figure 5.4: Illustrations done by Michael Springmann, face detection and labeling performed using Apple Aperture 3.1; source image DSC00912 in Figure 5.4(a) taken on March, 27 2009 in Philadelphia, USA by Michael Springmann, source image DSCN3089 see Figure 1.7, source image DSC01034 in Figure 5.4(c) taken on March, 15 2011 in Basel, Switzerland by Michael Springmann

Figure 5.5 and 5.6: Region illustrations generated with code that was part of [Kopp, 2009] and extended by Michael Springmann, source images DCS01931 and DSC01932 taken on April, 13 2009 near Achern, Germany by Michael Springmann

Figure 5.7: Illustration done by Michael Springmann, strongly inspired by [Dimai, 1999b, p. 13] and [Popper, 1959, p. 412]

Figure 5.8: Illustration redrawn using Pixelmator based on illustration in [Bober, 2001, p. 716]

Figure 5.9: Edge illustrations generated with code that was part of [Kopp, 2009] and extended by Michael Springmann, source image DSC01932 taken on April, 13 2009 near Achern, Germany by Michael Springmann

Figure 5.10: Images taken from the query images of the ImageCLEF 2005 Medical Image Retrieval Benchmark Task [Clough et al., 2005, Hersh et al., 2006]: Figure 5.10(a)

uses Image2_1.jpg, Figure 5.10(b) uses Image10.jpg, Figure 5.10(c) uses Image11.jpg, Figure 5.10(d) uses Image5.jpg, and Figure 5.10(e) uses Image14_1.jpg.

Figure 5.11 and 5.12: Distance illustrations generated with code developed by Michael Springmann, inspired by the illustrations used in [Weber, 2001]

Figure 5.13: Illustration done by Michael Springmann

Figure 5.14: Illustrations generated with code that was part of [Kopp, 2009] and extended by Michael Springmann, source image DSC01932 taken on April, 13 2009 near Achern, Germany by Michael Springmann

Figure 5.15 – 5.19: Distance illustrations generated with code developed by Michael Springmann, inspired by the illustrations used in [Weber, 2001]

Chapter 6

Figure 6.1: Illustration done by Michael Springmann

Figure 6.2 – 6.4: Screenshots taken on March 6 to 15, 2011: Figure 6.2(a) and 6.2(b) show content of `http://images.google.com/`; Figure 6.2(c) and 6.2(d) shows `http://www.flickr.com/photos/laugis_photo/2312129124/` containing picture taken by "photo-maker" on November 27, 2007; Figure 6.2(e) and 6.3(b) shows content of `http://www.corbisimages.com`; Figure 6.2(f) shows content of `http://www.bing.com/images`; Figure 6.3(a) shows content of `http://www.flickr.com/explore/`; Figure 6.4(a) shows content of `http://www.brunnenfuehrer.ch/home.htm`; Figure 6.4(b) shows content of `http://alipr.com`; Figure 6.4(c) and 6.4(d) show content of `http://labs.systemone.at/retrievr/`.

Figure 6.5: DSC06861 taken on December, 14 2010, DSC09687 and DSC09700 taken on March, 23 2010 by Michael Springmann; Hand in picture belongs to I. Al Kabary, application shown on Tablet PC screen has been developed in the context of [Kopp, 2009, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary, interactive paper designed in context of [Kreuzer, 2010] and [Agosti et al., 2007].

Figure 6.6: Screenshots of `http://images.google.com` taken on May 13, 2011

Figure 6.7 – 6.11: Screenshots of DelosDLMS [Agosti et al., 2007] taken on May 10 to 13, 2011

Figure 6.12: Screenshot of `http://image-swirl.googlelabs.com` taken on May 11, 2011

Figure 6.13: Screenshots of `http://www.wga.hu` taken on May 12, 2011

Figure 6.14: Screenshot of MedioVis as used in DelosDLMS [Agosti et al., 2007] taken on May 10, 2011

Figure 6.15: Screenshot of DARE as used in DelosDLMS [Agosti et al., 2007] taken on May 10, 2011

Figure 6.16: Illustration done by Michael Springmann, contains images of Figure 2.1 and DSC01931 as query image, taken by Michael Springmann

Figure 6.17 – 6.20: Distance illustrations generated with code developed by Michael Springmann, inspired by the illustrations used in [Weber, 2001]

Figure 6.21: Screenshots of QbS Prototype developed in the context of [Kopp, 2009, Giangreco, 2010, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al

Kabary; images in example have been crawled from the Web Gallery of Art `http://www.wga.hu` to show similar content as previous examples.

Chapter 10

Figure 10.1: See Figure 5.10

Figure 10.2: Images taken from the ImageCLEF 2007 Medical Annotation Benchmark Task [Deselaers et al., 2008a]: Figure 10.2(a) uses 2157.png, Figure 10.2(b) uses 18304.png, Figure 10.2(c) uses 18305.png (all three part of the *Training* set), Figure 10.2(d) uses 373432.png (part of the *Development* set).

Figure 10.4 and 10.5: Original images from Figure 10.2 processed with filter code developed by Michael Springmann with parameters related to `http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm`.

Figure 10.6 – 10.14: Illustrations done by Michael Springmann

Figure 10.15(a), 10.19(a), 10.21(a): 'Supermarine Seafire MKXVII' by Alex Layzell, License: CC BY NC ND; im1660.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – usage explicitly granted by copyright holder.

Figure 10.15(b), 10.19(b), 10.22(a): 'Perast - Montenegro' by Milachich, License: CC BY NC; im10853.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.15(c), 10.16(a), 10.17(a), 10.19(c), 10.23(a): 'Version 1.0 release candidate 1' by Petteri Sulonen, License: CC BY; im18707.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.16(b): same image, manually traced into segments with a path/Bézier curve tool in Apple Keynote '09 (Version 5.1) by Michael Springmann. *Figure 10.16(c):* sketch drawn by Michael Springmann. *Figure 10.16(d):* Screenshot of QbS [Springmann et al., 2010c] taken on September 28, 2011 using IDM as a feature with warp range of 2 and a local context of 2. *Figure 10.17(b) – 10.17(f), Figure 10.18:* Edge illustrations generated from same image with code that was part of [Kopp, 2009] and extended by Michael Springmann.

Figure 10.20: DSC09729 taken on March, 23 2010 by Michael Springmann; QbS application shown on Tablet PC screen has been developed in the context of [Kopp, 2009, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary

Figure 10.15(d), 10.19(d), 10.24(a): 'If Eiffel In Love With You' by Gideon, License: CC BY; im18797.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.21(b) – 10.21(p), Figure 10.22(b) – 10.22(o), Figure 10.23(b) – 10.23(p), Figure 10.24(b) – 10.24(k): Sketches drawn by Ihab Al Kabary, Nadine Fröhlich, Diego Milano, Thorsten Möller, Heiko Schuldt, Nenad Stojnić and Michael Springmann

Figure 10.25 and 10.35: Illustrations done by Michael Springmann based on illustration by Ihab Al Kabary published in [Springmann et al., 2010a]

Figure 10.26: Screenshots of QbS Prototype developed in the context of [Kopp, 2009, Giangreco, 2010, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary; im1660.jpg and other images in example are part of MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008].

Figure 10.27(a), 10.29(a): 'The Head On The Floor' by Zoe Chuang (ZORRO), License: CC BY NC SA; im972.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – not included in online version due to conflicting license.

Figure 10.27(b), 10.30(a): 'huckleberries' by julie (hello-julie), License: CC BY NC ND; im17919.jpg in the MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – not included in online version due to conflicting license.

Figure 10.27(c), 10.31(a): 'Kue Mangkok' by Riana Ambarsari (p3nnylan3), License: CC BY NC ND; im5820.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – usage explicitly granted by copyright holder.

Figure 10.27(d), 10.32(a): 'Happier than happy square' by Carina Envoldsen-Harris, License: CC BY NC ND; im2267.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – not included in online version due to conflicting license.

Figure 10.27(e), 10.33(a): 'Imperial Savoy' by John Kratz, License: CC BY SA; im10829.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.28: Illustrations generated with code that was part of [Giangreco, 2010]

Figure 10.29 – 10.33: Sketches drawn by Ivan Giangreco and his study colleagues, friends, and relatives.

Figure 10.36(a): 'Pontoon B&W' by Steve Bailey, License: CC BY NC ND; im12023.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – not included in online version due to conflicting license. *Figure 10.36(b):* sketch drawn by Roman Kreuzer.

Figure 10.37 – 10.43: Sketches drawn by Ihab Al Kabary, Nadine Fröhlich, Roman Kreuzer, Christoph Langguth, Diego Milano, Thorsten Möller, Heiko Schuldt, Nenad Stojnić and Michael Springmann

Figure 10.37: Illustration taken from [Kreuzer, 2010, p. 73]. Contains 'Tourists Often Say The Cutest Things Very Loudly. That's Why Virginia Beach Needs These Signs' by Bill Barber, License: CC BY NC; im1561.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.38: Illustration taken from [Kreuzer, 2010, p. 74]. Contains 'Japanese green tea' by E v a, License: CC BY NC SA; im4595.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – not included in online version due to conflicting license.

Figure 10.39: Illustration taken from [Kreuzer, 2010, p. 75]. Contains 'Proud Bird' by Daniel Greene, License: CC BY NC ND; im7799.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – usage explicitly granted by copyright holder.

Figure 10.40: Illustration taken from [Kreuzer, 2010, p. 76]. Contains 'winking pumpkin' by Philip Hay (minipixel), License: CC BY NC ND; im11541.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.41: Illustration taken from [Kreuzer, 2010, p. 77]. Contains 'L'avventura' by Jenna (ici et ailleurs), License: CC BY NC ND; im18791.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008] – usage explicitly granted by copyright holder.

Figure 10.42: Illustration taken from [Kreuzer, 2010, p. 78]. Contains 'Erin [118/365]' by Joe Plocki (turbojoe), License: CC BY NC SA; im15816.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.43(a): Screenshots of QbS Prototype developed in the context of [Kopp, 2009, Giangreco, 2010, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary; search based on [Kreuzer, 2010, p. 79]. *Figure 10.43(c):* 'Pitchers and catchers!' by Meg Hourihan (megnut), License: CC BY NC SA; im6140.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]

Figure 10.44: Illustration done by Djexplo, released as public domain (License: CC 0, available at `http://commons.wikimedia.org/wiki/File:Latitude_and_Longitude_of_the_Earth.svg`).

Figure 10.45: Images of Apple iPhone 3GS running iOS 5 and showing Maps app (DSC04562) as well as the Garafa GPS Kit app (`http://gpskit.garafa.com/GPSKit`, DSC04576) and images of Sony GPS-CS3 (DSC04569, DSC04571) taken on November, 11 2011 in Schlieren, Switzerland by Michael Springmann

Figure 14.13 – 10.70: Query images and images in the dataset of geotagged images taken in Basel, Switzerland and Innsbruck, Austria in the years 2008 and 2009 by Michael Springmann

Figure 14.13 – 10.51, 10.57 – 10.70: Screenshots of RetroGeotag Prototype developed in the context of [Morel and Schurter, 2009] and extended by Michael Springmann.

Figure 10.56, 10.48 – 10.50, 10.58: (Cropped) screenshots use map information from `http://www.openstreetmap.org/` accessed through JXMapViewer inside the code that is used in the RetroGeotag Prototype developed in the context [Morel and Schurter, 2009] and extended by Michael Springmann.

Figure 10.52, 10.53(b), 10.54(b), 10.54(d), 10.55: Illustrations of keypoints and matches generated with code that was part of [Studer, 2008] and extended by Michael Springmann

Figure 10.64 – 10.70: Reference images in database crawled from the website `http://www.brunnenfuehrer.ch/` on November 7, 2011. Website and images by Pascal Hess and Martin Stauffiger.

Chapter 10

Figure 10.71 – 10.73: Illustrations done by Michael Springmann, contains images of Figure 2.1 and DSC01931 as query image, taken by Michael Springmann

Figure 10.74 – 10.75: Illustrations done by Michael Springmann, originally published in [Springmann et al., 2008]

Figure 10.76 – 10.79: DSCN1726 and DSCN1728 taken on November, 4 2006 in Basel, Switzerland by Michael Springmann; rough segments manually selected using a path tool; Keypoint illustrations and image transformations generated with code that was part of [Studer, 2008] and extended by Michael Springmann

Chapter 11

Figure 11.1: Screenshots of prototype applications taken on December 19, 2011. Medical Image Classification GUI in (a) developed by Michael Springmann; query

image belongs to collection used for ImageCLEF 2007 Medical Annotation Benchmark Task [Deselaers et al., 2008a]. QbS Prototype in (b) developed in the context of [Kopp, 2009, Giangreco, 2010, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary. RetroGeotag Prototype developed in the context of [Morel and Schurter, 2009] and extended by Michael Springmann.

Figure 11.2 and Figure 11.3: DSC09686 and DSC09729 taken on March, 23 2010 by Michael Springmann; Hand in picture belongs to I. Al Kabary, application shown on Tablet PC screen has been developed in the context of [Kopp, 2009, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary. Images in search results belong to MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008].

Figure 11.4 and 11.5 Illustration taken from [Giangreco, 2010, pp. 17f]. QbS Prototype in 11.4 developed in the context of [Kopp, 2009, Giangreco, 2010]. Images in search results belong to MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008].

Figure 11.6 – 11.12 Screenshots of QbS Prototype developed in the context of [Kopp, 2009, Giangreco, 2010, Kreuzer, 2010] and extended by Michael Springmann and Ihab Al Kabary. Image 'Perast - Montenegro' by Milachich, License: CC BY NC SA; im10853.jpg in MIRFLICKR-25000 image retrieval benchmark collection [Huiskes and Lew, 2008]. Other images from this collection shown in search results.

Figure 11.13: DSC06861 taken by Michael Springmann, see Figure 2.9 for details; Figure 11.13(b) illustration Copyright © Anoto AB (`http://www.anoto.com`), licensed under Creative Commons-Attribution Non-Commercial No-Derivative Works available at `http://www.flickr.com/photos/anotogroup/3465589846/` CC BY NC ND. *Figure 11.14:* Illustration based on [Kreuzer, 2010, p. 35], restructured by Ihab Al Kabary for [Kreuzer et al., 2012].

Figure 11.15: Paper Interface of QbS Prototype for the MIRFLICKR-25000 collection developed in [Kreuzer, 2010]. Scan of interface performed by Ihab Al Kabary for publication in [Kreuzer et al., 2012].

Chapter 13

Figure 13.1: Illustrations done by Michael Springmann, contains images of Figure 2.1 and DSC01931 as query image, taken by Michael Springmann. Includes also DSC04944 of Yorkvill Public Library in Toronto, ON, Canada taken on October, 26 2010 by Michael Springmann, DSC01034 as in Figure 5.4(c), DSC04948 and IMG0020 as in Figure 10.55 and DSC06861 as in Figure 2.9; as well as company / product logos of Adobe Inc., Apple Inc., Google Picasa, Yahoo! flickr, Facebook, Twitter.

Figure 13.2: Illustrations done by Michael Springmann

Chapter 14

Figure 14.1: Illustrations done by Michael Springmann

Figure 14.2 Illustrations done by Heiko Schuldt, taken from [Schek and Schuldt, 2008].

Figure 14.3 and 14.4: Screenshots of ISIS / DelosDLMS [Brettlecker et al., 2007, Agosti et al., 2007] and O'GRAPE [Weber et al., 2003] taken by Michael Springmann on January, 20 2006.

Figure 14.5: Illustrations taken from [Agosti et al., 2007].
Figure 14.6 Illustrations done by Heiko Schuldt, taken from [Schek and Schuldt, 2008].
Figure 14.7 and 14.8: Illustration taken from [Candela et al., 2007].
Figure 14.9: Screenshots of Medical Image Classification GUI developed by Michael Springmann taken on January 4, 2012. Medical image taken from the ImageCLEF 2007 Medical Annotation Benchmark Task [Deselaers et al., 2008a].
14.10 – 14.12: Illustrations done by Michael Springmann

Curriculum Vitae

Michael Springmann

January 9, 1979	Born in Achern, Deutschland Son of Heidi and Thomas Springmann Citizen of the Federal Republic of Germany
1985–1989	Primary School, Oberachern, Germany
1989–1998	Gymnasium, Achern, Germany
1998–2001	Study of Business Information Systems Baden-Wuerttemberg Cooperative State University, Karlsruhe, Germany degree: Dipl.-Wirtschaftsinformatiker (DH) Bachelor of Arts
2001–2003	Study of Computer Science and Multimedia University of Applied Sciences, Karlsruhe, Germany degree: Master of Science in Computer Science
2002–2003	Study of Computer Science Eastern Michigan University, Ypsilanti, Michigan, USA degree: Master of Science in Computer Science
2003–2006	Research assistant in the group of Prof. H.-J. Schek Institute for Information Systems, UMIT Hall in Tyrol
2006–2011	Research assistant in the group of Prof. Dr. Heiko Schuldt Database and Information Systems, University of Basel