A Global-to-Local Model for the Representation of Human Faces

Inauguraldissertation

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

von

Reinhard Knothe
aus Freiburg im Breisgau, Deutschland

Basel, 2009

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät

auf Antrag von

Prof. Dr. Thomas Vetter, Universität Basel, Dissertationsleiter
Prof. Dr. Hans Burkhardt, Universität Freiburg, Korreferent

Basel, den 23.06.2009

Prof. Dr. Eberhard Parlow, Dekan

A Global-to-Local Model

for the Representation of Human Faces

PhD Thesis

Reinhard Knothe

2009

Originaldokument gespeichert auf dem Dokumentenserver der Universität Basel
edoc.unibas.ch

Dieses Werk ist unter dem Vertrag „Creative Commons Namensnennung-Keine kommerzielle Nutzung-Keine Bearbeitung 2.5 Schweiz" lizenziert. Die vollständige Lizenz kann unter
creativecommons.org/licences/by-nc-nd/2.5/ch
eingesehen werden.

Namensnennung-Keine kommerzielle Nutzung-Keine Bearbeitung 2.5 Schweiz

Sie dürfen:

das Werk vervielfältigen, verbreiten und öffentlich zugänglich machen

Zu den folgenden Bedingungen:

Namensnennung. Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen (wodurch aber nicht der Eindruck entstehen darf, Sie oder die Nutzung des Werkes durch Sie würden entlohnt).

Keine kommerzielle Nutzung. Dieses Werk darf nicht für kommerzielle Zwecke verwendet werden.

Keine Bearbeitung. Dieses Werk darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter welche dieses Werk fällt, mitteilen.
- Jede der vorgenannten Bedingungen kann aufgehoben werden, sofern Sie die Einwilligung des Rechteinhabers dazu erhalten.
- Diese Lizenz lässt die Urheberpersönlichkeitsrechte unberührt.

Abstract

In the context of face modeling and face recognition, statistical models are widely used for the representation and modeling of surfaces. Most of these models are obtained by computing Principal Components Analysis (PCA) on a set of representative examples. These models represent novel faces poorly due to their holistic nature (i.e. each component has global support), and they suffer from overfitting when used for generalization from partial information. In this work, we present a novel analysis method that breaks the objects up into modes based on spatial frequency. The high-frequency modes are segmented into regions with respect to specific features of the object. After computing PCA on these segments individually, a hierarchy of global and local components gradually decreasing in size of their support is combined into a linear statistical model, hence the name, Global-to-Local model (G2L). We apply our methodology to build a novel G2L model of 3D shapes of human heads. Both the representation and the generalization capabilities of the models are evaluated and compared in a standardized test, and it is demonstrated that the G2L model performs better compared to traditional holistic PCA models. Furthermore, both models are used to reconstruct the 3D shape of faces from a single photograph. A novel adaptive fitting method is presented that estimates the model parameters using a multi-resolution approach. The model is first fitted to contours extracted from the image. In a second stage, the contours are kept fixed and the remaining flexibility of the model is fitted to the input image. This makes the method fast (30 sec on a standard PC), efficient, and accurate.

Contents

Notation 10

1 Introduction 11
1.1 Prior Work 12
1.2 The Global-to-Local Model 15
1.3 Overview 15

2 Data and Parametrization 17
2.1 The Data 17
  2.1.1 MPI-TÜ Data 17
  2.1.2 UniBS Data 18
2.2 Scanner and Data Acquisition 20
2.3 Parametrization and Correspondence 21
2.4 The Template 22
  2.4.1 Parametrization 23
  2.4.2 Data Representation 24

3 The Global-to-Local Model 27
3.1 PCA Model 28
3.2 Global-to-Local Model 32
3.3 Multiresolution Analysis 34
  3.3.1 Gaussian- and Laplacian Pyramids 34
  3.3.2 Gaussian- and Laplacian Pyramids for Shapes 35
  3.3.3 Filterbank 38
3.4 Segmentation of the Details 39
3.5 Building a linear statistical model 41
  3.5.1 Global and Local Components 42
  3.5.2 Covariance 44
3.6 Regularization 46

4 Evaluation 49
- 4.1 Distance Measures . . . 50
- 4.2 Projection to the Face Space . . . 51
- 4.3 Reconstruction from a Sparse Set of Points . . . 52
- 4.4 Experiments . . . 53
  - 4.4.1 Experiment A . . . 53
  - 4.4.2 Experiment B . . . 58

5 Fitting 61
- 5.1 External Models . . . 62
  - 5.1.1 Shape and Color Models . . . 63
  - 5.1.2 Camera Model . . . 63
  - 5.1.3 Illumination Model . . . 66
- 5.2 Cost Function . . . 67
  - 5.2.1 Special Cost Functions . . . 69
  - 5.2.2 Regularization . . . 71
- 5.3 Optimization . . . 71
- 5.4 Stepwise Fitting System . . . 73
  - 5.4.1 Multiresolution . . . 73
  - 5.4.2 Multi-Level Fitting . . . 73
  - 5.4.3 2-Stage Fitting . . . 74
- 5.5 SGD and MF Fitter . . . 75
  - 5.5.1 SGD Fitter . . . 76
  - 5.5.2 Multi Feature Fitter . . . 77
- 5.6 Fitting Modules . . . 78
  - 5.6.1 Module 1: Pose from Landmarks . . . 78
  - 5.6.2 Module 2: Shading from Contour . . . 79
  - 5.6.3 Module 3: Illumination from Intensity . . . 84
  - 5.6.4 Module 4: Surface Color from Pixel Intensity . . . 85
  - 5.6.5 Module 5: Shape from Shading . . . 86
  - 5.6.6 Texture Extraction . . . 88
- 5.7 Efficiency . . . 90
  - 5.7.1 Multiresolution, Optimization and Cost Function . . . 91
  - 5.7.2 Implementation . . . 92
  - 5.7.3 Elimination of irrelevant terms of ∇ . . . 92
- 5.8 Results and Comparison . . . 93
  - 5.8.1 Experiment C . . . 93
  - 5.8.2 Experiment D . . . 94

6 **Conclusion and Future Work** **99**
6.1 Conclusion 99
6.2 Expression Model 100
6.3 Automatic Contour Extraction 102
6.4 Statistical Color Models 104
6.5 Color Correction 105

7 **Appendix** **109**
7.1 Derivatives 109
7.1.1 Derivatives of the Camera Model 109
7.1.2 Derivatives of the Shape Model 110
7.1.3 Derivatives of the Illumination Model 111
7.1.4 Derivatives of the Surface Color Model 112
7.1.5 Derivatives of the Surface Normals 113
7.1.6 Derivatives of the Shape Model (Color Intensity) 114
7.2 Computation of the Segments 115
7.3 Reconstruction and Blending of the Texture 117

Acknowledgments 118
Bibliography 119
Curriculum Vitae 127

Notation

n	the number of training examples
m	the number of vertices in each example
$\mathbf{x} = (\vec{x}_1^T, \ldots, \vec{x}_m^T)^T \in \mathbb{R}^{3m}$	3D shape of one training example
$\vec{x} = (x_x, x_y, x_z)^T$	one vertex (in model coordinates)
$\vec{e} = (e_x, e_y, e_z)^T$	one vertex (in eye coordinates)
$\vec{w} = (w_x, w_y)^T$	one vertex (in window coordinates)
$\mathbf{x}^l$	3D shape of example on level l
$\mathbf{y}^l$	details on level l
$\downarrow_l^h$	Fine-to Coarse Transition from level h (high) to level l (low)
$\uparrow_l^h$	Coarse-to Fine Transition from level l (low) to level h (high)
ℓ	level of the G2L model
$\mathcal{M} := (\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{U})$	linear generative statistical model
$\boldsymbol{\mu} \in \mathbb{R}^{3m}$	mean
$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n'}] \in \mathbb{R}^{3m \times n'}$	matrix with n' components
$\boldsymbol{\sigma} = (\sigma_1, \ldots \sigma_{n'}) \in \mathbb{R}^{n'}$	weighting of the components
$\mathcal{M}_{\text{PCA}}$	PCA model
$\mathcal{M}_{\text{G2L}}$	Global-to-Local model
$\mathbf{a} = (\vec{a}_1^T, \ldots, \vec{a}_m^T)^T \in \mathbb{R}^{3m}$	per-vertex-color of one training example
$\vec{a} = (a^R, a^G, a^B)^T$	color of one vertex
$\mathbf{p} = (\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta})$	model parameters
$\boldsymbol{\rho}$	camera model parameters (see Table 5.1)
$\boldsymbol{\lambda}$	illumination model parameters (Table 5.2)
$\boldsymbol{\alpha}$	shape model parameters
$\boldsymbol{\beta}$	surface color model parameters

Chapter 1

Introduction

Reconstructing the 3D shape of a human face from a photo is an ill-posed problem and therefore requires prior knowledge, which is usually introduced in the form of a statistical model. Statistical shape models represent object classes by parameters describing the variability of the elements within the class. In contrast to models that are purely descriptive, statistical models are generative, i.e. the shape is a function of the parameters. They are widely used in computer vision and computer graphics for the representation of human heads [1, 2, 3, 4, 5]. Similar models are used to represent human bodies [6], and in medical image analysis and computational anatomy to model the variability of biological shapes [7, 8, 9, 10, 11, 12].

The reconstructed 3D surfaces can be used for computer animation of faces and whole bodies [13, 6], e.g. for movies or computer games or to customize 3D avatars. Such avatars are used by computer users as an alter ego in computer games or online communities. The shape can also be used indirectly to manipulate the expression or attributes of faces in photographs, e.g. by transferring the expression from one photograph to another [13, 14], to exchange faces in photographs [15], to generate stimuli for psychological experiments [16], and for automatic face recognition [17, 2, 18, 19].

Avatar of the XBOX 360

Currently there exist two types of face recognition systems: Firstly, 3D face recognition systems which use a 3D scan. These systems are typically used for access control. The drawback of these systems, however, is that they require a 3D scan and can therefore only be used in a cooperative scenario. Secondly, 2D face recognition system which use a single image or a video stream. The advantage of these systems is their potential use in uncooperative scenarios like video surveillance. However, in practice, their application is prevented by too low recognition rates of the currently available commercial systems [20]. Automated 2D face recognition is still one of the most challenging research topics in computer vision, and it has been demonstrated that variations in pose and light are major problems [21]. Other problems are hair and beards and partial occlusion e.g. from glasses. Most of these systems use 2D images or photos to represent the subjects in the gallery. Hence, these methods are limited in their expressiveness and therefore most face recognition systems show good results only for faces under frontal or near frontal pose. Methods that are based on fitting 3D statistical models have been proposed to overcome this issue [17]. They are the key to recognize people in non-frontal poses or from images with unknown illumination conditions.

1.1 Prior Work

Most statistical models are based on a Principle Component Analysis (PCA) of a set of training data. Turk and Pentland use a 2D PCA model to represent faces, the Eigenfaces [22]. As training data set they use a set of photographs of human faces. The pictures are coarsely aligned, but not registered with dense correspondence. Their system works only well on adequate images, i.e. for pictures in frontal view with controlled illumination and without expression, e.g. passport or drivers license photographs.

Cootes, Taylor et al. represent the face as a set of 2D points (Active Shape Models, ASM [23]). As training data they also use a set of face images. But contrary to the Eigenface approach, they manually register the face images by labeling 152 2D landmark points. The ASM is a PCA model of these face shapes in frontal view. In [24, 25] they combine this model with a PCA model of pixel intensities. This model is called Active Appearance Model (AAM). The ASM/AAM separates shape from appearance, however, it does not separate between 3D pose changes and shape and between illumination and inherent color of the face.

In contrast to these 2D approaches, the Morphable Model represents a face as a 3D shape with per-vertex color. The Morphable Model is a statistical model for shape and per-vertex color that is trained from a dataset of 200 densely registered 3D scans. Blanz and Vetter [1] used their Morphable Model to reconstruct the 3D shape of a face from a photograph. This fitting is done in an Analysis-by-Synthesis approach by optimizing a cost function that consists of the difference between the rendered model and the input photo and a term that controls the probability within the model. Romdhani and Vetter [2] later improved the fitting by using a cost function that included several features extracted from the image, such as the contours, landmark points, and shading information.

PCA models are widely used, but they have some major drawbacks. PCA is focused on dimensionality reduction. The principal components are holistic, i.e. each component has global support. Hence the influence of each coefficient is not localized and affects the whole shape. As a result, there is, in general, no meaningful interpretation of the components. This is counter-intuitive when the model is used in an interactive tool. In the context of human faces, we would expect to be able to change e.g. the shape of the nose independently of the shape of the ear, but this is not possible with PCA models. Holistic Morphable Models are not flexible enough to locally adapt to several features at the same time.

Local Feature Analysis (LFA) [26] was introduced as an alternative method to PCA to overcome the holistic limitations. The PCA representation is typically not topographic, meaning that nearby components of the PCA-coefficient vector have no spatial relationship. In a topographic model, nearby components should have the same relationship as the corresponding components in the input vector. Penev and Atick [26] have introduced a model that achieves this goal, the Local Feature Analysis (LFA).

In [27] we developed an object class representation based on a combination of PCA and LFA approaches. Since both methods have complementary advantages, a combined method is reasonable. A low dimensional global PCA model is combined with a local representation by a LFA model. Local models are developed around landmark vertices. In this approach the PCA model is used for a coarse holistic shape representation and details are represented by the LFA-based local models. This object representation provides local adaptation of the surface and is able to fit 3D control points exactly without affecting areas of the surface far away from these control points. The disadvantage of this representation is that the local models depend on the 3D landmark vertices chosen while training the model.

For non-generative models (like models used for object or face detection) there also exist approaches to represent the localized spatial structure of face images [28, 29]. One such approach is based on the observation that many classes of images (e.g. wavelet transforms of geometrically aligned face images) have a sparse structuring of their statistical dependency [28]. Therefore a classifier for face detection is trained which decomposes the input images into subsets. Only the statistical dependency within each subset is represented and the subsets are treated as statistically independent. Previous parts-based methods for face recognition like [29] used manually defined parts.

In [30] a hierarchical Active Shape Model based on the wavelet decomposition is proposed. The contour shape is decomposed using Haar and Daubechies-7 wavelets. Similar to our approach, the covariance matrix of the wavelet coefficients is approximated by a matrix with block diagonal structure. The wavelet coefficients are divided into bands of fixed size and PCA is performed on each band individually. In [31] a similar model of 2D shapes that have a local self-similarity along their contour is developed. The contour is segmented and each segment is modeled using Legendre polynomials followed by a PCA of the co-registered set of contour segments.

In [32] an approach similar to [30] is used for 3D medical imagery using conformal mappings and spherical wavelets, and an algorithm is presented to cluster the coefficients via spectral graph partitioning to model 3D brain structures.

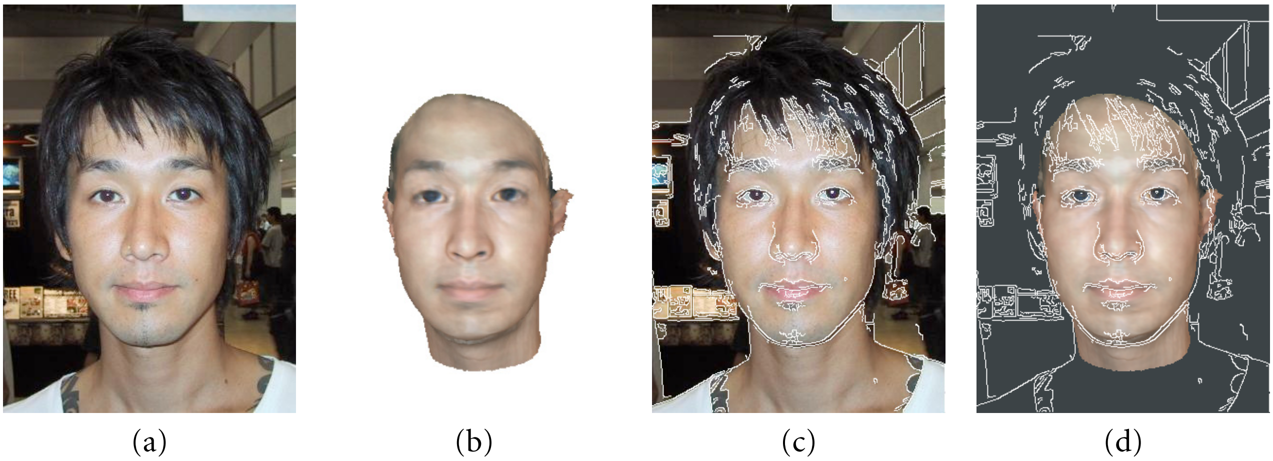
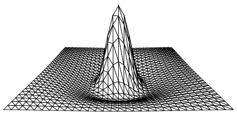


Figure 1.1: Photograph (a) and reconstruction of the 3D shape from (a) using a PCA model for shape and surface color (b). The fitting algorithm [2] uses multiple features, in particular the detected edges (Canny Edge Detector) (c). This involves a trade-off between likelihood of the face and accuracy of the feature matches. Not all features are matched perfectly (d), in particular at the chin. The result with the G2L model is shown in Figure 5.11.



However, not all wavelets are suitable for 3D shapes. In particular, Haar and Daubechies wavelets cause severe artifacts like undulations or spikes. We have investigated this approach in [33]. To overcome these problems in [33] we have used Surface Wavelets as proposed in [34]. A Surface Wavelets base function is shown in the margin.

The wavelet transform has two properties that are of interest in this context: the base functions have local support and the decorrelation of the wavelet coefficients. However, the wavelet transform does not completely decorrelate real-world signals [35]. In particular, wavelet coefficients tend to be large/small when adjacent coefficients are also large/small. For this reason, in [30, 32] patches of wavelet coefficients are used.

However, there are two reasons, why we are not using the wavelet transform of the surface. Firstly, the oscillation of wavelet base functions caused artifacts in our attempts of building statistical models of the 3D surface (even with surface wavelets).

Secondly, the tree-like structure of the wavelet coefficients is not compatible with a generative model in which the facial surface is represented as a linear combination of components. This is needed for fitting the model to data using an Analysis-by-Synthesis approach (see Chapter 5). For these reasons, we are not using wavelets, and use an additive approach instead.

1.2 The Global-to-Local Model

The face space spanned by the model should include all reasonable human faces and exclude all non-faces. The space spanned by the principal components is too limited and too flexible at the same time. On the one hand, it does not span the space of all faces. Every face in this space is an affine combination of the training samples. As a result, it can poorly represent novel faces, i.e. those which are not in the database used to train the model. Hence, the PCA model is not able to represent all possible human faces. On the other hand, overfitting occurs when the model is used for generalization from partial information and is forced to adapt locally to features [36]. In particular, overfitting is a practical problem when a PCA Morphable Model is fitted to a photograph. Hence, the model is too flexible, it is able to generate things that are *not* faces. This overfitting can be repressed by regularization on the cost of poor fitting of the partial information (for an example, see Figure 1.1). One has to choose a trade-off between accuracy of the reconstruction and likelihood of the result being a face [36].

In this work we present a novel analysis method to build a statistical model, the Global-to-Local (G2L) model. This model improves over PCA-based models in three respects: Firstly, it is more flexible. It can better represent novel faces and can better adapt locally to partial information. Secondly, the model is more strict, it only generates valid faces when forced to adapt locally. Thirdly, the model has components with local support which describe specific features of the face.

1.3 Overview

This thesis is organized as follows: Chapter 2 describes the training data. In Chapter 3 the G2L model is introduced. In this work, two different G2L models are built, which are trained from two different data sets. The first data set is the data set that has previously been used in our research group, and the second data set is trained from newly scanned faces, see Chapter 2. To compare the G2L model with the PCA model, two test scenarios are used. The performance of the models when used to reconstruct the 3D shape from a sparse set of 3D landmark points is compared in Chapter 4. This is a controlled and standardized test scenario to compare the models independently from the fitting algorithm, and we demonstrate the improvements of G2L models compared to PCA models for both data sets. In Chapter 5 we apply both models to reconstruct the 3D shape from images. For this we use a novel adaptive fitting method consisting of multiple modules that optimize the parameters successively.

Chapter 2

Data and Parametrization

Statistical models are built from data. The construction of these models, i.e. the data acquisition and registration, is a tedious and expensive process, both in time and money. The quality and expressiveness of statistical models heavily depend on the quality of the data. High quality data is the foundation and prerequisite of a good statistical model.

The data acquisition was a joint work with Pascal Paysan, while the registration was done by Brian Amberg, in [37] more detailed information about the data acquisition and the registration can be found.

2.1 The Data

In this work, we use two different data sets. From each data set, both a G2L model and a PCA model are built. In Table 2.1 a detailed comparison of the data sets (including several variants) is given.

2.1.1 MPI-TÜ Data

The first data set is the Max-Planck-Institut Tübingen (`MPI-TÜ`) Morphable Model [1], which has been intensively used in our research group before, e.g. for fitting [38, 1, 36, 2, 14, 15]. Currently, to the best of our knowledge, there exists only one comparable set of registered 3D scans of faces, the University of South Florida (`USF`) MM [39].

In their original version, the 200 `MPI-TÜ` scans are registered with a 3D optical flow algorithm. This data did not include the backs of the heads and no interior of the mouth, see Table 2.1, 1st row. Here, however, we use a variation of this data which is registered with a different method [40], thereby estimating the back of the head. In this model, all registered scans have an arbitrary polygonal mesh. To

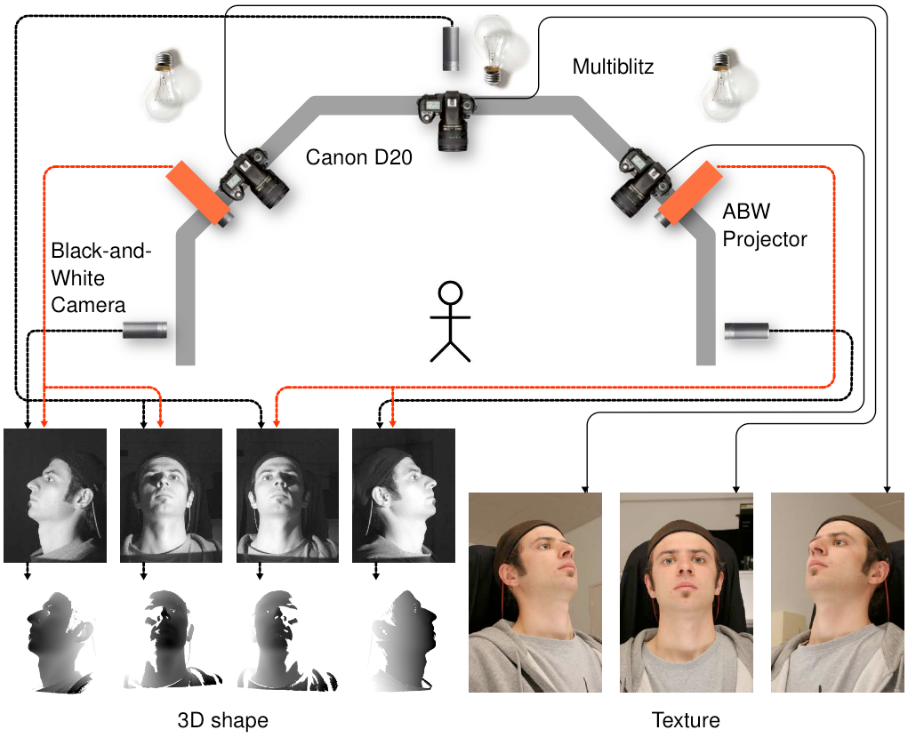


Figure 2.1: 3D face scanning device developed by `ABW-3D` [42]. The system consists of two structured light projectors, three black-and-white cameras for the shape, three 8 mega pixels DSLR cameras and three studio flash lights (Multiblitz).

be compatible with the G2L model, this data has been re-parametrized in a cube representation, see Table 2.1, 2nd row. In addition to this training set, there exists a test set of 25 `ABW-3D`-scans (see Section 2.2), for which the same registration method [40] was used.

2.1.2 UniBS Data

The second data set was acquired in our lab at the University of Basel between December 2006 and September 2008. Compared with the `MPI-TÜ` data, the newly scanned data (called `UniBS` data from now on) are superior in several aspects: Firstly, our 3D scanner [42] (see Figure 2.1) offers higher resolution and higher precision in shorter scanning time than the `Cyberware` [41] scanner used for the `MPI-TÜ` and `USF` models, resulting in more accurate scans. Secondly, a different registration method is used, which, together with a better parametrization, yields less correspondence artifacts, see Figure 2.6.

For the experiments we use different data sets that vary in number of samples and registration method used to establish correspondence. The `UniBS-A` data set consists of 125 scans of 125 individuals, all with neutral expression. They are split randomly into a training set of 100 scans and a test set of 25 scans. This data set has been acquired before February 2008.

	MPI-TÜ Model laser scanner, `Cyberware`[41], scan time $\sim$30 s 200 scans registered with optical flow [1] 75972 vertices / vertex color facial mask / mouth closed / no expression cylindrical coordinates
	MPI-TÜ-C Model laser scanner, `Cyberware`[41], scan time $\sim$30 s 200 scans registered with optical flow [40] 78082 vertices / texture mapping with back of the head / mouth opened / no expression arbitrary parametrization (mesh)
	Basel Face Model (BFM) structured light scanner, `ABW-3D`[42], scan time $\sim$0.5 s 200 scans registered with ICP (modification of [43]) 53490 vertices / vertex color facial mask / mouth opened / no expression cube parametrization
	UniBS-A structured light scanner, `ABW-3D`[42], scan time $\sim$0.5 s 100 scans (`TRAIN`), 25 scans (`TEST`) registered with ICP (modification of [43]) 97577 vertices / vertex color or high-res textures with back of the head / mouth opened / no expression cube parametrization
	UniBS-B structured light scanner, `ABW-3D`[42], scan time $\sim$0.5 s 200 scans (in total: 459 neutral scans available) additional 20 MRI scans registered with ICP (modification of [43]) 97577 vertices / vertex color or high-res textures full head (with back of the head) / mouth opened / plus 709 expression scans available cube parametrization

Table 2.1: The table compares the `MPI-TÜ` model, the Basel Face Model (`BFM`) and the data used in this work (`UniBS` data). The term 'mouth opened' means that the lips are not connected and that the interior of the mouth is modeled.

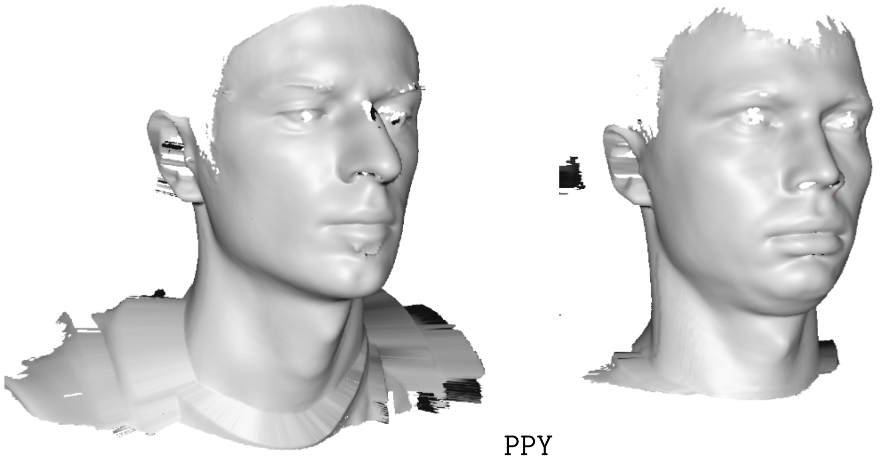


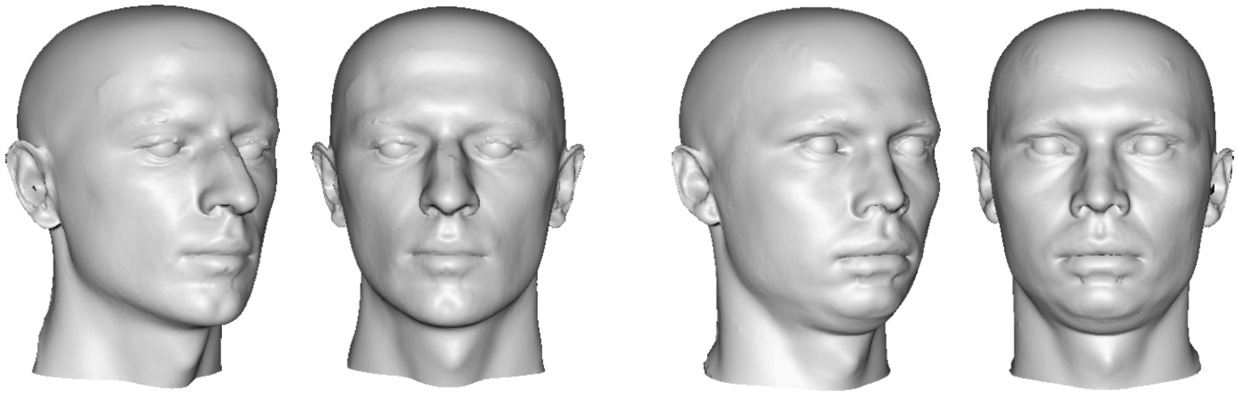
Figure 2.2: Scans acquired with our structured light scanner have holes, and the back of the head is missing. The image shows two examples of the test set, with id `FLX` and `PPY`. In Figure 2.3 the registered scans are shown.

The `UniBS-B` data set consists of 200 scans of 200 individuals, the same 200 scans that are used for the `BFM` (Basel Face Model), see below. This data set is a subset of all available data. In total, 1168 registered scans of 286 individuals are available, 459 of them have neutral expression. For 709 scans, we asked the subjects to show the basic emotions joy, fear, disgust, anger, sadness and surprise. In [44] Paul Ekman concludes that this list of basic emotions is biologically universal to all humans. In comparison to the `UniBS-A` data set, the back of the head is estimated during a different registration method.

The Basel Face Model (`BFM`) [37] is a PCA model that will be made publicly available. The `BFM` consists of 200 scans and only the frontal mask without the back of the head is used to compute the model. The intention here is to have a model that is comparable with the `MPI-TÜ` model, both in size and in resolution of the data.

2.2 Scanner and Data Acquisition

Scanning of human faces is a challenging task for several reasons. First of all, the acquisition time is critical. We decided to use a structured light system because of its shorter acquisition time ($\sim 0.5\ s$) compared to laser scanners ($\sim 30\ s$). This structured light system is a prototype system by `ABW-3D` [42] with some modifications. It is an active stereo vision system with white light sources. Active stereo systems have a higher accuracy compared to passive systems, because of the lack of features e.g. on the cheeks. Because the baseline of stereo systems is restricted, the scanner uses a multi-view approach with four subsystems, as shown in Figure 2.1. The left and the middle black-and-white camera together with the left projector build two stereo subsystems, and accordingly the middle and the right black-and-white camera together with the right projector build the remaining two



FLX PPY

Figure 2.3: To build statistical models, the data needs to be in correspondence: the registration establishes a common parametrization, fills in holes, and adds the back of the head. The image shows two examples of the `UniBS-A` test set. In Figure 2.2 the scanned data is shown that was used as input for the registration.

stereo subsystems. The system captures the facial surface from ear to ear, the 3D shape of the eyes and hair cannot be captured with our system due to their reflection properties. The resolution of the geometry measurement is higher than all comparable systems we are aware of. Examples of such scans can be seen in Figure 2.2. With each scan, three photos with DSLR (digital single-lens reflex) cameras (Canon D20, 3504×2336 pixel, sRGB color profile) are taken simultaneously to capture the surface color. Three studio flashes with diffuser umbrellas are used to achieve a homogeneous ambient illumination.

2.3 Parametrization and Correspondence

To build statistical models, the data needs to be in correspondence. This means that corresponding points are mapped to the same point in the parametrization for all shapes. When shapes are in correspondence, it is possible to morph between the shapes, and hence affine combinations of registered faces represent new faces.

To establish correspondence, a nonrigid ICP method is used. The method is a modification of the method developed by Brian Amberg [43]. In Figure 2.3 the registered scans of Figure 2.2 are shown. The registration method is applied in the 3D domain on triangular meshes. It progressively deforms a template mesh towards the measured surface. Missing data is added for regions that cannot be scanned due to their reflection properties (e.g. eyes, hair), for regions with artifacts from the scanning (e.g. the nose bridge) and for regions that are not covered by the scanner (i.e. the back of the head). All registered data share a common

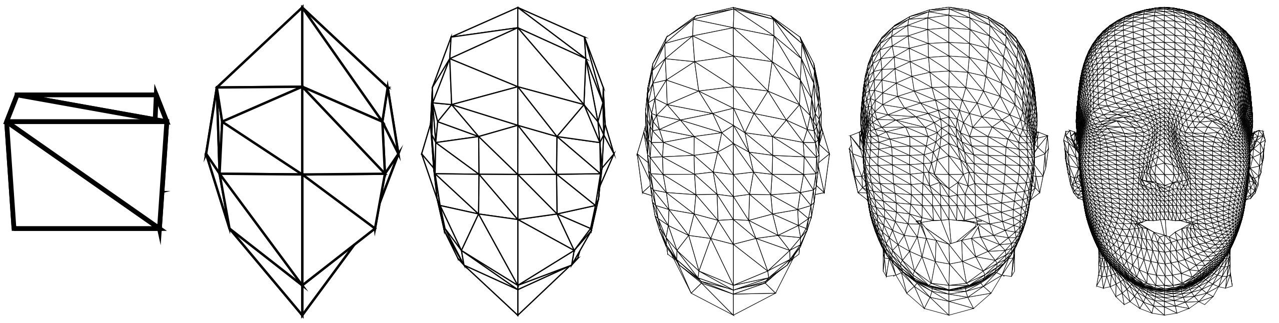
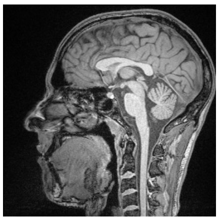


Figure 2.4: The regular grid structure of the template provides the hierarchy of the multiresolution scheme.

parametrization that is defined by the parametrization of the template (see Section 2.4).

For the `UniBS-A` data set, the back of the head is estimated only by deforming the template shape and not by statistical knowledge derived from measuring data. To compute the G2L model, the shape of the full head is needed because the multiresolution analysis would be disturbed by the boundary, if only a facial mask were used (this will be explained in Section 2.4 in more detail). The statistical analysis, at least that one for the G2L model, is not disturbed by the estimated data. However, as can be seen in Figure 2.3, the estimated backs of the heads are not optimal, they are slightly too small. To overcome this problem, the missing data at the back of the head is estimated using MRI scans (like the slice shown in the margin). Therefore, Anita Lerch, Thomas Albrecht and Marcel Lüthi segmented the facial surface of 20 MRI scans of human heads [45]. From these surfaces, Brian Amberg built a PCA model. The back of the head of the individuals scanned with the structured light scanner is estimated by using the MRI-PCA model in a bootstrapping process while registering the data.



2.4 The Template

Due to the correspondence of the training data, the meshes of the registered scans differ only in the position of the vertices, but not in the list of the quadrilateral. This list is the same for all registered scans and defined by the template. The template has a regular grid structure, i.e. each vertex has four neighbors (except for eight irregular vertices with three neighbors). The regular grid structure is important, since it easily allows a multiresolution analysis for shapes, as shown in Figure 2.4. The regular mesh is defined as a *Parametric Surface*

$$\Gamma = F(U) \tag{2.1}$$

where $U \subset \mathbb{R}^2$ is open and $F : U \to \mathbb{R}^3$.

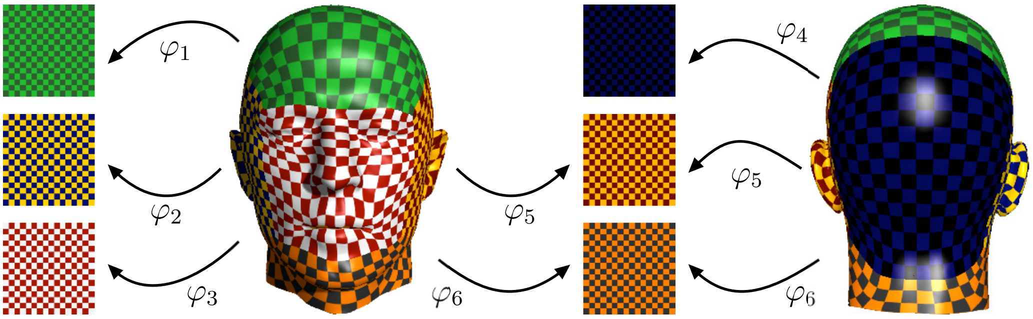
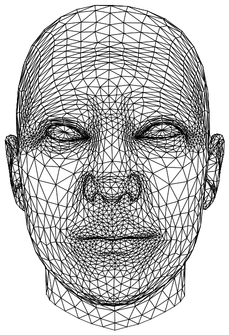


Figure 2.5: The template surface can be interpreted as a 2-dimensional submanifold with six charts.

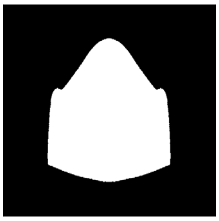
Representing the template as a parametric surface not only has the advantage that it provides multiresolution analysis in a straightforward way. It also simplifies the usage of measures like curvature. Most problems can be addressed in the parametric space $U \subset \mathbb{R}^2$ instead of the 3D surface $\Gamma \subset \mathbb{R}^3$. This simplifies problems for surfaces embedded in $\mathbb{R}^3$ to problems for images [46].

Meshes with arbitrary topology, like the mesh shown in the figure in the margin [40], have vertices with an arbitrary number of neighbors. For these meshes multiresolution analysis can be done with an approach like [34]. Here, however, we prefer to use a template that supports multiresolution by design.



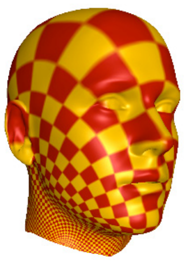
2.4.1 Parametrization

Parameterizations are in general not unique. We use a parametrization tailored especially for our needs. The surface of the template (model $\mathcal{T}$) is interpreted as a 2-dimensional submanifold of $\mathbb{R}^3$ and parametrized by six charts $\phi_i : U_i \to \phi(U_i)$ where $U_i \subset \mathcal{T}$ are the front, back, left, right, top or bottom face, see Figure 2.5.



The `MPI-TÜ` model is parametrized with cylindrical polar coordinates, where the parameter domain is cropped by the mask shown in the margin. This boundary interferes with the development of multiresolution analysis. For this reason, we prefer a parametrization with fewer boundaries, that covers the full head, not only the frontal face.

The shape of our template is open at the bottom (at the neck line). The mouth can be opened (i.e. the upper and lower lips are not connected), however, the interior of the mouth is modeled, therefore there are no boundaries. In a similar way, the nostrils are modeled without boundaries. For this reason, the Euler characteristic of the template shape is $\chi = 1$. The Euler characteristic is a topological invariant, and $\chi = 1$ means that the surface is topologically equivalent to a disk.



This would suggest a parametrization with a square as parameter domain. In the figure in the margin a parametrization is shown, where the neckline is mapped to the border of a quadratic parameter domain. This leads to severe distortion. Thus, it is more appropriate to consider the human face (with back head) as topologically equivalent to a sphere (Euler characteristic $\chi = 2$) and mark the region at the neck as invalid.

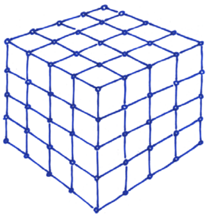
For this reason, we decided to use a parametrization where the parameter domain has also Euler characteristic $\chi = 2$ with six charts that are arranged as a cube (see Figure 2.5). This parametrization involves only eight irregular points, namely the eight corners of the cube. These points are placed in such a way that the boundaries between the charts pass through regions that are of no special interest: they go through the cheeks, the chin and the forehead, but not through the nose, the mouth, the eyes and the ears. Additionally, some important landmarks of the face are mapped to special points in the parametrization, e.g. the tip of the nose is mapped to the center of the frontal chart.

Choosing the parametrization involves a trade-off between the number of irregular points and the distortion caused by these points. A parametrization with two irregular points (e.g. the map of the earth with north and south pole) would also be a possible parametrization. However, such a parametrization would introduce quite severe distortion at each of the poles. Using a parametrization with eight irregular points is a reasonable trade-off between number of singularities and distortion.

A parametrization with low distortion is especially important when computing affine combinations (morphs) of the data. The effect of distortion on the model quality is illustrated in Figure 2.6. On the left side, the `MPI-TÜ` model with its cylindrical parametrization shows its typical artifacts on the chin. Due to the cylindrical parametrization, only a few vertices lie in the chin region. The cube parametrization distributes the vertices more uniformly over the surface. Notably it has enough vertices in the chin region and therefore avoids these artifacts.

The model used here only consists of data with neutral expression implying a closed mouth. In Section 6.2 (future work) we discuss an extension of the model towards expressions. For this, it is necessary that the template covers the lips and the interior of the mouth, since all scans are registered to a common reference frame.

2.4.2 Data Representation



Based on the parametrization, a mesh can be generated in different ways and with different resolutions. We use a regular grid-structured quadrilateral mesh with $(2^l + 1) \times (2^l + 1)$ vertices on each chart (on level l), in which the vertices at the boundaries are shared by two charts. In the figure in the margin, such a mesh on level $l = 2$ is shown with 5×5 vertices at each chart. In this mesh, each vertex has four neighbors, except for the eight corners, which have three neighbors. Typically, we use meshes up to level $l = 7$. Thus, the full mesh has $6 \cdot 2^{2l} + 2 = 98306$ vertices. As some of these vertices are marked as invalid (on the neck), the mesh

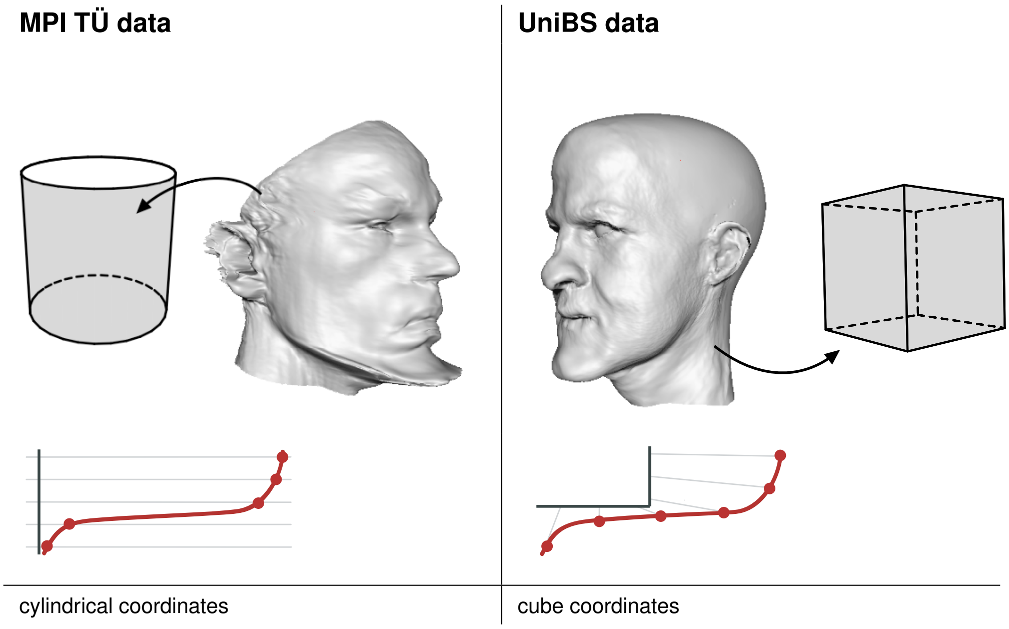


Figure 2.6: Correspondence artifacts caused by the surface parametrization occur especially for larger values of model coefficients. The renderings of the same random coefficient vector ($c_i \sim \mathcal{N}(0,(2.5)^2)$) with the `BFM` (right) show less artifacts compared with the `MPI-TÜ` model (left).

has fewer vertices ($m = 97577$). Each vertex j is represented by a 3D point $\vec{x} = (x_x, x_y, x_z)^T \in \mathbb{R}^3$. Due to the correspondence, the mesh topology is the same for each face. Each shape can be represented as a $3m$ dimensional vector

$$\mathbf{x} \;=\; (x_{x,1}, x_{y,1}, x_{z,1}, \ldots, x_{x,m}, x_{y,m}, x_{z,m})^T \,. \tag{2.2}$$

The inherent color or albedo of the face is represented with a simple reflectance color per vertex, i.e. for the head, the color is represented in the $3m$ dimensional vector

$$\mathbf{a} \;=\; (a_1^R, a_1^G, a_1^B, \ldots, a_m^R, a_m^G, a_m^B)^T \,. \tag{2.3}$$

Using per-vertex color has several advantages. When statistical models are used for fitting, per-vertex color simplifies the cost functions, since the cost function can then be defined as a sum over the vertex positions, see Section 5.2, Equation 5.19. Using color information in higher resolution would increase the cost for evaluating the cost function and therefore reduce the performance of the fitting.

In the pictures taken by the scanner, fine details like stubble and freckles are visible. However, these (fine) details are not in correspondence after registration, not only because color information is not used for registration, but also because it is unclear how to expand the concept of correspondence to such fine details. The statistical models used here, however, require correspondence, i.e. increasing the resolution would not necessarily improve the quality of the model, since the fine details get lost by the linear combinations. Hence, other types of statistical models are needed for this, in Section 6.4 (future work) this is discussed. In this work, we solely use per-vertex color for the statistical color model.

On the other hand, there clearly are applications that can take advantage of higher resolution color information. For this reason, the color information can be represented in the form of texture maps, where each of the six charts (see Figure 2.5) defines one texture map. This technique is used in Section 5.6.6, where in a post processing step after fitting, the texture of the face is extracted from the input photograph.

Chapter 3

The Global-to-Local Model

Models are simplified, abstract representations or descriptions of objects, systems, phenomena, etc. In our case, when we refer to the term model, we usually think of a representation not for one single object, but for a whole class of objects. Accordingly, a face model should be able to describe or code the shape (and the surface color) of all human faces.

The models used here are *parametric, generative* models. This means that an individual is described by a small vector of parameters or coefficients. From these coefficients, the 3D shape (and the surface color) can be generated. In computational anatomy, such models are often called *statistical* models. It is assumed that the coefficients are distributed according to a normal distribution.

Definition A *linear generative statistical* face model is a model

$$\mathcal{M} := (\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{U}) \tag{3.1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{3m}$ is the mean, $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{n'}] \in \mathbb{R}^{3m \times n'}$ is a matrix with n' components and $\boldsymbol{\sigma}$ is a weighting of the components, with the following properties:

- Given a coefficient vector $\boldsymbol{\alpha}$, the shape of a face can be generated by the mean plus a linear combination

  $$\mathbf{x} = \boldsymbol{\mu} + \sum_i \alpha_i \sigma_i \mathbf{u}_\mathbf{i} \,. \tag{3.2}$$

- The coefficients

  $$\alpha_i \sim \mathcal{N}(0, 1) \tag{3.3}$$

  are independent and distributed according to a normal distribution.

In this chapter, we discuss two methods of training such models from data. In previous work [1, 2], the well known Principle Component Analysis (PCA) was used. In order to improve it, in this work we develop a novel analysis method, called the Global-to-Local model (G2L). For comparison, both models are constructed from the same set of n training examples $\mathbf{x}_i \in \mathbb{R}^{3m}$, where m is the number of vertices.

3.1 PCA Model

The PCA model

$$\mathcal{M}_{\text{PCA}} := (\boldsymbol{\mu}, \boldsymbol{\sigma}_{\text{PCA}}, \mathbf{U}_{\text{PCA}}), \tag{3.4}$$

is constructed by applying a widely used technique for dimensionality reduction, Principle Component Analysis (PCA). For the ease of notation we will drop the subscript index in the following. The vector $\boldsymbol{\mu} \in \mathbb{R}^{3m}$ is the mean, $\boldsymbol{\sigma} \in \mathbb{R}^{n-1}$ the standard deviation and $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{n-1}] \in \mathbb{R}^{3m \times n-1}$ is an orthonormal basis of principle components. From n training examples, the mean and $n-1$ principle components are computed. The remaining principle components are irrelevant, since the corresponding standard deviation $\sigma_i = 0, \forall i \geq n$. The mean face is estimated as the average

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \tag{3.5}$$

of the examples. The mean-free data vectors are stacked into the data matrix

$$\mathbf{X} := [\mathbf{x}_1 - \boldsymbol{\mu}, \dots, \mathbf{x}_n - \boldsymbol{\mu}] \tag{3.6}$$

and decomposed into the column-orthonormal matrix $\mathbf{U}$, the diagonal matrix $\mathbf{W} \in \mathbb{R}^{n-1 \times n-1}$, and the orthonormal matrix $\mathbf{V}^T \in \mathbb{R}^{n-1 \times n}$, using singular value decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{W}\mathbf{V}^T \,. \tag{3.7}$$

The standard deviation $\sigma_i = w_i/\sqrt{n}$ of the examples projected onto the principle components $\mathbf{u}_i$ is computed from the diagonal elements w_i of the matrix $\mathbf{W}$. The principle components $\mathbf{u}_i$ form the basis of the model space. A face $\mathbf{x}$ can be described with respect to this basis by the parameters α_i as a linear combination

$$\mathbf{x} = \boldsymbol{\mu} + \sum_{i=1}^{n-1} \alpha_i \sigma_i \mathbf{u}_i \,. \tag{3.8}$$

The covariance matrix of the data is given by

$$\boldsymbol{\Sigma} \;=\; \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathbf{T}} = \frac{1}{n}\mathbf{U}\mathbf{W}\mathbf{V}^{\mathbf{T}}\mathbf{V}\mathbf{W}\mathbf{U}^{\mathbf{T}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathbf{T}} \tag{3.9}$$

where $\mathbf{\Lambda} = \frac{1}{n}\mathbf{W}^2$. Since the matrix $\mathbf{U}$ is column-orthonormal, the coefficients $\boldsymbol{\alpha}$ are given by the projection

$$\boldsymbol{\alpha} = \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_{n-1}^{-1})\mathbf{U^T}(\mathbf{x} - \boldsymbol{\mu}) \tag{3.10}$$

into the model space. Under the assumption that the training data $\mathbf{x_i}$ are distributed according to a normal distribution with mean $\boldsymbol{\mu}$,

$$\begin{aligned} p(\mathbf{x}) &= \nu \prod \exp(-\frac{1}{2\sigma_i^2}\langle(\mathbf{x} - \boldsymbol{\mu}), \mathbf{u}_i\rangle^2) & (3.11) \\ &= \nu \exp(-\frac{1}{2}\sum \frac{1}{\sigma_i^2}\langle(\mathbf{x} - \boldsymbol{\mu}), \mathbf{u}_i\rangle^2) & (3.12) \\ &= \nu \exp(-\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}) = \nu \exp(-\frac{1}{2}\|\boldsymbol{\alpha}\|^2), & (3.13) \end{aligned}$$

the coefficients $\alpha_i \sim \mathcal{N}(0, 1)$ are independent and identically distributed (ν is a normalization constant). An examples of such a PCA model can be seen in Figure 3.1.

PCA-based models, however, have some severe disadvantages. The number of coefficients is limited by the number of training samples n. For instance, the training set `UniBS-A` contains $n = 100$ scans and therefore the model is limited to 99 components. As a result, all possible faces have to be represented by a vector of length 99. The PCA model is not flexible enough to represent novel faces and performs well only on the training data, but not on test data.

PCA models are holistic, i.e. each component has global support. As a result, changing only one coefficient α_i affects every part of the shape. The components have no meaningful interpretation, as can be seen in Figure 3.1. Each component encodes some details of the nose and some details of the forehead and some details of the ears, etc. at the same time. Changing e.g. the shape of the nose is not possible without affecting the ears and the chin, i.e. it is not possible to change one facial feature and keep all other vertices fixed.

The very last components only encode noise in the training set, for an example see the figure in the margin, where the $99th$ principle component of the `UniBS-A` PCA model is shown (with $\alpha_{99} = 30 \cdot \sigma_{99}$).

When one tries to adapt the PCA model locally to facial features, this is either not possible at all, or, if possible, only with severe overfitting. Overfitting occurs when an unsuitable statistical model with too many parameters is fitted to novel data. The data is matched well and the function or the surface passes through the points or vertices. However, in between, the model generates arbitrary results. In the toy example in Figure 3.2, green curve, this effect is demonstrated: a *holistic* or *global* model (i.e. all components in this model have global support, here: sine functions of different frequency) is fitted to five landmark points. Overfitting can

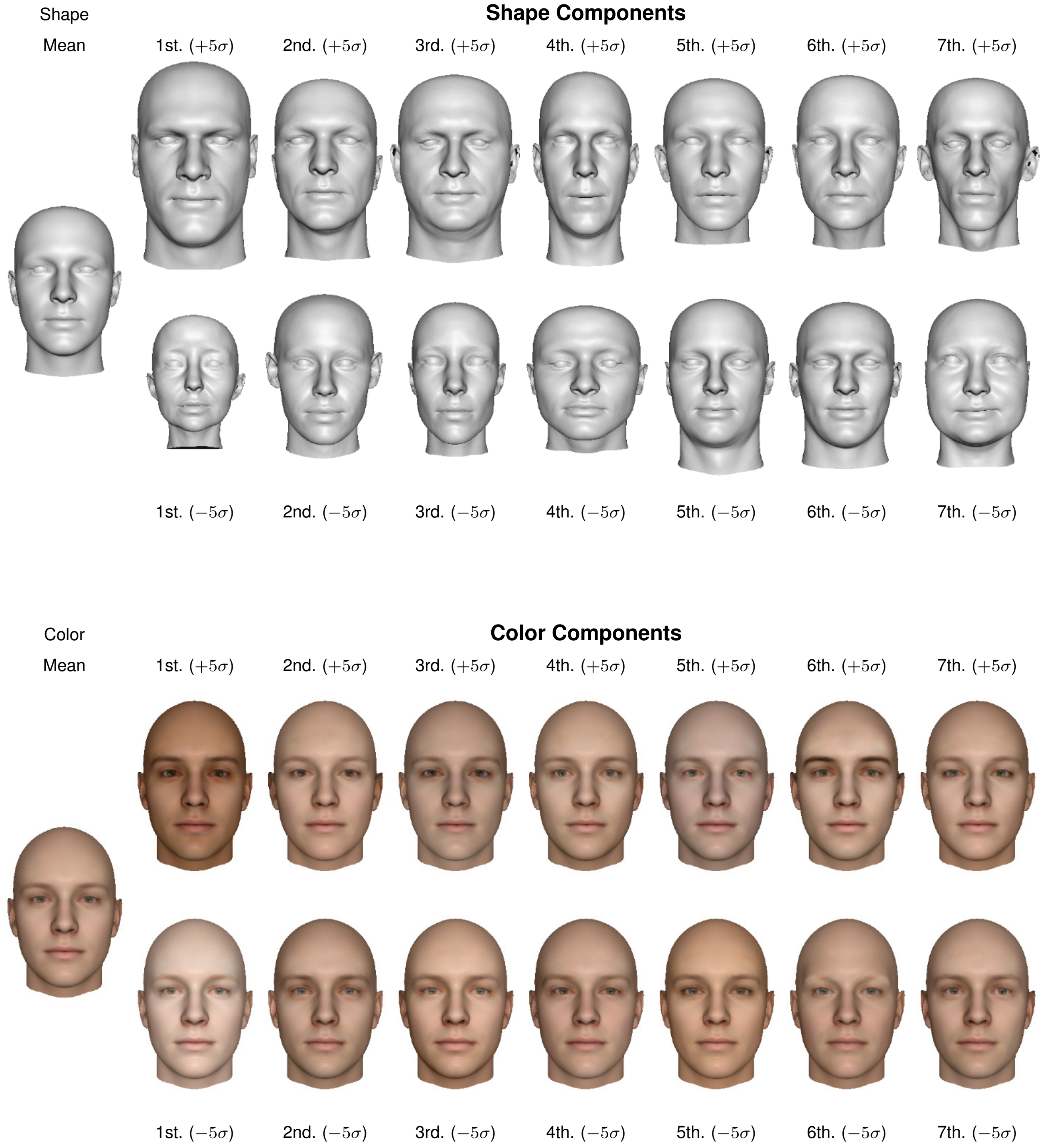


Figure 3.1: The mean together with the variation of the first seven principle components of the shape (top, `UniBS-A` data set) and surface color (bottom, `UniBS-B` data set) PCA model.

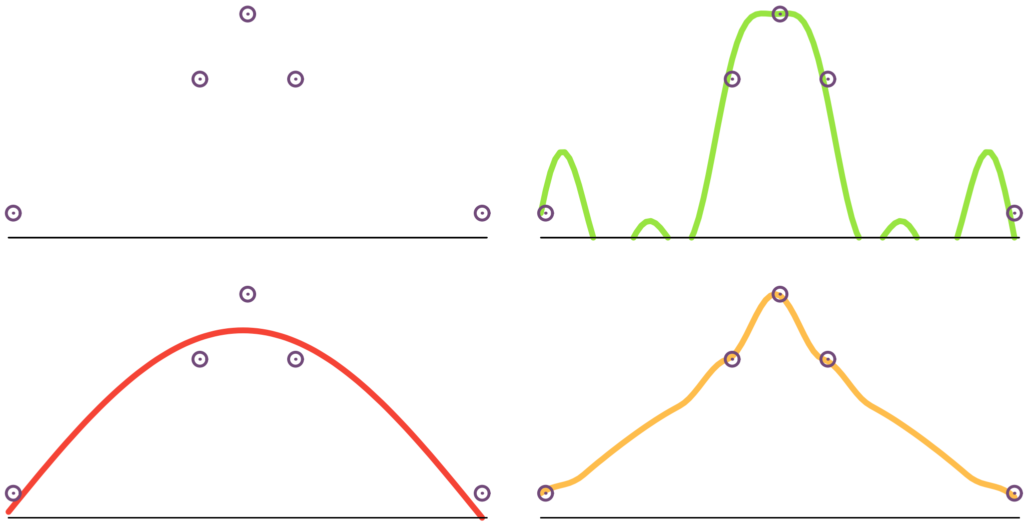


Figure 3.2: In a toy example, different models are used to fit a function to a small set of points (1st row/left). With a model where the basis function have global support ($\sin$-functions), either overfitting occurs (green) or the measured data are not well matched (red). The model (yellow) has a global component and several local components (segmented, overlapping $\sin^2$-functions) and fits the data well without arbitrary results in between.

be prevented by reducing the number of parameters of the model or by regularization. Large coefficients lead to a low prior probability. Regularization penalizes large coefficients, thereby preventing overfitting. Both approaches work in a similar way, since the most relevant components with large standard deviation σ_i are effected less by regularization than those with small σ_i [36].

In the toy example in Figure 3.2, red curve, a holistic model with one component (in this example: one sine function) yields a better fit. However, the measured data is only poorly approximated and for our application, we require the measured data to be fitted more accurately. For our purpose, we can assume that the noise of the measurement (i.e. acquisition of the photo) is much below the error introduced by the (PCA-) model, and therefore a large fitting error is not acceptable. For global models, we have to choose a trade-off between accuracy at the landmark points and plausibility of the reconstruction.

For the yellow curve in Figure 3.2, we have used a toy G2L model that consists of one component with full support (*global* component, here: one sine function) and several components with limited support (*local* components, here: several segmented and overlapping $\sin^2$ functions. This toy G2L model has the same number of coefficients as the global sine model (12 components). This model fits the data without overfitting and approximates the landmark points very well.

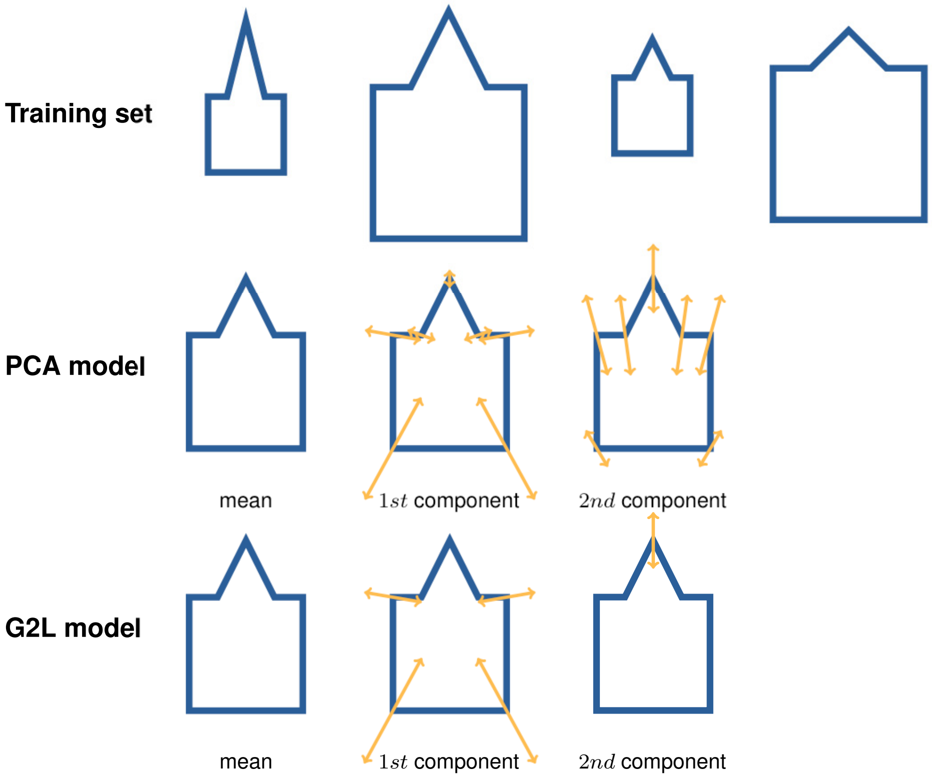


Figure 3.3: From the toy example training set (the four shapes in the $1st$ row) a PCA model ($2nd$ row) and a G2L model ($3rd$ row) are trained. The PCA model has two holistic components. The G2L model has one global component that affects the size of the square and one local component that modifies only the triangle (the *nose*). Although this is a toy example, the PCA and G2L models are in fact computed from the 2D vertex positions of the training set.

3.2 Global-to-Local Model

The goal is to have a model that can adapt to features locally. The model should contain components with limited support, i.e. most of the components are non-zero only within a (small) region. The size of the support has to depend on the kind of the feature. Features like the chin, the mouth and lips require components of different size. Besides the local components, the model has a few global components that control the size of the head and height-to-width ratio without affecting the shape of e.g. the nose or the eyes.

The principle of the global to local analysis is outlined in the toy example in Figure 3.3. In the first row, a set of four shapes can be seen that is used as the training set. The training set suggests that the length of the triangle is independent from the size of the square. However, applying PCA to this set leads to two holistic principle components (the third component has a zero eigenvalue). With this model it is not possible to change the shape of the triangle without changing the position of the vertices of the square.

One possible approach to improve PCA model fits is to split up the surface into multiple regions and fit the model independently in each region. This approach was used for faces by [1, 2] with four regions (eyes, nose, lips, rest). The disadvantage of this approach is that it completely eliminates any global correlation. It is not guaranteed that the separately modeled segments always fit together. To avoid sharp transitions, the separate segments are blended into a single surface. These combined surfaces do not necessarily represent a valid face, as the correlation between segments is neglected.

The G2L model is a different attempt for building a locally adaptable shape model. Instead of allowing local adaption by separate models, we propose a model in which local variation is modeled locally, but these local models are integrated into a single model that represents the whole face. The key idea is to unite the global and local modeling with a multiresolution approach. Global features such as the overall head size and width are modeled by low resolution components while the high resolution components model local features such as the shape of the nose or mouth. The high resolution components are specifically organized in such a way that independent details of the face surface are modeled by separate components. The support of these components is locally limited to the respective facial feature.

The covariance matrix exhibits a specific sparsity pattern: high resolution components are decorrelated from low resolution components and independent facial features within each resolution level are decorrelated from each other. This decomposition is inspired by [30], where 1D wavelet functions were used to analyze a contour in a 2D image.

To construct a model with global and local components (G2L model), we need:

- *Multiresolution analysis of the facial shape.*
  The key idea is to break up the facial shape into components based on spatial frequency. The resulting hierarchy contains components gradually decreasing in spatial frequency.

- *Segmentation of the details.*
  The high frequency components code small facial features, that are clustered in multiple regions, each representing a specific feature. Therefore, these components can be segmented into regions in a meaningful way. For each level in the multiresolution hierarchy, different regions are used, the size of the regions depends on the spatial frequency. Hard edges at the boundaries of the segments are avoided by the multiresolution approach.

- *Building a linear statistical model.*
  From the low frequency components and the segmented high frequency components, separate statistical models are computed. The resulting global and local components are combined into one linear statistical model. This statistical model can be used in the same way as PCA-based statistical models.

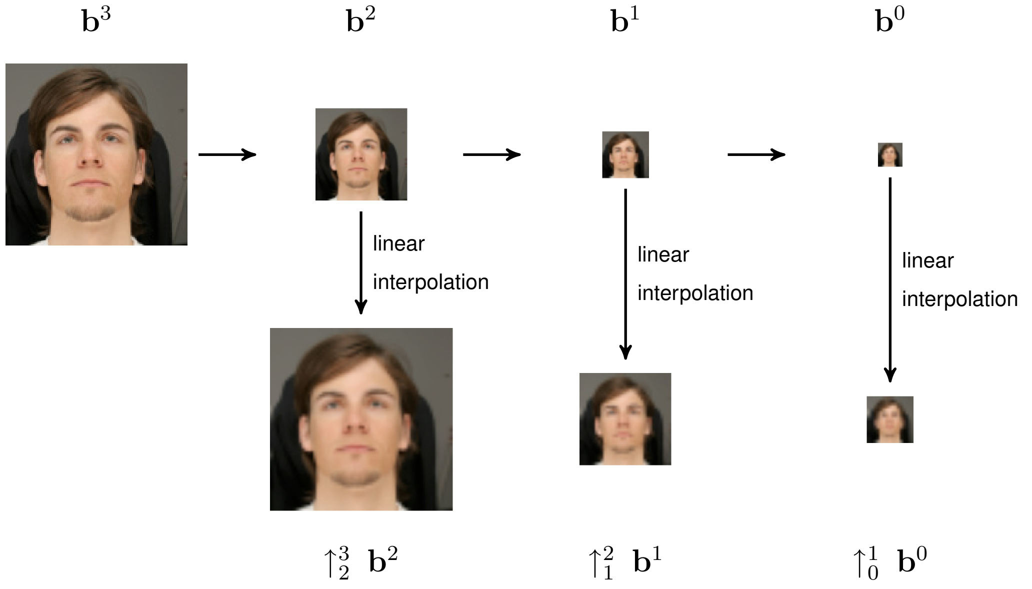


Figure 3.4: The Gaussian Pyramid for images is computed by repeatedly applying a Gauss Filter and subsampling. The horizontal arrows denote Gaussian filtering and subsampling.

3.3 Multiresolution Analysis

In order to develop the multiresolution scheme of the G2L models, we use a concept from image processing to decorrelate image pixels, the Gaussian- and Laplacian Pyramids [47]. This principle is transferred to 3D shapes. For the G2L model, it is used to split up the facial shape and later recombine it without artifacts at the boundaries.

3.3.1 Gaussian- and Laplacian Pyramids

A Gaussian Pyramid for images is generated by repeatedly applying a low-pass filter, typically a Gaussian Filter, to an image $\mathbf{b}$. The filtered images contain less information, therefore they can be subsampled. This process can be repeated recursively, which gives rise to the pyramid structure. The filtered and subsampled images are denoted by $\mathbf{b}^l$, where l is the level in the pyramid. An example of a Gaussian Pyramid is shown in Figure 3.4. Closely related to Gaussian Pyramids are Laplacian Pyramids, where instead of a low-pass filter a high-pass filter is used. The high-pass filtering is done by computing the difference between two consecutive levels of the Gaussian Pyramid

$$\mathbf{d}^l = \mathbf{b}^{l+1} - \uparrow_l^{l+1} \mathbf{b}^l \tag{3.14}$$

where $\uparrow_l^h$ is an interpolation operator. For this toy example, linear interpolation is used. An example of a Laplacian Pyramid is shown in Figure 3.5. Given a Laplacian

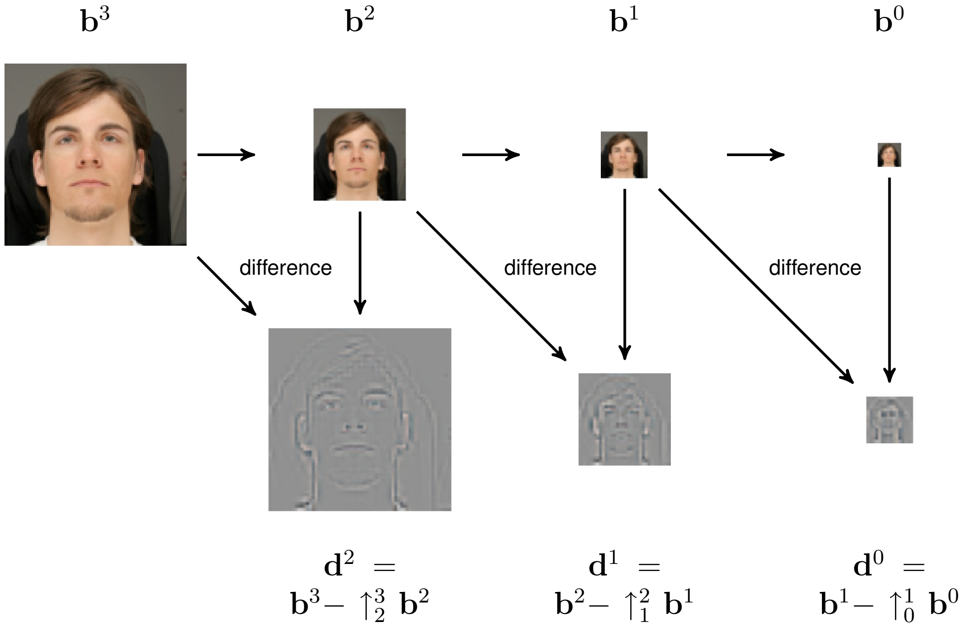

$\mathbf{b}^3$
$\mathbf{b}^2$
$\mathbf{b}^1$
$\mathbf{b}^0$
difference
difference
difference
$\mathbf{d}^2 = \mathbf{b}^3 - \uparrow_2^3 \mathbf{b}^2$
$\mathbf{d}^1 = \mathbf{b}^2 - \uparrow_1^2 \mathbf{b}^1$
$\mathbf{d}^0 = \mathbf{b}^1 - \uparrow_0^1 \mathbf{b}^0$

Figure 3.5: The Laplacian Pyramid for images is computed using the Gaussian Pyramid. This defines a filter-bank of images. The input image $\mathbf{b}^3$ is band-pass filtered: $\mathbf{d}^2$ contains only high spacial frequencies, $\mathbf{d}^1$ only mid-high frequencies, $\mathbf{d}^0$ only mid-low frequencies and $\mathbf{b}^0$ only low frequencies.

Pyramid, the original image on level l' can be recovered

$$\uparrow_0^{l'} \mathbf{b}^0 + \sum_l \uparrow_{l+1}^{l'} \mathbf{d}^l = \uparrow_0^{l'} \mathbf{b}^0 + \sum_l \uparrow_{l+1}^{l'} \mathbf{b}^{l+1} - \uparrow_l^{l'} \mathbf{b}^l = \mathbf{b} \tag{3.15}$$

by recombining all pyramid levels. This is called collapsing the pyramid.

Gaussian and Laplacian Pyramids can be used to blend two images [47]. For example, this technique is used in software for blending multiple photos of a scenery to a large panorama (e.g. [48]). Our main focus is to use this principle on 3D shapes to build the G2L. However, we have also used texture blending to estimate the color of the back of the head, see Appendix 7.3.

3.3.2 Gaussian- and Laplacian Pyramids for Shapes

The concept of Gaussian- and Laplacian Pyramids is defined for 2D images and we now transfer this concept to 3D shapes. As with images, the Gaussian shape pyramid is created by repeatedly applying a low-pass filter and subsampling. The Laplacian shape pyramid is defined as the difference between consecutive levels of the Gaussian pyramid. Thus we need to define two transitions: a *fine-to-coarse* transition that smooths and subsamples the mesh and a *coarse-to-fine* transition that takes a low resolution mesh and a computed and interpolated (or rather approximated) copy of the mesh.

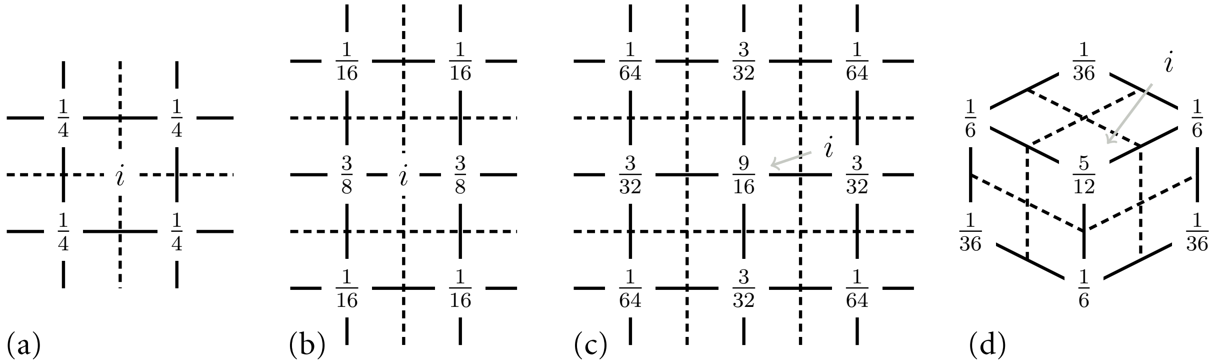


Figure 3.6: In this work, we use the Catmull-Clark subdivision scheme for meshes with quadrilateral faces [49]. The figures show the Catmull-Clark subdivision mask for inserting a face vertex (a) and edge vertex (b), and for regular (c) and irregular (d) vertices. For arbitrary polygonal meshes, a more general form of Catmull-Clark subdivision an be used [50].

Coarse-to-Fine Transition

For the coarse-to-fine transition we use Catmull-Clark [50] subdivision, an approximating subdivision scheme that creates smooth surfaces.

To interpolate low resolution images, usually bilinear interpolation is used. This interpolation method can also be used for shapes. However, bilinear interpolation applied to shapes leads to hard edges that look unnatural for smooth objects like human faces. Another possibility to interpolate the surface would be to use an interpolating subdivision scheme, like *Modified Butterfly Subdivision* [51]. Interpolating subdivision schemes do not change the position of a vertex, once a vertex exists in the mesh. Common for all interpolating subdivision schemes is that they may generate strange shapes with unnatural undulations [51].

To avoid these artifacts, we prefer an approximating subdivision scheme that creates smooth surfaces without unnatural undulations. Approximating subdivision schemes may change the position of a vertex that already exists in the mesh. Here we use one of the most popular subdivision schemes, the Catmull-Clark subdivision scheme for meshes with quadrilateral faces. It is based on tensor product bi-cubic uniform B-splines [49]. The limit surface obtained by repeated subdivision is called Catmull-Clark surface. It can be shown that this surface is at least C^1. The Catmull-Clark subdivision algorithm is defined recursively by applying the subdivision rules shown in Figure 3.6. Let $\mathbf{x}^l$ be a low resolution mesh on some level l. Catmull-Clark subdivision is a linear operation. To generate a mesh on a higher level $h = l+1$ we use $\uparrow_l^h \mathbf{x}^l$ where $\uparrow_l^h \in \mathbb{R}^{3m_h \times 3m_l}$ is the Catmull-Clark subdivision matrix that is defined by the subdivision rules shown in Figure 3.6, and m_h and m_l are the number of vertices in the mesh on level h and l, respectively. To simplify notation, the matrix $\uparrow_a^c := \uparrow_b^c \uparrow_a^b$ denotes the subdivision matrix to subdivide a mesh on level a into a mesh on level c. and the matrix $\uparrow_l := \uparrow_l^{l_{\max}}$ denotes

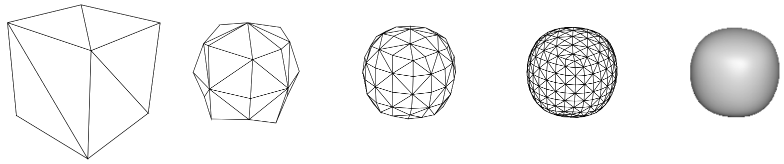


Figure 3.7: This figure shows the first three steps of Catmull-Clark subdivision of a cube and the Catmull-Clark subdivision surface.

subdivision to the maximal level $l_{\max}$ that is given by the maximal resolution of the template used for registration.

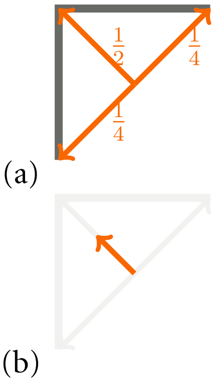
Fine-to-Coarse Transition

Beside the coarse-to-fine transition, we need to define a fine-to-coarse transition. The downsampled version of the mesh

$$\mathbf{x}^l := \downarrow_l^h \mathbf{x}^h \tag{3.16}$$

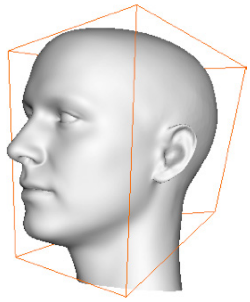
is generated from the high resolution mesh $\mathbf{x}^h$ on level h where $\downarrow_l^h \in \mathbb{R}^{3m_l \times 3m_h}$ is a matrix. This requires three steps:

- To avoid aliasing effects, the data is low-pass filtered by Gaussian smoothing. For meshes it is more appropriate to smooth the direction and the length of the vectors. When smoothing is applied to the x-, y- and z-coordinates of the vertices independently, the mesh tends to shrink, as illustrated in the margin: the middle vector of the surface (a) is filtered with Gaussian filter $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$. The result is the downsized vector in (b). This effect is undesired, the surface should be a coarse approximation and not a smaller version.



- Downsampling is performed using the regular structure of the mesh by removing every second vertex in parametrization. This is done in both the x and y directions of the parametrization, and as a result the mesh on level $l-1$ has approximately $1/4$ of the vertices of the mesh on level l.

- As illustrated in Figure 3.7, the volume of an object usually changes by Catmull-Clark subdivision. As a result, when applying the subdivision scheme to the downsampled mesh, the result is a downsized version of the face. The vertices of the low resolution mesh are control points that affect the shape and size of the Catmull-Clark surface. Hence, we want to find those control points that generate a mesh that is a coarse approximation of the input mesh while maintaining overall size. This is illustrated in the margin, where the (orange) cube is given by those control points, whose Catmull-Clark surface is similar to the surface of the face. Hence, the mesh $\mathbf{x}^l$ is not a coarse approximation of the input mesh, but $\mathbf{x}^l$ are the control points



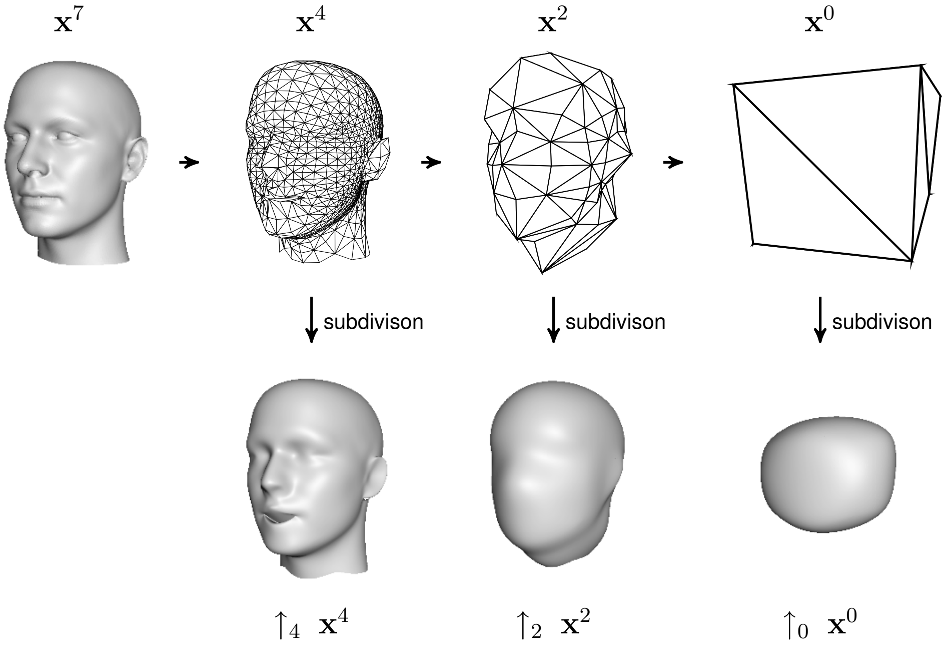


Figure 3.8:
As for images (see Figure 3.4), the Gaussian Pyramid for 3D shapes (shown in the first row) is computed by repeatedly applying a Gauss Filter and subsampling. In this figure, we also shown the Catmull-Clark surfaces that are generated by the low resolution meshes.

for Catmull-Clark subdivision. To find these control points, it is necessary to invert the subdivision matrix $\uparrow_l^h$. Inverting the subdivision matrix is an overconstrained problem, so this can only be approximated.

The resulting hierarchy of meshes, together with the resulting limit surfaces can be seen in Figure 3.8.

3.3.3 Filterbank

The mesh $\mathbf{x}^l, l < h$ contains fewer vertices than the mesh $\mathbf{x}^h$, so some details are lost. These details are captured by the residual between the mesh $\mathbf{x}^h$ and the subdivided copy of the mesh $\mathbf{x}^l$

$$\mathbf{y}^{l,h} = \mathbf{x}^h - \uparrow_l^h \mathbf{x}^l. \tag{3.17}$$

Analogously to the Laplacian Pyramid for images, the original mesh $\mathbf{x}^h$ can be recovered given the low resolution mesh $\mathbf{x}^l$ and the details, according to

$$\mathbf{x}^h = \uparrow_l^h \mathbf{x}^l + \mathbf{y}^{l,h} \in \mathbb{R}^{3m_h}. \tag{3.18}$$

This process can be continued recursively. For the G2L model, a filterbank with the four steps $s_4 = 7,\ s_3 = 4,\ s_2 = 2$ and $s_1 = 0$ is used (see Figure 3.9). This is a reasonable trade-off and having a model with too many levels is not helpful for

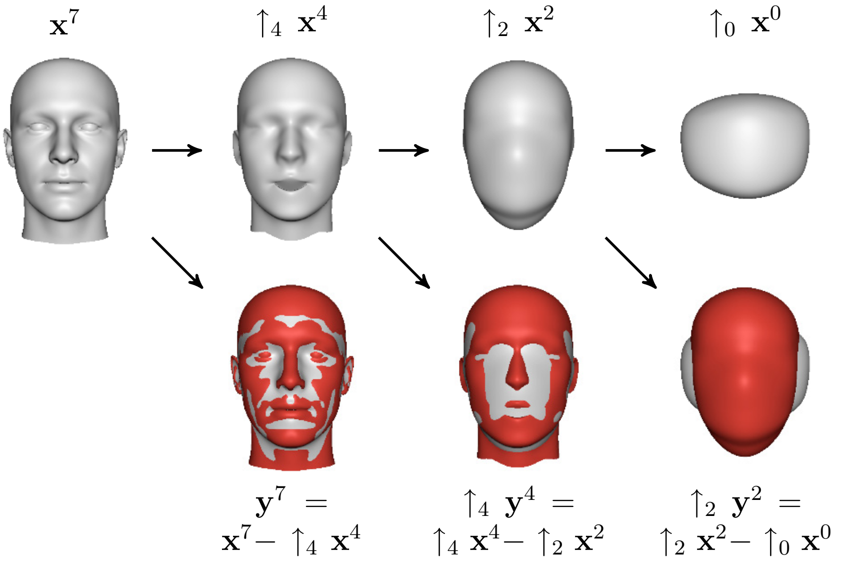

$\mathbf{x}^7$
$\uparrow_4 \mathbf{x}^4$
$\uparrow_2 \mathbf{x}^2$
$\uparrow_0 \mathbf{x}^0$
$\mathbf{y}^7 = \mathbf{x}^7 - \uparrow_4 \mathbf{x}^4$
$\uparrow_4 \mathbf{y}^4 = \uparrow_4 \mathbf{x}^4 - \uparrow_2 \mathbf{x}^2$
$\uparrow_2 \mathbf{y}^2 = \uparrow_2 \mathbf{x}^2 - \uparrow_0 \mathbf{x}^0$

Figure 3.9: As for images (see Figure 3.5), the Laplacian Pyramid for shapes is computed using the Gaussian Pyramid. This defines a filterbank of 3D shapes. The details in the second row ($\mathbf{y}^7$,$\uparrow_4 \mathbf{y}^4$ and $\uparrow_2 \mathbf{y}^2$) are the offset between the red and the white surface.

the statistical analysis. To simplify notation, the details between two consecutive levels are denoted by

$$\mathbf{y}^{s_\ell} = \mathbf{y}^{s_{\ell-1}, s_\ell}, \qquad \ell = 2, 3, 4 \tag{3.19}$$

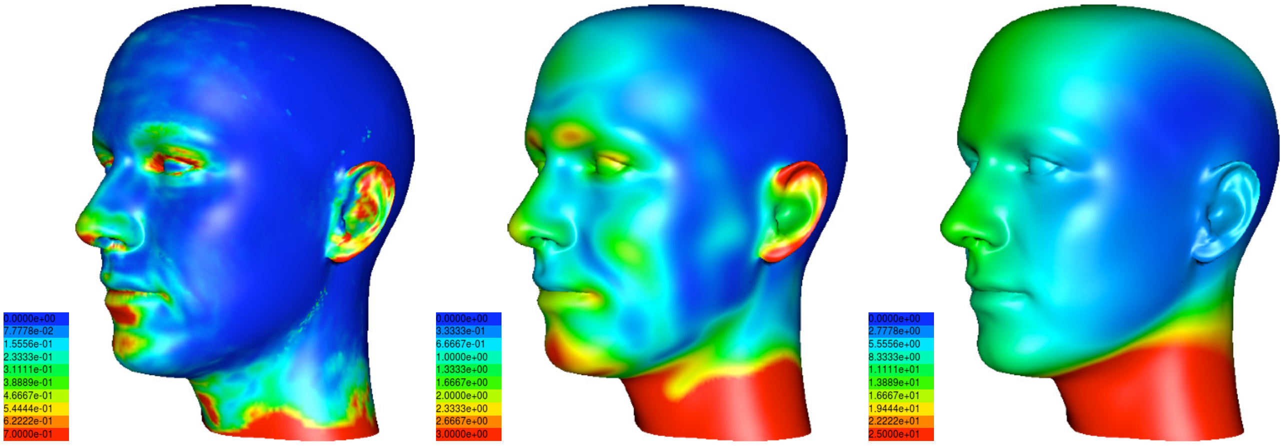
(ℓ denotes the level in the G2L model, $\ell = 1$ is the global level). By collapsing the pyramid (i.e. recombining all pyramid levels of the Laplacian pyramid), the original data

$$\mathbf{x} = \mathbf{x}^7 = \uparrow_0 \mathbf{x}^0 + \sum_{\ell \in \{2,3,4\}} \uparrow_{s_\ell} \mathbf{y}^{s_\ell} = \uparrow_0 \mathbf{x}^0 + \sum_{\ell \in \{2,3,4\}} \left(\uparrow_{s_\ell} \mathbf{x}^{s_\ell} - \uparrow_{s_{\ell-1}} \mathbf{x}^{s_{\ell-1}} \right) \tag{3.20}$$

is reconstructed. Turning a surface into this kind of Laplacian Pyramid and collapsing is a lossless transformation.

3.4 Segmentation of the Details

By the multiresolution analysis with the Laplacian Pyramid, the facial surface is broken up into components according to spatial frequency. But the detail components do not have a local support yet. They influence the whole face and not a specific feature. However, as with images, the Laplacian Pyramid removes correlation [47] and the details $\mathbf{y}^l$ are sparse: we observe that they cluster into regions, as can be seen in Figure 3.10. The reason for this is that in some regions (e.g. at the cheeks), there are usually no high frequency details. To achieve local support, the details are segmented into regions of facial features, such that the details in each



(a) level 4: $\mathrm{var}(\mathbf{y}^7)$ (b) level 3: $\mathrm{var}(\uparrow_4 \mathbf{y}^4)$ (c) level 2: $\mathrm{var}(\uparrow_2 \mathbf{y}^2)$

Figure 3.10: The variance of the detail components $\mathbf{y}^l$ for levels $l = 1, 2,$ and 3 is estimated from the training data set. For higher frequency components, the variance is sparse and clustered into regions.

segment are correlated with details in the same segment, but not (or as little as possible) with details in other segments. The segmentation considers the variance, i.e. the boundaries of the segments go through regions with small variance. In this way, components with local support are created, which independently model separate facial features.

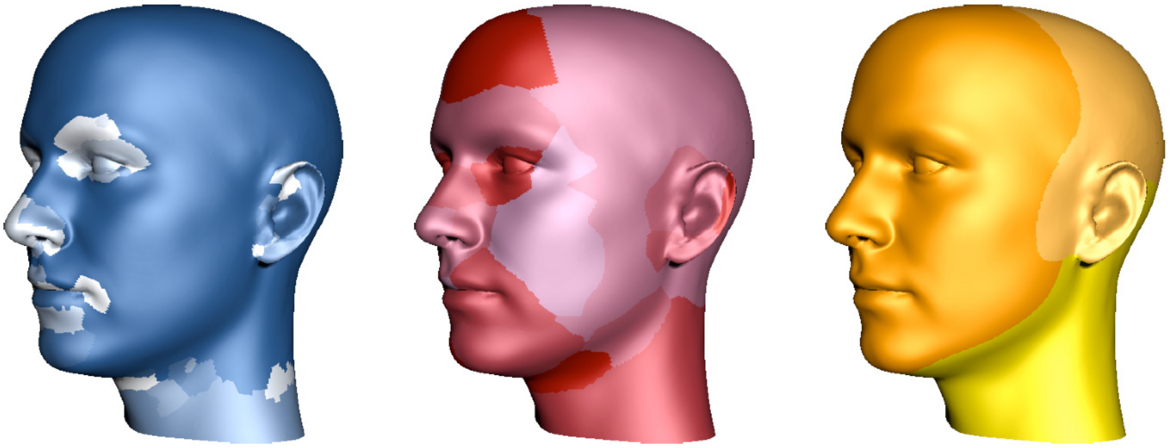
For each level, a different segmentation is used, and the size of the segments is related to the spatial frequency of the features, see Figure 3.11. Although, in principle, these segments could be defined manually, we determined them automatically using the procedure described in Appendix 7.2.

For the G2L model it is necessary to recombine the segments to a single shape, i.e. we meet the problem of blending different regions of the shape. To avoid hard edges at the boundary of the segments, blending of the segments is done using the Laplacian pyramids of shapes, in a similar way as the Laplacian Pyramids are used to blend multiple images [47]. The segments are blended across a transition zone proportional in size to the spatial frequency of the facial features represented in the segment. At the top level of the pyramid, a sharp blend mask is used, while at the bottom level a wide one is used. Thus, high frequency details are blended over a narrow region and low frequency details are blended over a wide region.

To segment the detail components $\mathbf{y}^l$, binary masks $\mathbf{m} \in \{0,1\}^{m^l}$ are used, where m^l is the number of vertices in $\mathbf{y}^l$. The mask $\mathbf{m}_j$ for segment j is a vector of binary indicator variables $m_{i,j} \in \{0,1\}$

$$\mathbf{m}_j = (m_{1,j}, \ldots, m_{m^l,j})^T \tag{3.21}$$

where $0 < j < \tau^l$ and τ^l is the number of segments for level l, such that $m_{i,j} = 1$



(a) level 4 (b) level 3 (c) level 2

Figure 3.11: To segment the high frequency details, different masks are used for the different levels. The segments are defined taking into account the variance shown in Figure 3.10.

if vertex i belongs to segment j and $m_{i,j} = 0$ otherwise. The set of masks for level l

$$M^l = \{\mathbf{m}_1^l, \ldots, \mathbf{m}_{\tau^l}^l\}, \tag{3.22}$$

describes a partition of the set of all vertices, i.e. $\sum_{j=1}^{\tau} m_{i,j} = 1;\ \forall i$. The masks for level $\ell = 2, 3$ and 4 are visualized in Figure 3.11.

The segment components

$$\mathbf{s_m} = \uparrow_l \mathrm{diag}(\mathbf{m})\mathbf{y}^l \tag{3.23}$$

are given by masking the detail components and subdividing up to the maximal level. The segments $\mathbf{s_m} \in \mathbb{R}^{3m}$ are all represented by vectors of the same length as the original mesh $\mathbf{x}$. However, the segments $\mathbf{s_m}$ have local support. By using the masks on a low resolution and subdividing the masked segments, the boundaries are smoothed with a transition zone that depends on the level. By summing up all segments

$$\sum_{\mathbf{m} \in M^l} \mathbf{s_m} = \sum_{\mathbf{m} \in M^l} \uparrow_l \mathrm{diag}(\mathbf{m})\mathbf{y}^l = \uparrow_l \sum_{\mathbf{m} \in M^l} \mathrm{diag}(\mathbf{m})\mathbf{y}^l = \uparrow_l \mathbf{y}^l \tag{3.24}$$

the detail components can be recovered.

3.5 Building a linear statistical model

The facial surface of a single face is broken up into features with local support. This can be done for all faces in the database. In the following, we describe how

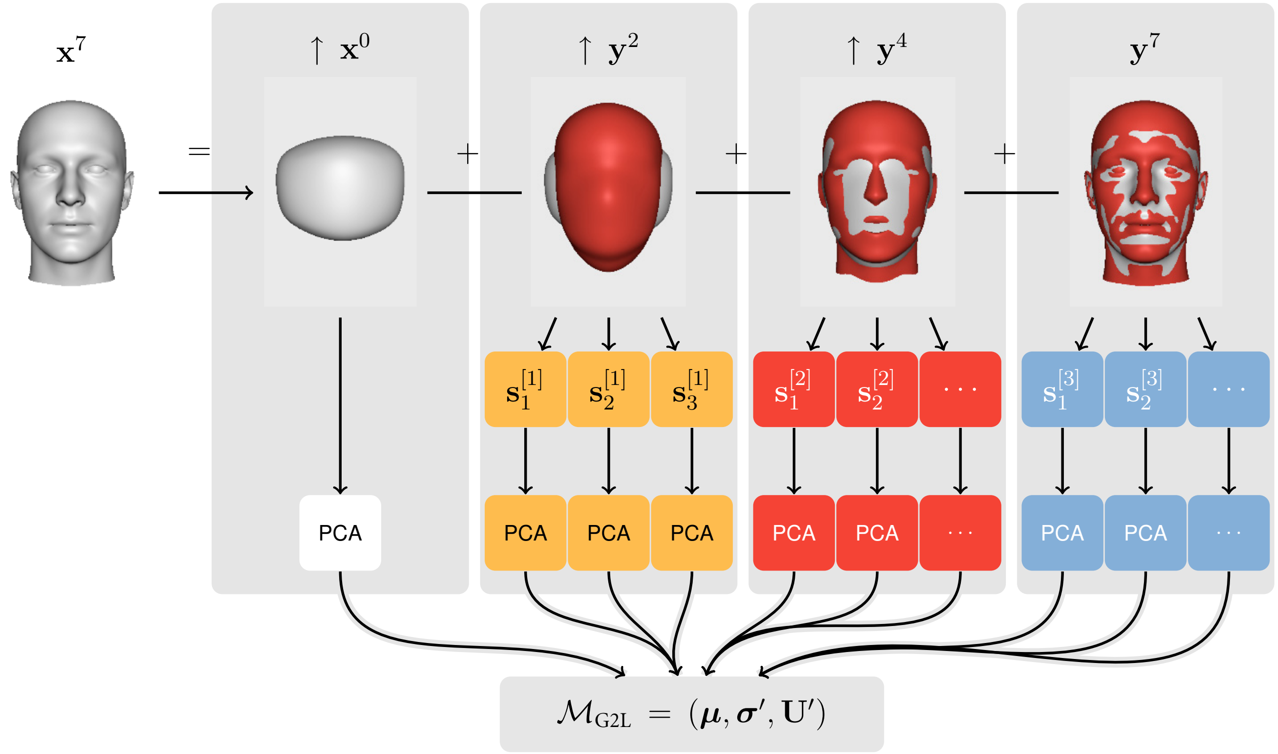


Figure 3.12: The figure illustrates the different steps needed to build the G2L model: The facial surface is broken up according to spatial frequency by multiresolution analysis. The high frequency details are segmented. For the global and local segments, individual PCA models are computed and the components are combined into a linear statistical model.

the global and local features are used to define the local and global components of a linear statistical model. In Figure 3.12 this process is visualized.

3.5.1 Global and Local Components

With the Gaussian pyramid of shapes, the facial surface $\mathbf{x}$ is split into a hierarchy of low-frequency features and high-frequency features (Equation 3.20):

$$\mathbf{x} = \uparrow_0 \mathbf{x}^0 + \sum_l \uparrow_l \mathbf{y}^l \tag{3.25}$$

The high-frequency features are segmented into multiple regions (Equation 3.24), i.e. each face $\mathbf{x}$ is split into global features and a set of local features

$$\mathbf{x} = \uparrow_0 \mathbf{x}^0 + \sum_l \sum_{\mathbf{m} \in M^l} \mathbf{s}_{\mathbf{m}}. \tag{3.26}$$

The segments $\mathbf{s}_{\mathbf{m}} \in \mathbb{R}^{3m}$ are all represented by vectors of the same length, see Equation 3.23. The number of global and local features is $\tau = 1 + \sum \tau^l$. These global and local features are combined to a feature vector

$$\mathbf{x}' := \begin{pmatrix} \mathbf{s}^0 \\ \mathbf{s}^1 \\ \vdots \\ \mathbf{s}^{\tau-1} \end{pmatrix} \in \mathbb{R}^{3m\tau} \tag{3.27}$$

where

$$\mathbf{s}^0 := \uparrow_0 \mathbf{x}_0 \in \mathbb{R}^{3m} \tag{3.28}$$

are the the global features and

$$\mathbf{s}^j := \mathbf{s}_{\mathbf{m}_j} \in \mathbb{R}^{3m} \qquad j > 0 \tag{3.29}$$

are the local features, and $\mathbf{m}_j \in M^l$ is one of the masks. The computation of the global and local features, i.e. the high-pass and low-pass filtering and segmentation can be written as a linear transformation

$$\mathbf{x}' = \mathbf{A}\mathbf{x}\,. \tag{3.30}$$

The original surface $\mathbf{x}$ can be recovered by summing up the global and local features, see Equation 3.26

$$\mathbf{x} = \mathbf{B}\mathbf{x}' = \sum_{j=1}^{\tau} \mathbf{s}^j\,, \tag{3.31}$$

where $\mathbf{B} = (1, 1, \ldots, 1) \otimes \mathbf{I}$.

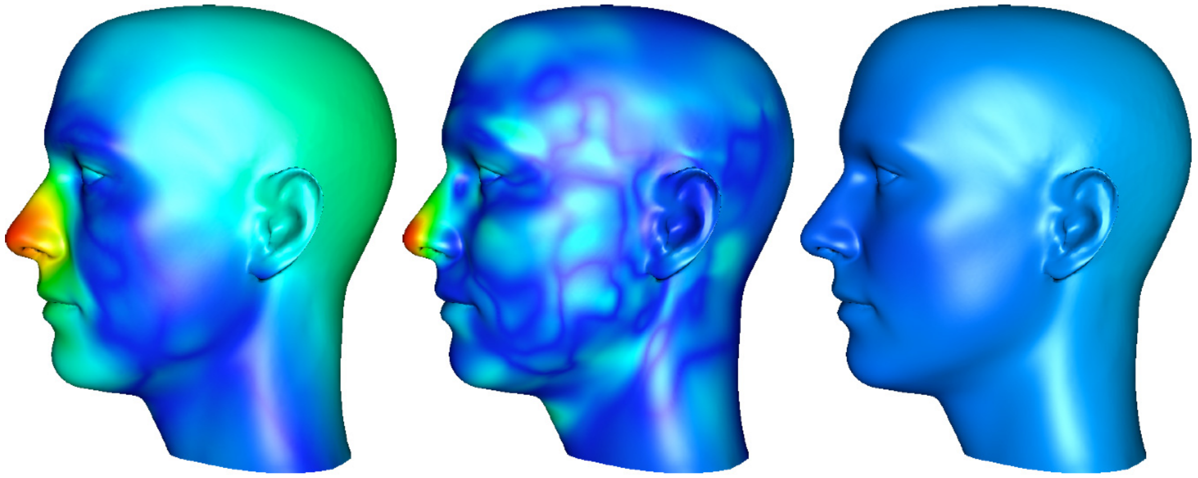
The mean-free feature vectors of the training set are stacked into the data matrix (analogously to Equation 3.6)

$$\mathbf{X}' := [\mathbf{x}'_1 - \boldsymbol{\mu}', \ldots, \mathbf{x}'_n - \boldsymbol{\mu}'] = \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{X}^2 \\ \vdots \\ \mathbf{X}^{\tau-1} \end{pmatrix} \tag{3.32}$$

where $\boldsymbol{\mu}' = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}'_i$ is the mean of the feature vectors. The sub-matrix $\mathbf{X}^j = [\mathbf{s}_1^j - \boldsymbol{\mu}^j, \ldots, \mathbf{s}_n^j - \boldsymbol{\mu}^j]$ is the mean-free data matrix for feature j and $\boldsymbol{\mu}^j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i^j$ is the corresponding the mean.

In particular, the mean of the data

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{B}\mathbf{x}'_i = \mathbf{B}\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}'_i = \mathbf{B}\boldsymbol{\mu}' = \sum_j \boldsymbol{\mu}_j \tag{3.33}$$



(a) $\mathrm{corr}(x_{8320}, \mathbf{x})$ (b) $\mathrm{corr}(\uparrow y^2_{8320}, \uparrow \mathbf{y}^2)$ (c) $\mathrm{corr}(\uparrow y^2_{8320}, \uparrow \mathbf{x}^0)$

Figure 3.13: This figure visualizes the correlation between the tip of the nose (vertex number 8320) and all other vertices. The correlation is estimated over the `UniBS-A` data set. Sown is the absolute value of correlation coefficient of the distance of each vertex from the mean, red indicates maximal value 1 and blue indicates no correlation. Figure (a) shows that for data $\mathbf{x}$ the tip of the nose is correlated also with the back of the head. Figure (b) shows the correlation of the detail features on level 2, where the correlation is mostly sparse and large only in a region surrounding the nose. Figure (c) shows that there is almost no correlation between detail features on level 2 the global features.

can be recovered from the mean of the features and the mean-free feature matrix can be written as

$$\begin{aligned} \mathbf{B}\mathbf{X}' &= \mathbf{B}[\mathbf{x}'_1 - \boldsymbol{\mu}', \ldots, \mathbf{x}'_n - \boldsymbol{\mu}'] && (3.34) \\ &= [\mathbf{B}\mathbf{x}'_1 - \mathbf{B}\boldsymbol{\mu}', \ldots, \mathbf{B}\mathbf{x}'_n - \mathbf{B}\boldsymbol{\mu}'] && (3.35) \\ &= [\mathbf{x}_1 - \boldsymbol{\mu}, \ldots, \mathbf{x}_n - \boldsymbol{\mu}] = \mathbf{X}. && (3.36) \end{aligned}$$

3.5.2 Covariance

The features are constructed in such a way that the covariance matrix exhibits a specific sparsity pattern: high resolution components are decorrelated from low resolution components and independent facial features within each resolution level are decorrelated from each other. To estimate the covariance matrix, we assume or rather enforce complete decorrelation, similar to [30]. That these assumptions are reasonable for the face model is illustrated in Figure 3.13, where the pairwise linear correlation coefficient between the tip of the nose and all other vertices is visualized. Figure 3.13(a) shows strong correlation for the data $\mathbf{x}$, while the correlation of the high-frequency features $\mathbf{y}^2$ is sparse (see Figure 3.13(b)) and the correlation between high-frequency features $\mathbf{y}^2$ and low-frequency features $\mathbf{x}^0$ is negligible (see Figure 3.13(c)).

For the PCA model, the covariance matrix

$$\mathbf{\Sigma} \;=\; \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathbf{T}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathbf{T}} \tag{3.37}$$

is diagonalized (Equation 3.9). In term of the global and local features, the covariance matrix can be written as

$$\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}\mathbf{X}^{T} = \frac{1}{n}\mathbf{B}\mathbf{X}'\mathbf{X}'^{T}\mathbf{B}^{T} \tag{3.38}$$

where $\mathbf{X}'$ is the mean-free feature matrix (Equation 3.32). To diagonalize the matrix $\mathbf{X}'\mathbf{X}'^{T}$, however, we can make use of the sparsity and approximate the matrix by

$$\frac{1}{n}\mathbf{X}'\mathbf{X}'^{T} \approx \frac{1}{n}\begin{bmatrix} \mathbf{X}^0\mathbf{X}^{0^T} & & & \\ & \mathbf{X}^1\mathbf{X}^{1^T} & & \\ & & \ddots & \\ & & & \mathbf{X}^{\tau}\mathbf{X}^{\tau^T} \end{bmatrix} =: \mathbf{\Sigma}' \,. \tag{3.39}$$

Even if the high and low resolution components are not completely decorrelated in the original data $\mathbf{X}'$, we assume or rather enforce complete decorrelation by using above approximation. Therefore, the SVD is computed individually on each block with $\mathbf{X}^j = \mathbf{U}^j\mathbf{W}^j\mathbf{V}^{j^T}$ and $\mathbf{\Lambda}^j = \frac{1}{n}\mathbf{W}^{j^2}$

$$\mathbf{\Sigma}' = \begin{bmatrix} \mathbf{U}^0\mathbf{\Lambda}^0\mathbf{U}^{0^T} & & & \\ & \mathbf{U}^1\mathbf{\Lambda}^1\mathbf{U}^{1^T} & & \\ & & \ddots & \\ & & & \mathbf{U}^{\tau}\mathbf{\Lambda}^{\tau}\mathbf{U}^{\tau^T} \end{bmatrix} \tag{3.40}$$

$$= \begin{bmatrix} \mathbf{U}^0 & & \\ & \ddots & \\ & & \mathbf{U}^{\tau} \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda}^0 & & \\ & \ddots & \\ & & \mathbf{\Lambda}^{\tau} \end{bmatrix} \begin{bmatrix} \mathbf{U}^0 & & \\ & \ddots & \\ & & \mathbf{U}^{\tau} \end{bmatrix}^T . \tag{3.41}$$

Hence, the covariance matrix is approximately diagonalized by

$$\mathbf{\Sigma} = \mathbf{B}\mathbf{X}'\mathbf{X}'^{T}\mathbf{B}^{T} \approx [\mathbf{U}^0, \mathbf{U}^1, \cdots, \mathbf{U}^{\tau}]\mathbf{\Lambda}'[\mathbf{U}^0, \mathbf{U}^1, \cdots, \mathbf{U}^{\tau}]^T \tag{3.42}$$

$$= \mathbf{U}'\mathbf{\Lambda}'\mathbf{U}'^{T} \tag{3.43}$$

where

$$\mathbf{U}' := [\mathbf{U}^0, \mathbf{U}^1, \cdots, \mathbf{U}^{\tau}] \tag{3.44}$$

is the combined matrix of all components. The matrix

$$\mathbf{\Lambda}' = \mathrm{diag}(\mathbf{\Lambda}^0, \mathbf{\Lambda}^1, \cdots, \mathbf{\Lambda}^\tau) \tag{3.45}$$

is the diagonal matrix of all eigenvalues. The difference to the estimation of the covariance in Equation 3.37 is that most of the components in $\mathbf{U}'$ have local support, since $\mathbf{U}^i, i > 1$ is computed from segments s_m with local support. The matrix $\mathbf{U}'$ consist of $\tau \cdot (n-1)$ components, since it is composed from τ models, and each model has $n-1$ components.

PCA minimizes the projection error of the samples to a lower dimensional subspace, i.e. it minimizes the deformation introduced by using a restricted number $n' < n - 1$ of components, for more details see [52]. For this reason, for PCA models, the number of components can be reduced by using the n' principle components with the largest eigenvalues λ_i, which by convention are the first n' components, see Figure 3.14.

For the G2L model, we use a similar approach. For each component $\mathbf{u}'_i \in \mathbb{R}^{3m}$ from $\mathbf{U}'$, the corresponding value of deformation measure is given by λ'_i. Hence, we use the component $\mathbf{u}'_i$ with the largest λ'_i over all global and local components. In Figure 3.14 the first 400 eigenvalues are plotted.

Together with the mean $\boldsymbol{\mu}$, the component matrix $\mathbf{U}'$ and the the corresponding standard deviation $\sigma'_i = w'_i/\sqrt{n}$ define a *linear generative statistical* face model (Equation 3.1)

$$\mathcal{M}_{\mathrm{G2L}} = (\boldsymbol{\mu}, \boldsymbol{\sigma}', \mathbf{U}') \,. \tag{3.46}$$

Analogously to Equation 3.11, the probability distribution is given by

$$p(\mathbf{x}) = \nu \exp(-\frac{1}{2}\|\boldsymbol{\alpha}'\|^2) \tag{3.47}$$

where $\boldsymbol{\alpha}'$ are the coefficients of the G2L model (and ν is a constant factor).

3.6 Regularization

The main application of statistical face models (Equation 3.1) is their fitting to incomplete data. The model is used as prior knowledge: the model coefficients are estimated and the complete reconstructed surface is generated by the model. However, not every arbitrary vector of coefficients encodes a valid face, the coefficients have to be distributed according to a normal distribution (Equation 3.1). In the following, we will use a Bayesian approach for reconstruction [36], where from the prior probability of the model and prior assumption of Gaussian noise on the measurement a combined cost function is derived. Let $\tilde{\mathbf{r}}$ be a (incomplete)

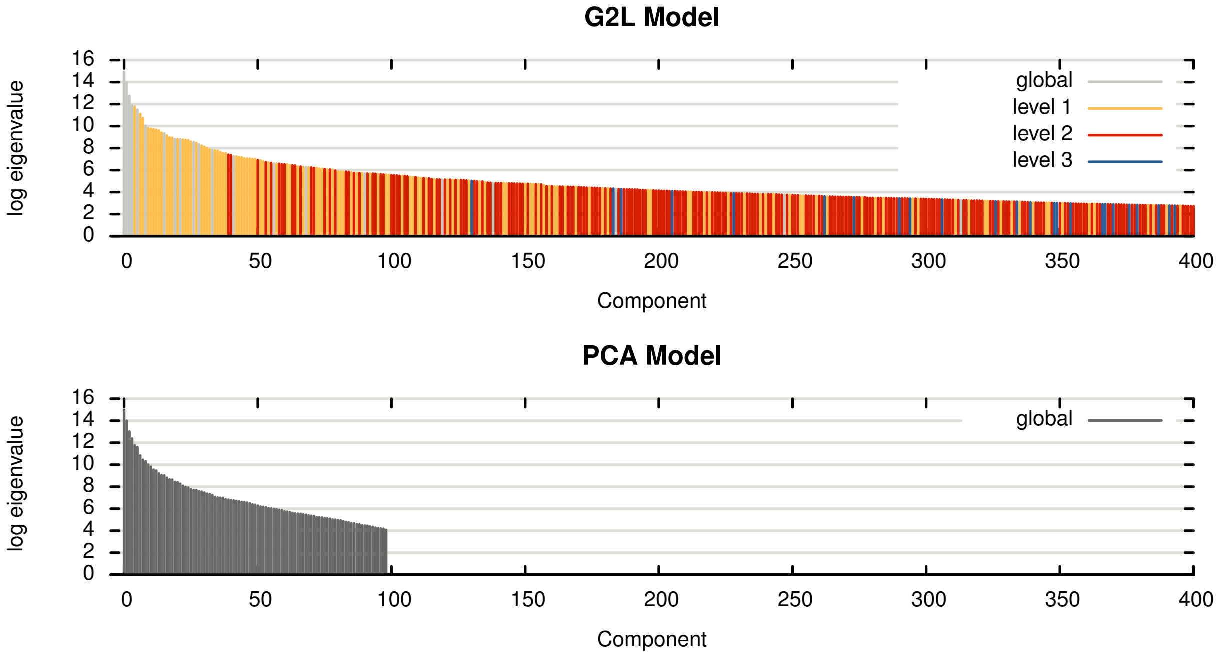


Figure 3.14: For the `UniBS-A` data, the first 400 eigenvalues of G2L model (top) are compared to the eigenvalues of PCA model (bottom). For the G2L model, the global and local components on level 1,2, and 3 are mixed and sorted according to their eigenvalues. The colors indicate the level in the hierarchy of the local components. The more local components with higher frequency are less important. The PCA model is restricted to 99 principle components, since the `UniBS-A` data set contains 100 scans; the G2L model has more components and offers more flexibility.

measurement.[1] Let $\mathbf{r}(\boldsymbol{\alpha})$ be the corresponding noiseless measurement that is generated by the model through the parameters $\boldsymbol{\alpha}$. When the measurement consists of the 3D coordinates of a vertex j, then

$$\vec{r}_i(\boldsymbol{\alpha}) = \vec{x}_j(\boldsymbol{\alpha}) = \left[\mathbf{U}\ \mathrm{diag}(\sigma_1, \ldots, \sigma_{n'})\ \boldsymbol{\alpha}\right]_j \tag{3.48}$$

is the noiseless measure generated by the model. However, depending on the type of measurement, the transformation $\vec{r}_i$ may also contain a perspective projection or shading calculation. It processes the vertex j such that it can be compared with the corresponding measurement $\tilde{r}_i$.

The residual between the measurement $\tilde{\mathbf{r}}$ and the model $\mathbf{r}(\boldsymbol{\alpha})$

$$F(\boldsymbol{\alpha}, \tilde{\mathbf{r}}) = \sum_i f_i(\boldsymbol{\alpha}, \tilde{r}_i) \tag{3.49}$$

---

[1] Statistical face model can be fitted to photographs or other types of incomplete data, such as a sparse set of 3D vertices, 2D landmark points or contour edges extracted from an image. Therefore, $\tilde{\mathbf{r}}$ can be different types of measurement, e.g. the 2D or 3D position of vertices or pixel color intensities.

is given as the sum over residual per vertex

$$f_i(\boldsymbol{\alpha}, \tilde{r}_i) = \|\tilde{r}_i - \vec{r}_i(\boldsymbol{\alpha})\|^2 \tag{3.50}$$

where $\tilde{r}_i$, $\vec{r}_i(\boldsymbol{\alpha})$ are the ith entries of the measurement $\tilde{\mathbf{r}}$ and the noiseless measurement $\mathbf{r}$, respectively.

For each dimension, we assume that the measurement $\tilde{\mathbf{r}}$ is subject to uncorrelated Gaussian noise with standard deviation $\tilde{\sigma}$. The likelihood to measure $\tilde{\mathbf{r}}$ given the model (see [36]) is given by

$$\begin{aligned}
P(\tilde{\mathbf{r}} \mid \boldsymbol{\alpha}) &= \prod P(\tilde{r}_i \mid \vec{r}_i(\boldsymbol{\alpha})) && (3.51)\\
&= \prod \nu_r \exp\left(-\frac{1}{2\tilde{\sigma}^2} f_i(\boldsymbol{\alpha}, \tilde{r}_i(\boldsymbol{\alpha}))\right) && (3.52)\\
&= \nu_r \exp\left(-\frac{1}{2\tilde{\sigma}^2} \sum f_i(\boldsymbol{\alpha}, \tilde{r}_i(\boldsymbol{\alpha}))\right) && (3.53)\\
&= \nu_r \exp\left(-\frac{1}{2\tilde{\sigma}^2} F(\boldsymbol{\alpha}, \tilde{\mathbf{r}})\right), && (3.54)
\end{aligned}$$

where ν_r is a constant factor. Given the measurement $\tilde{\mathbf{r}}$, the posterior probability is given by (using Bayes rule)

$$\begin{aligned}
P(\boldsymbol{\alpha} \mid \tilde{\mathbf{r}}) &= \nu' p(\tilde{\mathbf{r}} \mid \boldsymbol{\alpha}) p(\boldsymbol{\alpha}) && (3.55)\\
&= \nu' \nu_r e^{-\frac{1}{2\tilde{\sigma}^2} F(\boldsymbol{\alpha}, \tilde{\mathbf{r}})} \nu e^{-\frac{1}{2}\|\boldsymbol{\alpha}\|^2} && (3.56)
\end{aligned}$$

where ν' is another constant factor and $p(\boldsymbol{\alpha})$ is the prior probability of the model (Equation 3.11). Thus, the coefficients with maximal posterior probability are found by minimizing the cost function

$$F = -2\tilde{\sigma}^2 \log P(\boldsymbol{\alpha} \mid \tilde{\mathbf{r}}) = F(\boldsymbol{\alpha}, \tilde{\mathbf{r}}) + \eta\|\boldsymbol{\alpha}\|^2 + c \tag{3.57}$$

where $\eta = \tilde{\sigma}^2$ is the regularization parameter and c is a constant.

Chapter 4

Evaluation

Fitting the model to face images, i.e. reconstructing the 3D surface of a face from a single photograph, is the most important application of statistical shape models. However, evaluating these reconstructions allows only limited conclusions about the quality of the model. It is unclear whether artifacts or fails are caused by deficiencies of the model, deficiencies of the fitting method, or deficiencies of the evaluation method. For example, the edges are usually not fitted well with the PCA model and it is difficult to determine whether the model is too inflexible to be able to fit the edges or if the optimizer gets stuck in a local minimum. Additionally, it is difficult to judge the quality of fitting results; the l^2-norm (or Euclidean norm) between the original surface and the reconstruction is of very limited use here. For this reason, in [14] the fitting is evaluated by performing recognition experiments and measuring the recognition rates. This, however, makes it even more difficult to judge the quality of the model. To overcome these problems, a much more controlled environment is used in this chapter. In this controlled environment, the correspondence between the measured vertices and the vertices in the model is known. In particular, all fittings in this chapter are computed by a closed-from expression and not by an iterative optimization algorithm. Therefore, we can be sure that the global minimum is found and we do not have to worry if the fitting gets stuck in a local optimum.

In order to evaluate the quality of a model, two points need to be considered: *representation* and *generalization*. Firstly, we compare how well novel faces can be represented with PCA and G2L models. This can be seen as the difference $\mathcal{E}_A$ (approximation error, see Figure 4.1) between the novel face and the best approximation in the space spanned by the models.

But we cannot expect to find this best approximation when only partial information is available. So, in a second experiment, we evaluate how well the models generalize when the surface of a novel face is reconstructed from a sparse set of feature points. Usually, this gives an additional difference, the sample error, $\mathcal{E}_S$. Re-

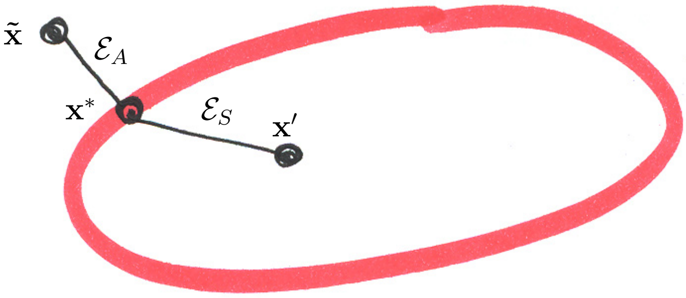


Figure 4.1: To evaluate the quality of statistical models, two types of errors need to be considered: the *approximation error* $\mathcal{E}_A$ and the *sample error* $\mathcal{E}_S$. The red line visualizes the space of all faces that can be represented by the model. The approximation error is the difference between a novel face $\tilde{\mathbf{x}}$ and the best approximation $\mathbf{x}^*$ within the model. It is zero when the novel face is already in the model space, which is generally not the case. When a face is reconstructed from partial information, usually not the best approximation $\mathbf{x}^*$ is found but a different face $\mathbf{x}'$. The difference between $\mathbf{x}^*$ and $\mathbf{x}'$ is the sample error. By making the model space larger, $\mathcal{E}_A$ can be reduced, however, this may happen at the cost of a larger $\mathcal{E}_S$.

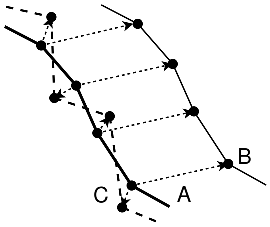
construction from a sparse set of feature points is similar to fitting a curve through some observed values, which usually calls for regularization or a restriction on the admissible function. Here, we restrict the possible solutions to the space spanned by the model.

4.1 Distance Measures

To evaluate the quality of a reconstruction (with respect to its original surface scan), we need an appropriate distance measure to compare two 3D surfaces. The sum of the Euclidean norm (or l^2-norm) over all vertices of the shapes is the most obvious choice

$$d(\mathbf{x}_i, \mathbf{x}_j)_{l^2} = \frac{1}{m} \sum_{k=1}^{m} \|\vec{x}_{k,i} - \vec{x}_{k,j}\|^2 \tag{4.1}$$

where $\vec{x}_{k,i}$ and $\vec{x}_{k,j}$ are corresponding vertices of the two surfaces. However, Euclidean distance treats all vertices independently and does therefore not detect visually unpleasing effects such as oscillations or random noise. As one can see in the illustration in the margin, optimizing the Euclidean distance favors near but noisy surfaces over distant but similar ones. Comparing reconstructed surfaces B and C with the original surface A, the Euclidean distance between B and A is much larger than between C and A, although C appears quite different from A and we would prefer B as a reconstruction result.



A: original surface
B: large Euclidean distance
C: small Euclidean distance

Accordingly, a small Euclidean distance is necessary but not sufficient for a good surface reconstruction and the visual quality of a reconstructed surface depends not only on the Euclidean distance. This is visible in the rendered surfaces, e.g. in Figure 4.2 where the PCA190 heads have a smaller Euclidean distance than the PCA30 heads, see Figure 4.3, left plot, but we would clearly consider the PCA30 head a better reconstruction result. To arrive at a comparison method which is closer to our intuitive perception of the quality of a reconstruction result, we propose to compare the *Mean Curvature* of the surfaces,

$$d(\mathbf{x}_i, \mathbf{x}_j)_{mc} = \frac{1}{m} \sum_{k=1}^{m} |H(k)_i - H(k)_j| \tag{4.2}$$

where H is the mean curvature. This measure can be computed easily due to the parametrization (see Section 2.4.1) and it is sensitive to effects such as noise, oscillations, and other artifacts. Moreover, it is a neutral measure, since the mean curvature is not minimized in the cost function. Nevertheless (or for that reason), this measure is not only able to detect synthetic oscillations, but also appropriate to detect overfitting, which is the main problem while fitting.

Other possibilities are to compare the *Shape Index* [53] or the angle between the normals of the faces. Here, we use the mean curvature distance, but the shape index distance and the normal distance showed similar results. As can be seen in Figure 4.3, right plot, the mean curvature distance increases for noisy and ripply surfaces.

4.2 Projection to the Face Space

The first test setting focuses on how well novel faces can be represented by the different models. Since the faces in the test set are not used for training, they are not in the face space spanned by the models. As a result, the test faces can only be approximated by the models and there will always be an *approximation error*, see Figure 4.1,

$$\mathcal{E}_A = d(\tilde{\mathbf{x}}, \mathbf{x}^*) \tag{4.3}$$

where $\tilde{\mathbf{x}}$ is a face of the test set and $\mathbf{x}^*$ is its optimal approximation. The optimal approximation minimizes the l^2-approximation error $\mathcal{E}_A$, i.e. it minimizes the distance

$$F^{\text{proj}} = \sum_{i=1}^{m} \|\tilde{x}_i - \vec{x}_i(\boldsymbol{\alpha})\|^2 \tag{4.4}$$

between all measured vertices $\tilde{x}_i$ and all model vertices

$$\vec{x}_i(\boldsymbol{\alpha}) = [\mathbf{U}\ \mathrm{diag}(\sigma_1, \ldots, \sigma_{n'})\ \boldsymbol{\alpha}]_i \tag{4.5}$$

In case of the PCA-model, the columns of $\mathbf{U}$ are orthogonal and the coefficients can be computed by projection (see Equation 3.10). For the G2L model, a linear equation system has to be solved. Examples of projections can be seen in Figure 4.2, and results can be found in Figure 4.3 and Figure 4.6.

4.3 Reconstruction from a Sparse Set of Points

The second test setting focuses on the generalization capabilities of the models. The surfaces in the test set are estimated from a small set of m' 3D feature points. For each feature point, the correspondence is known and $\mathcal{S}^{\text{pnts}}$ denotes the set of indices. In this case, we cannot expect to find the optimal approximation $\mathbf{x}^*$ of the test sample $\tilde{\mathbf{x}}$. Instead, we can only find a different surface $\mathbf{x}' \neq \mathbf{x}^*$, with an estimation error (also called sample error), see Figure 4.1

$$\mathcal{E}_S = d(\mathbf{x}^*, \mathbf{x}') \; . \tag{4.6}$$

To fit the models to the points, the difference between the measured landmark vertices $\tilde{x}_i$ and the corresponding model vertices $\vec{x}_i(\boldsymbol{\alpha})$

$$F^{\text{pnts}} = \sum_{i \in S^{\text{pnts}}} \|\tilde{x}_i - \vec{x}_i(\boldsymbol{\alpha})\|^2 \tag{4.7}$$

is minimized (compare with Equation 3.49 and Equation 3.50). However, minimizing this distance without considering the model probability leads to overfitting. This can be avoided by using a Bayesian approach (see Section 3.6). Under the assumption that the measurement of the feature points is subject to uncorrelated Gaussian noise and using the probability density of the model as a prior knowledge, the cost function is (see Equation 3.57)

$$F = F^{\text{pnts}} + \eta \|\boldsymbol{\alpha}\|^2 \; . \tag{4.8}$$

This reconstruction implicates a trade-off between accuracy of the measured feature points and likelihood of the reconstructed surface being a face. This trade-off is controlled by the regularization parameter $\eta = \tilde{\sigma}^2$ (Equation 3.57). In [36] it is shown that the coefficients can be obtained by

$$\boldsymbol{\alpha} = \mathbf{V}_q \text{diag}\left(\frac{w'_i}{w'^2_i + \eta}\right) \mathbf{U}_q^T \tilde{\mathbf{r}} \tag{4.9}$$

where $\mathbf{U}_q \text{diag}(w'_i) \mathbf{V}_q^T = \mathbf{Q}$ is the singular value decomposition of the reduced and scaled model components

$$\mathbf{Q} = \mathbf{R}\mathbf{U}\text{diag}(\sigma_1, \ldots, \sigma_{n'}) \tag{4.10}$$

and $\mathbf{R} \in \mathbb{R}^{m' \times m}$ is the matrix that selects the vertices in S^{pnts} from the vector $\mathbf{x}$. The measurement $\tilde{\mathbf{r}} = [\tilde{x}_1, \ldots, \tilde{x}_{m'}] - \mathbf{R}\mu$ is given by the mean-free landmark vertices.

4.4 Experiments

To evaluate and compare the PCA and the G2L models, we have done two experiments: Experiment A uses the `MPI-TÜ-C` data set and Experiment B uses the `UniBS-A` data set (see Table 2.1). For both experiments, we compare the representation and generalization capabilities of the G2L model with the PCA model.

4.4.1 Experiment A

Data

Both the G2L and the PCA model are trained from the same training set of 198 scans of faces. These faces are scanned with a Cyberware Scanner and registered as described in [40] (this is the data `MPI-TÜ-C`, see Table 2.1). As test set, a set of 20 additional scans (10 male / 10 female) is used. These faces are scanned with a structured light scanner and registered with the same method as the rest of the scans.

Projection

When projecting the test data into the PCA model, severe artifacts appear when all or almost all PCA components are used, see Figure 4.2 for an example. With the same number of components, the G2L model reconstructions have much less artifacts. The G2L model reconstructions can be further improved by using more components than is possible for the PCA model. Interestingly, when we compare the representation error (i.e. the distance between reconstruction and its ground truth), the G2L model reconstructions are better than the PCA reconstructions, see left plot in Figure 4.3. As expected, the mean curvature distance increases when more PCA components are used (see right plot). For the G2L model, however, the mean curvature distance does not increase, it even decreases.

Generalization

Reconstructing the surface from a few 3D landmark points involves a trade-off between accuracy of the fit at those points and likelihood of the fitting result within the model. This trade-off is controlled with a regularization parameter $\tilde{\sigma}$ (see Equation 4.8). In Figure 4.4 the reconstruction from 42 landmark points is shown for varying regularization. Without regularization, the PCA model reconstructions suffer from severe overfitting, with strong regularization, the mean face is reconstructed. The G2L model reconstructions, on the other hand, do not suffer from overfitting. At the same time, the G2L model accomplishes much better accuracy at the landmark points, see plot in Figure 4.5. In these plots, a PCA

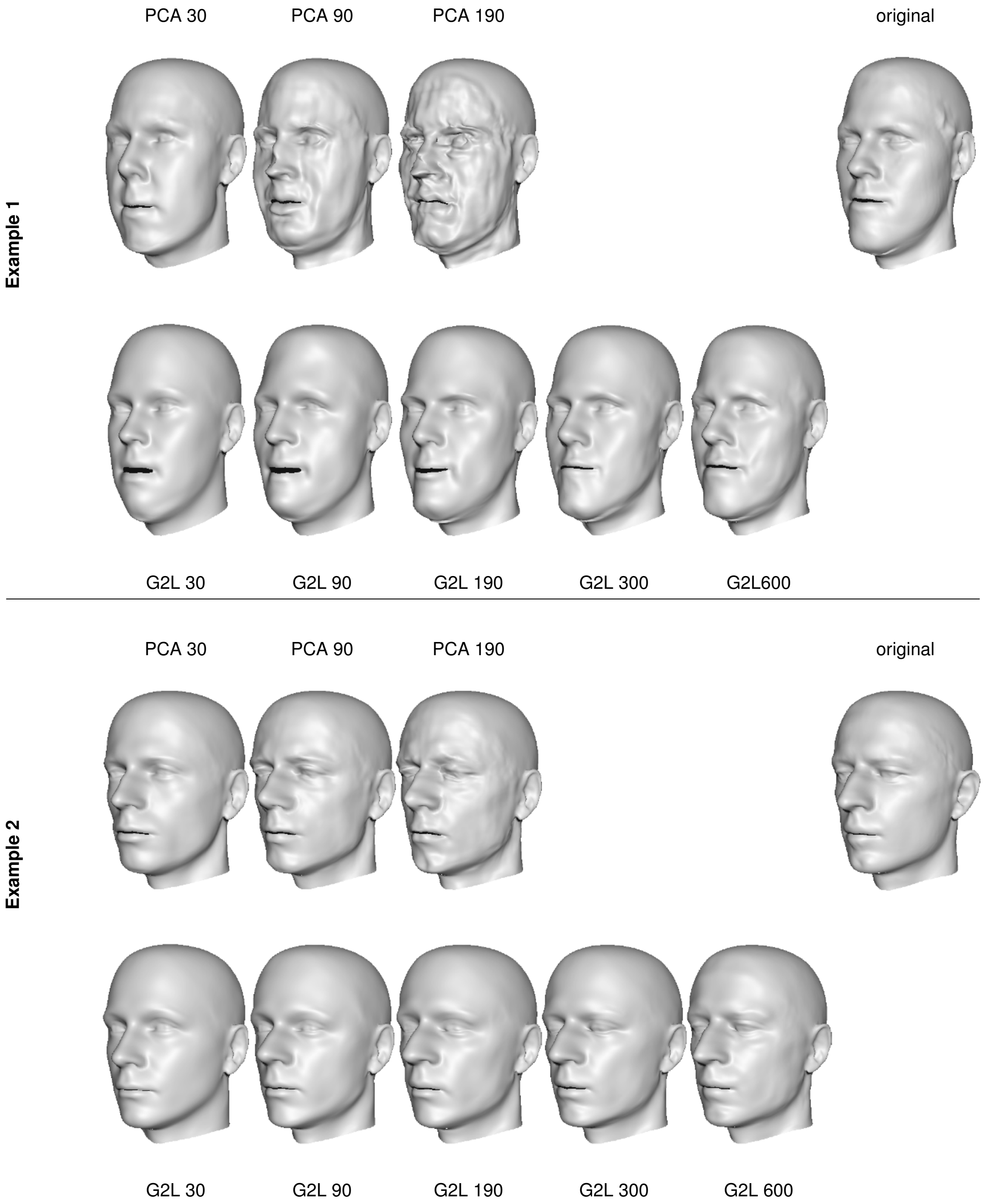


Figure 4.2: Experiment A (`MPI-TÜ-C` data set): Projection of two novel examples (from the test set) to PCA model (first row) and to G2L model (second row) with different number of components. The Euclidean distance and the mean curvature distance evaluated over the test set is plotted in Figure 4.3. For the PCA model, the surface gets distorted using too many coefficients, even though the Euclidean distance decreases.

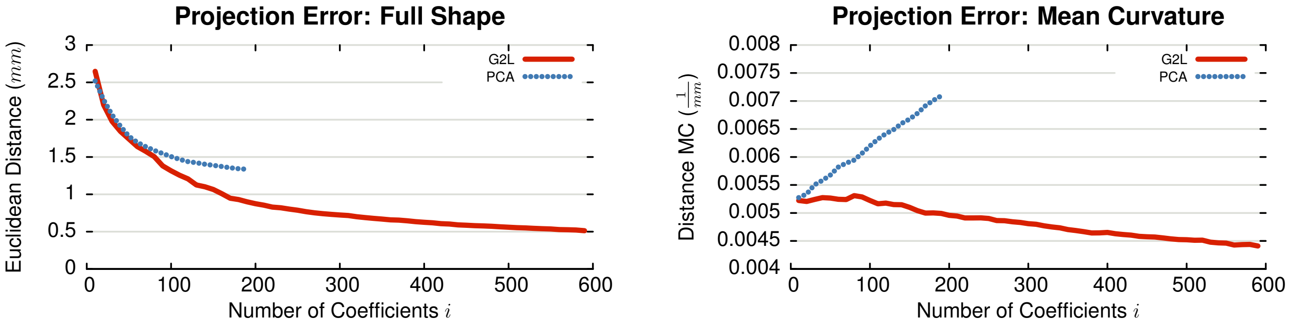


Figure 4.3: Experiment A (`MPI-TÜ-C` data set): Projection of novel examples (from the test set) to PCA model (blue dotted line) and to G2L model (red line) with respect to different number of components. The plot shows the average over a test set of 20 examples. Example of these projections can be found in Figure 4.2. On this test set, the G2L model has a smaller Euclidean distance, even with the same number of coefficients. For the PCA model projections, the mean curvature distance increases using more components, since the surface gets distorted.

model with 190 components (the size of the training set is 198) is compared with a G2L model with also 190 components and with a G2L model with 600 components. The minimal residual that could be achieved are $0.381\,mm$ with the PCA model, compared to $0.045\,mm$ with the G2L model (600 components) and $0.321\,mm$ with the G2L model (190 components). For a reasonable regularization (e.g. $\tilde{\sigma} = 10^{\frac{1}{2}}$) not only the accuracy at the landmark points is much better with the G2L model, but also the approximation error (i.e. the distance between reconstruction and its ground truth) is much smaller, both in l^2-distance as well as in mean curvature distance. As it has already been shown for the projection (see Figure 4.3, right plot), the mean curvature distance does not improve significantly compared with the mean curvature distance of a reconstruction with strong regularization. However, the mean curvature distance is important to detect overfitting. For $\tilde{\sigma} < 10^0\,mm$ the PCA model shows severe overfitting, while the G2L model suffers much less from overfitting.

Using a G2L model with the same number of components as are available for the PCA model, we have a smaller generalization error and better accuracy at the landmark vertices. However, from the same data, we can compute a G2L model with more components, what further improves generalization and approximation error.

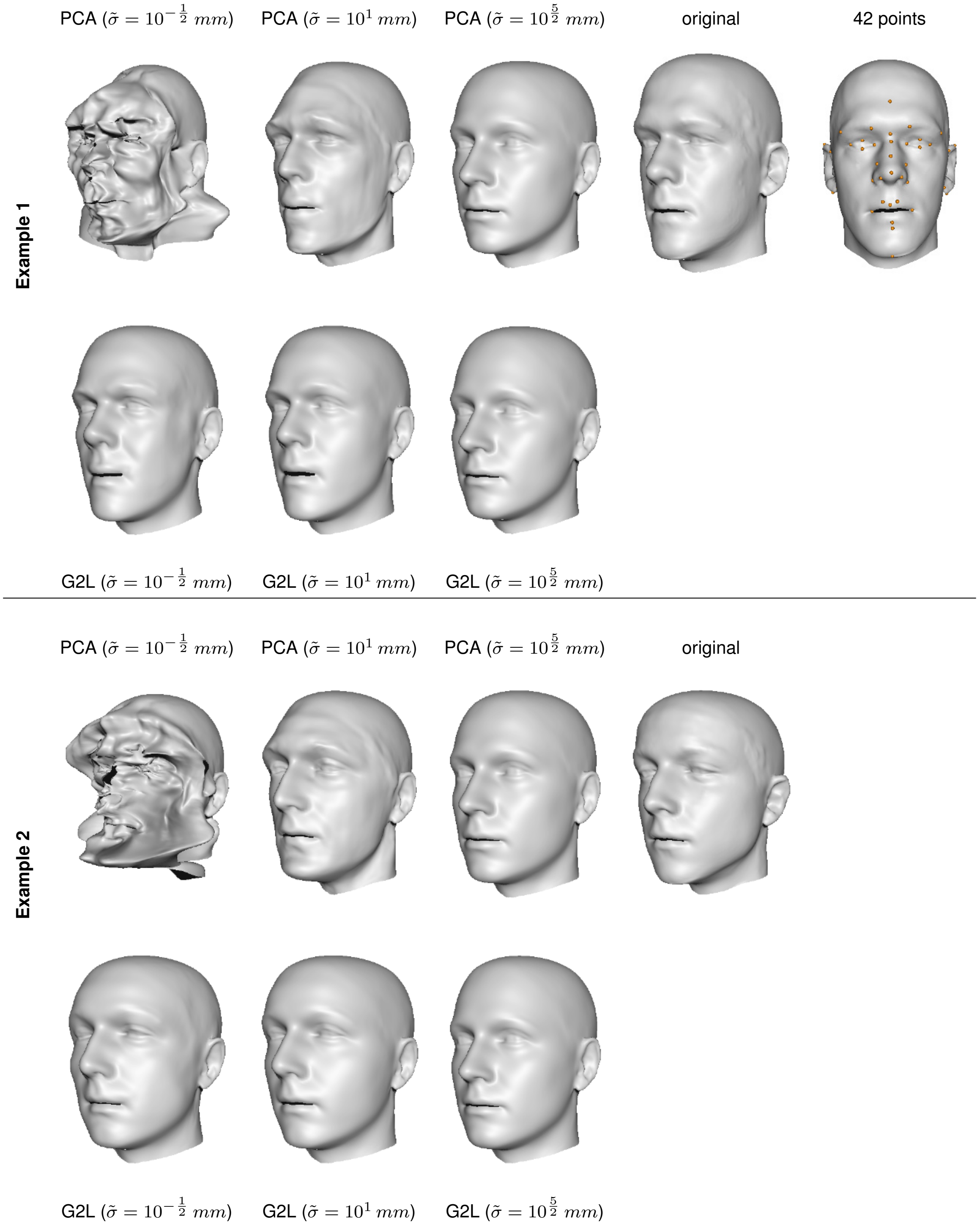

PCA ($\tilde{\sigma} = 10^{-\frac{1}{2}}\,mm$)
PCA ($\tilde{\sigma} = 10^{1}\,mm$)
PCA ($\tilde{\sigma} = 10^{\frac{5}{2}}\,mm$)
original
42 points
Example 1
G2L ($\tilde{\sigma} = 10^{-\frac{1}{2}}\,mm$)
G2L ($\tilde{\sigma} = 10^{1}\,mm$)
G2L ($\tilde{\sigma} = 10^{\frac{5}{2}}\,mm$)
PCA ($\tilde{\sigma} = 10^{-\frac{1}{2}}\,mm$)
PCA ($\tilde{\sigma} = 10^{1}\,mm$)
PCA ($\tilde{\sigma} = 10^{\frac{5}{2}}\,mm$)
original
Example 2
G2L ($\tilde{\sigma} = 10^{-\frac{1}{2}}\,mm$)
G2L ($\tilde{\sigma} = 10^{1}\,mm$)
G2L ($\tilde{\sigma} = 10^{\frac{5}{2}}\,mm$)

Figure 4.4: Experiment A (`MPI-TÜ-C` data set): Generalization from 42 landmark points ($1st$ row, right most image): Two examples of the test set are reconstructed using PCA model (first row) and G2L model (second row) with different regularization parameter $\eta = \tilde{\sigma}^2$. The parameter $\tilde{\sigma}$ is chosen within a reasonable range, where both, the G2L model and the PCA model reconstructions have their optimum, see the plots of the similar experiment (using 100 vertices) in Figure 4.5. The PCA reconstructions suffer from severe overfitting, while the G2L reconstructs the surfaces much better.

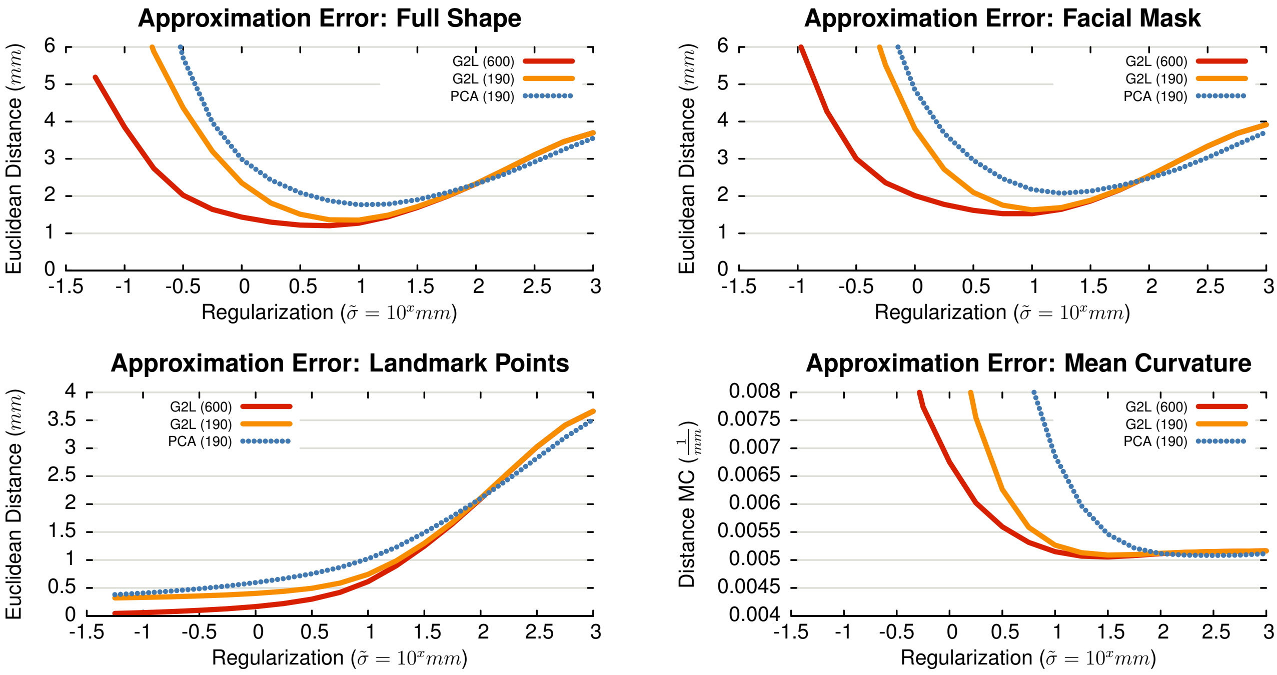


Figure 4.5: Experiment A (`MPI-TÜ-C` data set): Generalization from 100 vertices: The 20 examples of the test set are reconstructed using PCA model with 190 components (blue dotted line) and two G2L models (with 190 components: orange line / with 600 components: red line) with different regularization. Using a reasonable regularization (e.g. $\tilde{\sigma} = 10^{\frac{1}{2}}$), the Euclidean distance is smaller for the G2L model compared with the optimal regularization for the PCA model. This applies when the distance is evaluated over the full face as well as when the distance is evaluated over the facial mask. This is also true when the number of components of the G2L model is restricted to the number of components of the PCA model. Additionally, the G2L model shows less error at the landmark points.

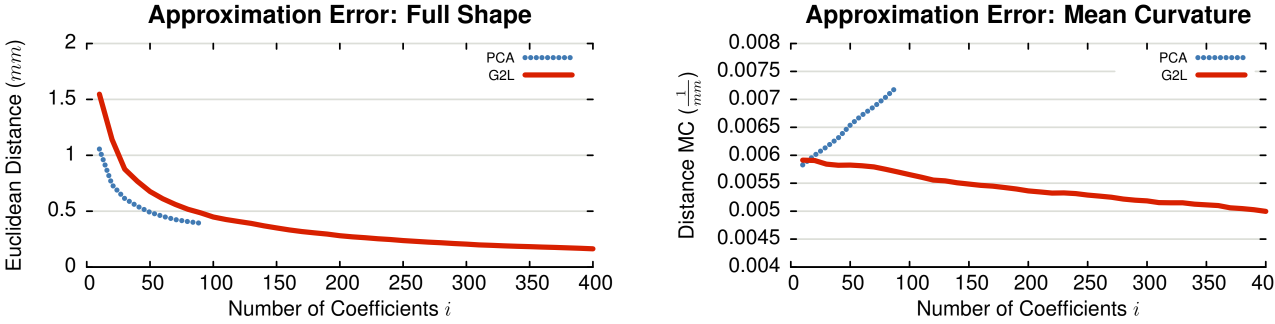


Figure 4.6: Experiment B (`UniBS-A` data set): Projection of novel examples (from the test set) to PCA model (blue dotted line) and to G2L model (red line) with respect to different number of components. The plot shows the average over a test set of 25 examples. Here, in contrast to Experiment A (see Figure 4.3), with the same number of coefficients, the PCA model shows a smaller Euclidean Distance. However, for the G2L model, the Euclidean distance can be reduced using more components. For the G2L model, using more components does not distort the shape, as it is the case with the PCA model, as it can be seen in the mean curvature plot.

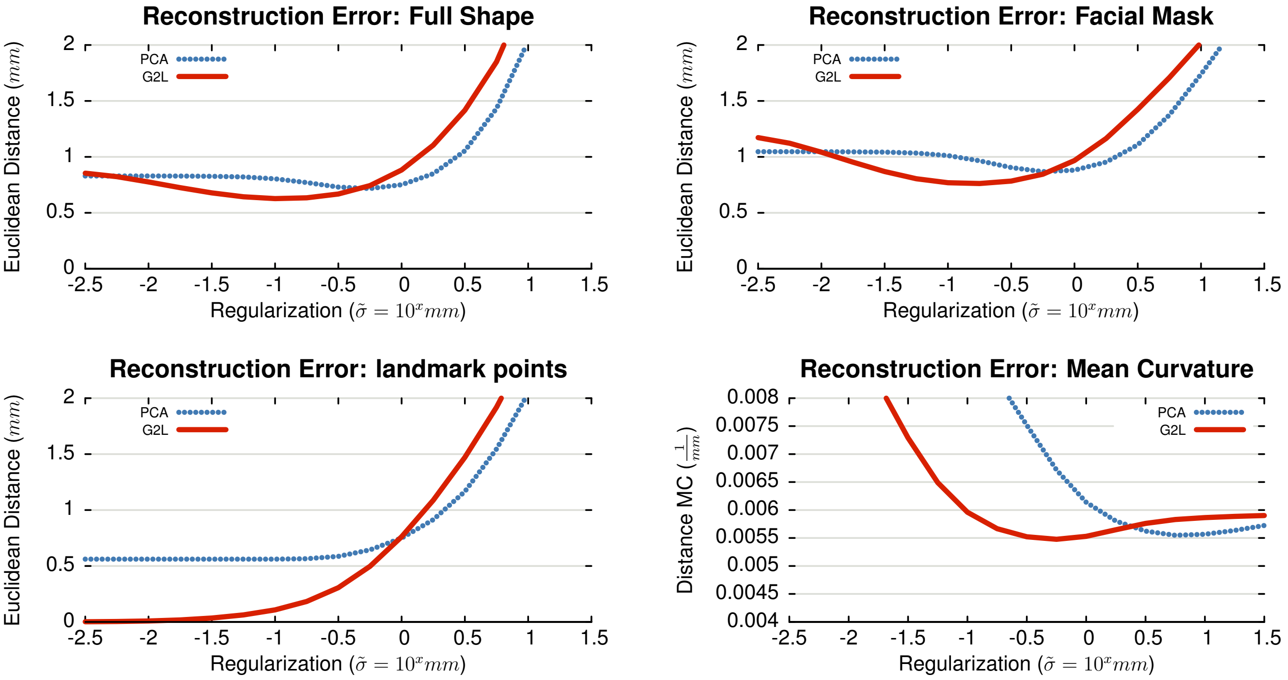


Figure 4.7: Experiment B (`UniBS-A` data set): Generalization from 77 vertices: The 25 examples of the test set are reconstructed using PCA model (blue dotted line) and G2L model (red line) with different regularization. In contrast to the G2L model, the PCA model is too inflexible to match the 77 landmark points. The G2L model is much more flexible and can match the landmark points almost perfectly. At the same time, it generalizes slightly better than the PCA model: The Euclidean Distance is smaller, i.e. on the facial mask. Overfitting can be avoided using regularization, as the mean curvature plot demonstrates. Examples of such reconstructions are shown in Figure 4.8.

4.4.2 Experiment B

Data

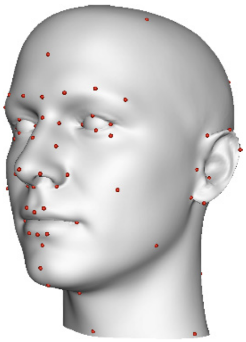
For this experiment, we use `UniBS-A` data, a data set of 125 registered scans, see Table 2.1. In contrast to Experiment A, where training and test data differ significantly, here, one data set was split randomly into a training set of 100 scans and a test set of 25 scans.

Projection

Unlike Experiment A, here the l^2 error is smaller for the PCA model, compared with the G2L model, see the left plot in Figure 4.6. This is to be expected since the PCA model minimizes the l^2 error on the training set [52], and the test and training set do not differ significantly as in Experiment A.

However, this is only true when models with the same number of components are compared, and the PCA model is limited to 99 components. Using a G2L model with more components, the l^2 error is much smaller then the best l^2 error of the PCA model. The mean curvature error plots (Figure 4.6, right) are similar to those of experiment A. Using more components, for PCA models, the mean curvature error increases, and for G2L model, it decreases.

Generalization



For generalization from a sparse set of landmarks points, the results are similar for those of Experiment A. The PCA model is too inflexible and cannot fit 77 3D landmark points. In this experiment, we have used more landmark points than in Experiment A, since the resolution of the `UniBS-A` data is higher then the resolution of the `MPI-TÜ` data, and therefore we are able to locate more feature points. The feature points used in this experiment can be seen in the figure in the margin. The PCA model has at the most 99 components and hence the problem is overconstrained. For this reason overfitting is not the main issue here. As can be seen in Figure 4.7 bottom left, even without regularization, the error at the landmarks is greater than $0.5\,mm$ in average. The G2L model, however, can fit the landmarks (almost) perfectly. For a reasonable regularization (e.g. $\tilde{\sigma} = 10^{-1}$) the l^2-error for the G2L model is a little better than the best one for the PCA model (Figure 4.7, top left). This effect is even stronger when only the facial mask is considered (here, we computed the error only over the face without the back of the head, Figure 4.7, top right). Nevertheless, also in this experiment slight overfitting appears. This can be seen in the mean curvature distance (Figure 4.7, bottom right) as well as in the examples in Figure 4.8. The smallest approximation error for the G2L model appears for a regularization parameter of $\tilde{\sigma} = 10^{-1}\,mm = 0.1\,mm$ (compared with $\tilde{\sigma} = 10^{-\frac{1}{4}}\,mm = 0.56\,mm$), i.e. we have to assume less noise on the landmark points to prevent overfitting.

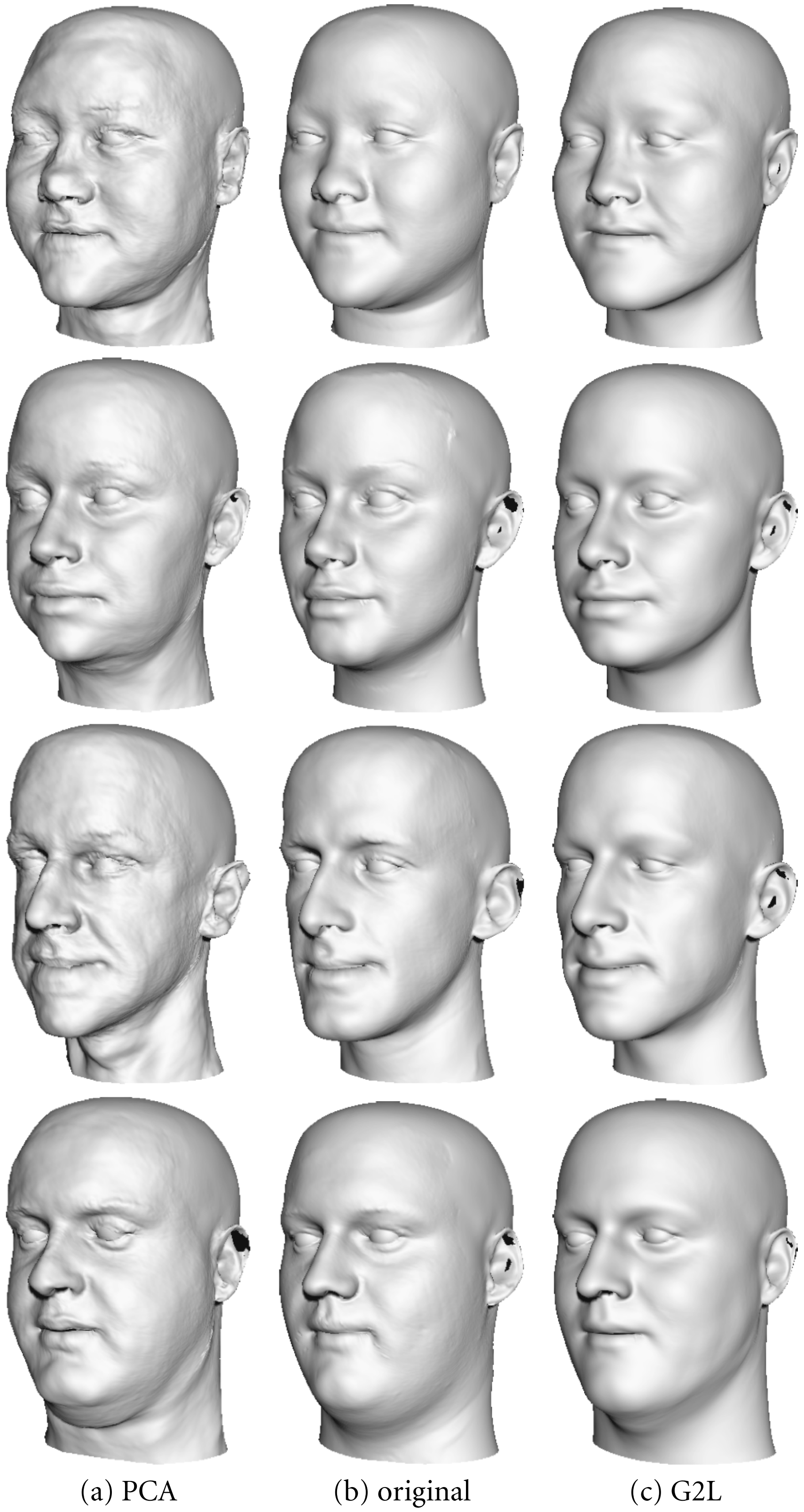

(a) PCA (b) original (c) G2L

Figure 4.8:
Experiment B (`UniBS-A` data set): Generalization from 77 vertices: the examples of the test set (b) are reconstructed using the PCA model (a) and the G2L model (c) with a regularization of $\tilde{\sigma} = 0.1$. For this parameter, the average distance at the landmark points is $0.11mm$ for the G2L model, compared with $0.56mm$ for the PCA model. Nevertheless, the PCA model shows slight overfitting. This can be also seen in the plot in Figure 4.7.

Chapter 5

Fitting

The most important application of statistical shape models is their use as prior knowledge in the reconstruction of the 3D shape of a face. For example, as input data for reconstruction noisy and/or incomplete 3D data can be used, as done in Chapter 4. The much more difficult scenario is given when no 3D data is available and the model parameters have to be estimated from a 2D projection of the face. In contrast to purely descriptive models, the parameters can not only be used as an abstract description of the face, but also to reconstruct the 3D shape.

In this chapter we describe a method to estimate the model parameters. As input we use a single photograph of the subject, together with a few (typically five to seven) manually labeled feature points. Fitting is done by estimating the model parameters with an iterative approach, using an *Analysis-by-Synthesis* loop: the 3D model is rendered as an image and this image is compared with the input image. The parameters are updated to minimize the residual. Automatically fitting the model to an image requires three components: a statistical model, a cost function and an optimization algorithm.

Statistical Model

To render the image, additionally to the statistical shape model, a camera model is used that describes the pose of the face, its 3D position in the world space and the internal parameters of the camera. Analogously, in addition to the statistical surface color model, a simple global illumination model is used. While the shape and color models have already been introduced, we derive the external models (i.e. the camera and the illumination model) in Section 5.1.

Cost Function

To model the optimization problem, a cost function (also called objective function) is needed. In the Analysis-by-Synthesis loop, the cost function compares the

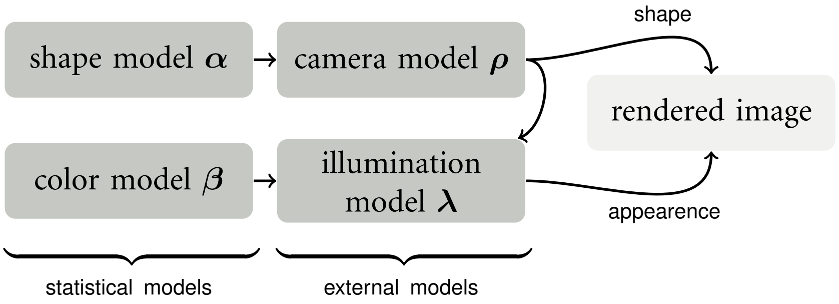

shape model $\boldsymbol{\alpha}$
camera model $\boldsymbol{\rho}$
color model $\boldsymbol{\beta}$
illumination model $\boldsymbol{\lambda}$
rendered image
shape
appearence
statistical models
external models

Figure 5.1: To render the images in an analysis-by-synthesis loop, the camera and the shape model define the 2D shape of the face in the image and the illumination model and the color model define the appearance.

input image with a rendered image. The most commonly used cost function is the l^2-distance between the generated image and the input image. However, this type of cost functions has many local minima. For this reason, instead of using one single cost function, we use different cost functions measuring the distance of landmark points, contours and pixel intensities.

Optimization Algorithm

To minimize the cost function, i.e. to find the parameters for which the cost is minimal, an optimization algorithm is needed. An large number of optimization algorithms have been developed. An overview of the most commonly used optimization algorithms can be found in [54] and [55]. The algorithms differ in terms of robustness, efficiency, and accuracy. These desired properties may conflict with each other, and different optimization strategies are suitable for different problems. In this work we use a variant of the Quasi-Newton algorithm (L-BFGS). We will justify this choice in Section 5.3.

5.1 External Models

The fitting system uses an analysis-by-synthesis approach: it minimizes the difference between the input image and the reconstructed scene. To render the image, in addition to the shape and color model, two external models (camera and illumination) are needed. The shape model (parameter $\boldsymbol{\alpha}$) gives the 3D position of each vertex in 3D. The camera model (parameter $\boldsymbol{\rho}$) projects the vertices to 2D positions in the image.

The relation of surface color model and illumination model is analogous, see Figure 5.1. The color model (parameter $\boldsymbol{\beta}$) defines inherent color of the surface. Since we are using per-vertex color, there is one RGB color per vertex. The illumination model (parameter $\boldsymbol{\lambda}$) simulates the light source and estimates the final appearance or pixel color in the image. While doing so, the estimated normals of the shape are used.

Usually, one is not interested in the external parameters, but mostly in the model parameters. These parameters describe the shape and color or albedo of the face and can be used to reconstruct the face. The parameters can also be used directly for recognition [17, 14, 18, 19]. The external parameters that describe pose, 3D position, internal camera parameter, and illumination conditions are most often not of direct interest.

5.1.1 Shape and Color Models

The shape model is a statistical model $\mathcal{M} := (\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{U})$ (see Equation 3.1) that defines the 3D position of each vertex

$$\vec{x}_i(\boldsymbol{\alpha}) = \vec{\mu}_i + \mathbf{U}_i \mathrm{diag}(\sigma_1 \dots \sigma_{n'})\boldsymbol{\alpha} \tag{5.1}$$

in model coordinates (i.e. in the model coordinate frame), where $\mathbf{U}_i \in \mathbb{R}^{3\times n'}$ are the corresponding rows of the model matrix $\mathbf{U}$ for vertex i (i.e. rows $3i,\ 3i+1,\ 3i+2$) and $\vec{\mu}_i$ are the corresponding entries of the mean $\boldsymbol{\mu}$. The statistical model can be either a G2L or a PCA model that is trained with the previously introduced training data (Equation 2.2).

Analogously, the color model is a statistical model that is trained with the corresponding per-vertex color (Equation 2.3). Here, we solely use a PCA-based color model. The color model defines the color of each vertex

$$\vec{a}_i(\boldsymbol{\beta}) = \vec{\mu}_{a,i} + \mathbf{U}_{a,i} \mathrm{diag}(\sigma_{a,1} \dots \sigma_{a,n'_a})\boldsymbol{\beta} \tag{5.2}$$

In Section 6.4 (future work) we discuss other possibilities for statistical color models. In order to render an actual image of the face described by the statistical model, we need a camera model.

5.1.2 Camera Model

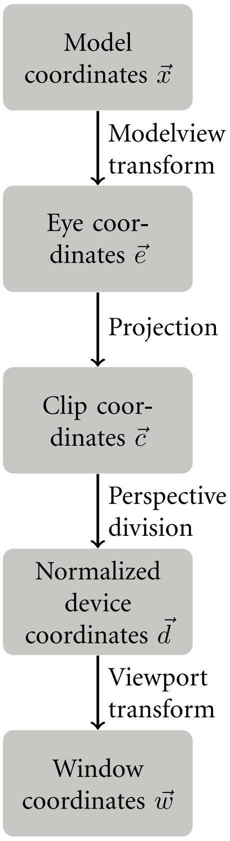
As camera model, a pinhole camera model is used. A general pinhole camera has nine degrees of freedom [56]: three internal camera parameters (focal length and principle point) and six external camera parameters that describe the rotation (three parameters) and 3D translation (three parameters) between the model coordinate (or object coordinate) system and the camera or eye coordinate system. We use a 9-parameter pinhole camera model $\vec{P}_{\boldsymbol{\rho}} : \mathbb{R}^3 \to \mathbb{R}^2$ where

$$\boldsymbol{\rho} = (\gamma, \varphi, \psi, t_x, t_y, t_z, f, o_x, o_y)^T \tag{5.3}$$

are the nine camera parameters, see Table 5.1.

ψ	nodding (rotation around x-axis)
φ	azimuth (rotation around y-axis)
γ	image plane rotation (rotation around z-axis)
$\vec{t} = (t_x\ t_y\ t_z)^T$	3D translation
f	focal length
$\vec{o} = (o_x\ o_y)^T$	offset of principle point

Table 5.1: The parameters of camera model $\boldsymbol{\rho} = (\gamma, \varphi, \psi, t_x, t_y, t_z, f, o_x, o_y)$ describe a pinhole camera.


Model coordinates $\vec{x}$
Modelview transform
Eye coordinates $\vec{e}$
Projection
Clip coordinates $\vec{c}$
Perspective division
Normalized device coordinates $\vec{d}$
Viewport transform
Window coordinates $\vec{w}$

Let $\vec{x} = (x_x, x_y, x_z)^T$ be a 3D point of the object in model coordinates. To simplify the notation, we have omitted the index identifying the vertex number. The corresponding 2D position in the image (*window coordinates* $\vec{w}$) is given by

$$\vec{w} = \vec{P}_{\boldsymbol{\rho}}(\vec{x}) = \vec{\pi}_{f,\vec{o}}(m_{\gamma,\varphi,\psi,\vec{t}}(\vec{x})) \tag{5.4}$$

where $\vec{\pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is a perspective projection from eye to window coordinates and the modelview transform $m_{\gamma,\varphi,\psi,\vec{t}}$ transforms the model coordinate to eye coordinates. In the following, we will derive the perspective projection and the modelview transformation and relate them to the OpenGL rendering pipeline [57]. The perspective projection π is split into the projection, the perspective division, and the viewport transform, see the figure in the margin. We use the notation used in the OpenGL specification [58], making it straightforward to implement the rendering using OpenGL.

Modelview Transformation

The modelview transformation transforms the vertices in model coordinates $\vec{v}$ into eye coordinates $\vec{e}$. To simplify the calculation, we assume a world with only one object. This implies that all vertices are transformed by the same modelview matrix. The camera is fixed, placed at the point $(0, 0, \mathfrak{d})^T$, and is looking along the negative z-axis. The parameter $\mathfrak{d}$ is fixed and is no longer acting as a parameter of the model. The modelview transformation is given by

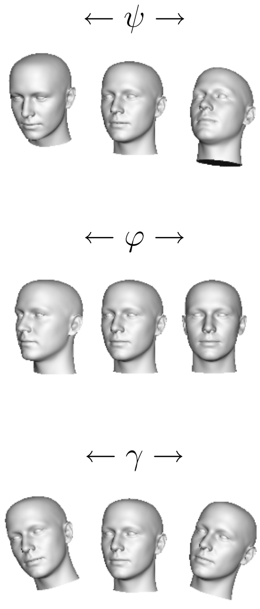
$$\vec{e} = m_{\gamma,\varphi,\psi,\vec{t}}(\vec{x}) = \mathbf{R}_\gamma \mathbf{R}_\varphi \mathbf{R}_\psi \vec{x} + (t_x, t_y, t_z - \mathfrak{d})^T \tag{5.5}$$

where $\mathbf{R}_\gamma, \mathbf{R}_\varphi, \mathbf{R}_\psi$ are elementary rotation matrices

$$\mathbf{R}_{\gamma,\varphi,\psi} = \mathbf{R}_\gamma \mathbf{R}_\varphi \mathbf{R}_\psi \tag{5.6}$$

$$= \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \tag{5.7}$$

around the z-, y- and x-axis, respectively. By defining the rotation matrix in this way, the angle φ is the azimuth or rotation of the face to profile view, the angle ψ describes the nodding of the face and the angle γ describes the image plane rotation, as shown in the margin figure.


$\leftarrow \psi \rightarrow$
$\leftarrow \varphi \rightarrow$
$\leftarrow \gamma \rightarrow$

Projection

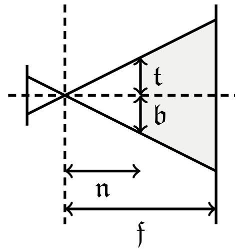
The frustum matrix $\mathbf{F}$ transforms the eye coordinates $\vec{e}$ into the (homogeneous) *clip coordinates* $\vec{c} = (c_x, c_y, c_z, c_w)^T$. With this transformation, the viewing frustum is defined, and the visible vertices lie within this frustum.

$$\vec{c} = \mathbf{F}\,(e_x, e_y, e_z, 1)^T\ ,$$

where $\mathbf{F}$ is the frustum matrix, given by

$$\mathbf{F} = \begin{pmatrix} \frac{2\mathfrak{n}}{\mathfrak{r}-\mathfrak{l}} & 0 & \frac{\mathfrak{r}+\mathfrak{l}}{\mathfrak{r}-\mathfrak{l}} & 0 \\ 0 & \frac{2\mathfrak{n}}{\mathfrak{t}-\mathfrak{b}} & \frac{\mathfrak{t}+\mathfrak{b}}{\mathfrak{t}-\mathfrak{b}} & 0 \\ 0 & 0 & -\frac{\mathfrak{f}+\mathfrak{n}}{\mathfrak{f}-\mathfrak{n}} & -\frac{2\mathfrak{f}\mathfrak{n}}{\mathfrak{f}-\mathfrak{n}} \\ 0 & 0 & -1 & 0 \end{pmatrix}. \qquad (5.8)$$

with the parameters $\mathfrak{n}$ear, $\mathfrak{f}$ar, $\mathfrak{r}$ight, $\mathfrak{l}$eft, $\mathfrak{t}$op, $\mathfrak{b}$ottom describing the clipping planes of the viewing frustum [58], as shown in the margin figure. For the camera model used, we assume that the viewing frustum is symmetric with $\mathfrak{l} = -\mathfrak{r}$ and $\mathfrak{b} = -\mathfrak{t}$. Additionally, we assume that the far clipping plane is much behind the near clipping plane ($\mathfrak{f} \gg \mathfrak{n}$). As a result, matrix $\mathbf{F}$ reduces to



$$\mathbf{F} = \begin{pmatrix} \mathfrak{n}/\mathfrak{r} & 0 & 0 & 0 \\ 0 & \mathfrak{n}/\mathfrak{t} & 0 & 0 \\ 0 & 0 & -1 & -2\mathfrak{n} \\ 0 & 0 & -1 & 0 \end{pmatrix}. \qquad (5.9)$$

In a pinhole camera, the focal length f is equal to the distance between the image plane and the pinhole. To manipulate the focal length without affecting the near clipping plane, we allow for scaling the width and height of the frustum inversely by f, i.e. $\mathfrak{r} = \frac{r_0}{f}$ and $\mathfrak{t} = \frac{t_0}{f}$, where r_0 and t_0 are kept fixed. Thus the frustum transformation reduces to

$$\vec{c} = \begin{pmatrix} c_x \\ c_y \\ c_z \\ c_w \end{pmatrix} = \mathbf{F} \begin{pmatrix} e_x \\ e_y \\ e_z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f\mathfrak{n}}{r_0} e_x \\ \frac{f\mathfrak{n}}{t_0} e_y \\ -e_z - 2\mathfrak{n} \\ -e_z \end{pmatrix}. \qquad (5.10)$$

$\vec{l} = (l_x, l_y, l_z)^T$	light direction
$\vec{c}_a = (c_a^R, c_a^G, c_a^B)^T$	color intensity of ambient light
$\vec{c}_d = (c_d^R, c_d^G, c_d^B)^T$	color intensity of directional light
c	color contrast
$\vec{c}_g = (c_g^R, c_g^G, c_g^B)^T$	color gain
$\vec{c}_o = (c_o^R, c_o^G, c_o^B)^T$	color offset

Table 5.2: The parameters of the illumination model $\boldsymbol{\lambda} = (\vec{l}^T, \vec{c}_a^T, \vec{c}_d^T, c, \vec{c}_g^T, \vec{c}_o^T)^T$ describe one directed light source.

Perspective Division

The *normalized device coordinates* $\vec{d}$ are computed by homogenizing the clip coordinates

$$\vec{d} = \begin{pmatrix} c_x/c_w \\ c_y/c_w \\ c_z/c_w \end{pmatrix} = \begin{pmatrix} -\frac{f\mathfrak{n}}{r_0}\frac{e_x}{e_z} \\ -\frac{f\mathfrak{n}}{t_0}\frac{e_y}{e_z} \\ 1 + \frac{2\mathfrak{n}}{e_z} \end{pmatrix}. \tag{5.11}$$

Viewport Transformation

The 2D *window coordinates* $\vec{w} = (x, y)^T$ (in pixels) are obtained by viewport transformation

$$\vec{w} = \begin{pmatrix} \frac{p_x}{2}(1 + d_x) + o_x \\ \frac{p_y}{2}(1 \pm d_y) \pm o_y \end{pmatrix} \tag{5.12}$$

where $\vec{p} = (p_x, p_y)^T$ are the width and height of the final image (both in pixels) and $\vec{o} = (o_x, o_y)^T$ is an offset (also in pixels) measured from the center of the image. The plus sign in the second component of the viewport transformation corresponds to an OpenGL-compliant window coordinate system with the origin in the lower left corner, the minus sign corresponds to a coordinates system with the origin in the upper left. We will use the OpenGL coordinate system. Summarizing, the perspective projection from Equation 5.4 can be written as

$$\vec{\pi}_{f,\vec{o}}(\vec{e}) = \begin{pmatrix} \frac{p_x}{2}(1 - \frac{\mathfrak{n}fe_x}{r_0 e_z}) + o_x \\ \frac{p_y}{2}(1 - \frac{\mathfrak{n}fe_y}{t_0 e_z}) + o_y \end{pmatrix}. \tag{5.13}$$

5.1.3 Illumination Model

We now have the 2D position for each vertex in the image. In principle, we could render a 2D image using the corresponding color value from the model.[1] The re-

---

[1] We omit rasterization here, since we do not need it for the definition of the cost function, i.e. we only render the vertices without rastering the triangles between the vertices. Rasterization is only needed for z-buffering, i.e. to determine which vertices are visible.

sult would be an image showing the pure albedo without any shading. To compute the pixel intensities for a shaded surface, we need a (local) illumination model.

The illumination model uses light which comes directly from only one light source (*direct illumination*), i.e. we do not consider reflected light (like from the walls) and effects like cast shadows. To model the reflection, we use the Phong reflection model, an empirical model of local illumination. This is an adequately simple illumination model and is very often used, in particular it is the default illumination model used in OpenGL. The color of the reflected light $\vec{c_l} = (c_l^R, c_l^G, c_l^B)^T$ is computed independently for the three color channels as

$$c_l = ac_a + ac_d \max(\langle \hat{n}', \hat{l} \rangle, 0) + sc_d \max(\langle \hat{r}, \hat{v} \rangle^\nu, 0) \tag{5.14}$$

so we can omit the color index here. The inherent color (albedo) of the surface a comes from the color model (Equation 5.2), c_a, c_d are the ambient and diffuse light colors, resp. (see Table 5.2).[2] With $\hat{l} = \frac{\vec{l}}{\|\vec{l}\|}$ we denote the normalized direction of the incoming light and $\hat{v}$ is the normalized viewing direction. Instead of transforming the surface normals into the eye coordinate system (similar to the modelview transform), the eye normals $\hat{n}'$ are estimated from the neighboring vertices in eye coordinates. This simplifies the computation of the derivatives of the surface normals, see Appendix 7.1.5. The (normalized) reflectance ray

$$\hat{r} = 2\langle \hat{n}', \hat{l} \rangle \hat{n}' - \hat{l} \tag{5.15}$$

can be computed given the normal and the light direction. As in [38] the color contrast is corrected (parameter c) and each color channel is multiplied by the color gain and an offset is added. The final pixel intensities are then computed by

$$\vec{I} = \begin{pmatrix} c_g^R & & \\ & c_g^G & \\ & & c_g^B \end{pmatrix} \mathbf{M}_c \begin{pmatrix} c_l^R \\ c_l^G \\ c_l^B \end{pmatrix} + \begin{pmatrix} c_o^R \\ c_o^G \\ c_o^B \end{pmatrix} \tag{5.16}$$

where

$$\mathbf{M}_c = c \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} + (1-c) \begin{pmatrix} 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \end{pmatrix} . \tag{5.17}$$

5.2 Cost Function

In each iteration of the analysis-by-synthesis loop, the cost function is given as the difference between the input image and the rendered image. The naïve way of

---

[2] The parameters $\nu = 10$ (shininess) and $s = 0.12$ (intensity of specular reflection) are kept fixed.

computing the difference is the sum over all pixels of the image

$$F_p(\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{\text{pixels}} p_i(\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.18}$$

where p_i is the cost per pixel. Defining the cost function as a sum over vertices (instead of a sum over pixels)

$$F(\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i} f_i(\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{5.19}$$

has the advantage that it is not necessary to render the full image, i.e. to rasterize the triangles of the projected model mesh. This assumes regularly distributed vertices, otherwise a weighting would be needed. Avoiding rasterization in the definition of cost function makes computing its derivatives much easier.

The cost per vertex f_i compares the calculated color of the vertex ($\vec{I}_i$; Equation 5.16) with the color from the input image $\widetilde{I}_i = \widetilde{\mathbf{I}}(\vec{P}_{\boldsymbol{\rho}}(\vec{x}_i(\boldsymbol{\alpha})))$ (i.e. the color from the input image $\widetilde{\mathbf{I}}$ at that position onto which vertex i is projected). The intensity cost function is given by the squared difference between the RGB intensities of input image $\widetilde{\mathbf{I}}$ and the RGB intensities of the rendered model

$$f_i^{\text{int}} = \|\vec{I}_i - \widetilde{I}_i\|^2 = \sum_{c \in \{R,G,B\}} (I_i^c - \widetilde{I}_i^c)^2 \, . \tag{5.20}$$

In Equation 5.19 the sum runs over some (not necessarily all) vertices $\vec{x}_i$. For different cost functions, the sum can run over a different set of vertices, e.g. the sum runs over all vertices that are visible in the input image, or just the vertices at the contour, see Section 5.2.1. For the intensity cost function, the sum runs over all visible vertices S^{vis}, i.e. over all vertices that are neither occluded by other parts of the face nor at the back of the face

$$F^{\text{int}} = \sum_{i \in S^{\text{vis}}} f_i^{\text{int}}. \tag{5.21}$$

In gradient-based optimization algorithms, the gradient $\vec{\nabla} F$ is needed to determine the local search direction. The derivative with respect to any parameter ξ is given by

$$\frac{\partial F^{\text{int}}}{\partial \xi} = \sum_i \frac{\partial f_i}{\partial \xi} = \sum_i \sum_{c \in \{R,G,B\}} 2(I_i^c - \widetilde{I}_i^c)(\frac{\partial I_i^c}{\partial \xi} - \frac{\partial \widetilde{I}_i^c}{\partial \xi}) \tag{5.22}$$

Here, ξ can be any of the parameters $\boldsymbol{\alpha}$ (shape), $\boldsymbol{\beta}$ (surface color), $\boldsymbol{\rho}$ (pose) or $\boldsymbol{\lambda}$ (light). The second term $\frac{\partial \widetilde{I}_i^c}{\partial \xi}$ is given by

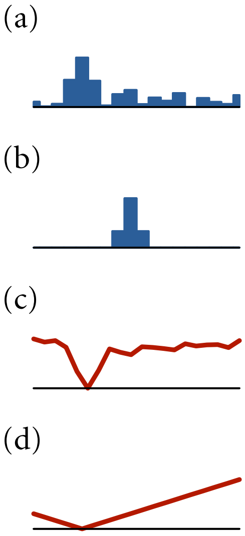
$$\frac{\partial \widetilde{I}_i^c}{\partial \xi} = \frac{\partial \widetilde{I}_i^c}{\partial x} \frac{\partial P_x}{\partial \xi} + \frac{\partial \widetilde{I}_i^c}{\partial y} \frac{\partial P_y}{\partial \xi} \tag{5.23}$$

where the gradients of the input images $\frac{\partial \widetilde{\mathbf{I}}^c}{\partial x}$, $\frac{\partial \widetilde{\mathbf{I}}^c}{\partial y}$ are estimated using finite differences (Sobel operator). The derivatives of the image projection $\frac{\partial P_x}{\partial \xi}$ and $\frac{\partial P_y}{\partial \xi}$ do not depend on the color parameters and the lighting parameters, therefore $\frac{\partial P_x}{\partial \xi} = \frac{\partial P_y}{\partial \xi} = 0; \xi \in \{\boldsymbol{\beta}, \boldsymbol{\lambda}\}$, the derivatives with respect to the camera parameters and the shape parameters are given in Appendix 7.1.1 and 7.1.2, respectively.

We use this cost function to estimate the illumination, the surface color and the shape (the pose is estimated by the landmark points and the contour, see below). Therefore we need the derivatives of the cost function (Equation 5.22) with respect to $\boldsymbol{\lambda}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$. The derivatives $\frac{\partial I_i^c}{\partial \lambda}$, $\frac{\partial I_i^c}{\partial \beta}$ and $\frac{\partial I_i^c}{\partial \alpha}$ can be found in Section 7.1.3, Section 7.1.4 and 7.1.6, respectively.

5.2.1 Special Cost Functions

When fitting the model to the image, we whish to find the optimal model and camera parameters by minimizing the difference between the rendered model and the input image. The cost function measures this distance between image and model. It should be neither too complicated nor too simple. When the cost function is too simple, the resulting parameters are only a poor approximation. When the cost function is too complicated, the optimizer is not able to find the global minimum. An optimal cost function is minimal at the best fit $\mathbf{p}^*$ and has no other (local) minima. This allows us to use gradient based optimization algorithms which work best on well behaved functions. In this way, we achieve fast and accurate optimization. Cost functions with many local minima require the use of robust optimization algorithms (e.g. stochastic optimization) that require many evaluations of the cost function (and possibly the gradient) and are therefore very slow.



In our fitting system, we use several types of cost functions. In this section, we introduce these cost functions and in Section 5.4 and Section 5.6 we describe in detail how they are applied in the fitting system.

The intensity cost function (Equation 5.21) has many local minima, as is illustrated in the toy example in the margin figure. Let (a) be the input image and (b) be a simple rendered model. The intensity cost function for translating (b) with respect to (a) is plotted in (c). Although this cost function has its minimal value for the desired translation of (b), it is very difficult for a gradient based optimizer to find the optimum, if the initial guess is far away from the optimum. An ideal cost function would have the form as shown in (d).

We wish to formulate our fitting algorithm in terms of such an ideal cost function. To compute this cost function, additional features such as edges or landmark points are extracted from the image, and the cost function is given as the distance to the nearest feature. Recently, approaches to learn locally ideal cost functions for face model fitting have been proposed [59]. The advantage of their approach is that the potentially difficult process of extracting edges from the image is avoided.

These cost functions used to determine the camera parameters $\boldsymbol{\rho}$ and shape parameters $\boldsymbol{\alpha}$ have the form of a general least squares cost function

$$F(\boldsymbol{\rho}, \boldsymbol{\alpha}) = \sum_i f_i(\vec{P}_{\boldsymbol{\rho}}(\vec{x}_i(\boldsymbol{\alpha}))), \qquad f_i \geq 0, \tag{5.24}$$

where f_i only depends on the 2D position onto which the vertex has been projected. In the fitting system, we use two such cost functions:

- Landmark Cost Function

  $$f_i^{\mathrm{lm}} = \|\vec{w}_i - \vec{\ell}_i\|^2 = \|\vec{P}_{\boldsymbol{\rho}}(\vec{x}_i(\boldsymbol{\alpha})) - \vec{\ell}_i\|^2 \tag{5.25}$$

  In this case, the sum runs over a set S^{lm} of predefined landmarks and ℓ_i denotes the 2D position of landmark i.

  $$F^{\mathrm{lm}} = \sum_{i \in S^{\mathrm{lm}}} f_i^{\mathrm{lm}} \tag{5.26}$$

- Contour Cost Function

  $$f_i^{\mathrm{cnt}} = d(\vec{w}_i)^2 = d(\vec{P}_{\boldsymbol{\rho}}(\vec{x}_i(\boldsymbol{\alpha})))^2 \tag{5.27}$$

  where $d : \mathbb{R}^2 \rightarrow \mathbb{R}$ is the distance transform of the contour image (see Section 5.6.2) i.e. $d(x, y)$ is the Euclidean distance of the pixel at (x, y) to the nearest contour pixel. In this case, the sum runs over all vertices on the contour S^{cnt}.

  $$F^{\mathrm{cnt}} = \sum_{i \in S^{\mathrm{cnt}}} f_i^{\mathrm{cnt}} \tag{5.28}$$

In the cases of the least squares cost function, the derivative (Equation 5.22) simplifies to

$$\frac{\partial F}{\partial \xi} = \sum_i \frac{\partial f_i}{\partial \xi} = \sum_i \left[\frac{\partial f_i}{\partial x} \frac{\partial P_x}{\partial \xi} + \frac{\partial f_i}{\partial y} \frac{\partial P_y}{\partial \xi} \right] \tag{5.29}$$

where ξ is one of the camera or shape model parameters. The important observation here is that the derivatives $\frac{\partial P_x}{\partial \xi}$ and $\frac{\partial P_y}{\partial \xi}$ do not change for different functions f (they have already been used in Equation 5.23). We will give these derivatives for all parameters $\xi \in \{\boldsymbol{\rho}, \boldsymbol{\alpha}\}$ in the Appendix 7.1.1 and Appendix 7.1.2. In most cases, the derivatives of the function f are either straightforward or are obtained numerically.

5.2.2 Regularization

Not every possible parameter vector represents a valid face. With statistical models, it is assumed that the model parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are distributed according to a specific distribution, in our case a normal distribution (Equation 3.3). The cost functions F^{int} (Equation 5.21), F^{lm} (Equation 5.26), and F^{cnt} (Equation 5.28) measure the difference between the input image and the rendered image (defined by $\mathbf{p}$) in different ways, without considering the likelihood of the parameters. As a result, minimizing these cost functions sometimes leads to very unlikely faces. For the cost function, this means that not only the difference has to be minimized, but we also have to assure that the parameters are distributed properly. The assumption that the parameters are distributed according to a normal distribution lead to the regularization term (see Section 3.6)

$$F^{\text{reg}} = \|\boldsymbol{\alpha}\|^2 . \tag{5.30}$$

5.3 Optimization

Given the cost function F that depends on the parameters $\mathbf{p} = (\boldsymbol{\rho}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, we have to find the parameters $\mathbf{p}^*$ that minimize the cost. A huge number of minimization algorithms has been developed and the area of numerical optimization was studied intensively in the last decades. Mainly, optimization methods can be divided into three categories: Methods that only evaluate the function for parameters $\mathbf{p}$, methods that require to compute the gradient and methods that also require to compute the second derivatives. Another type of optimization algorithms focuses on finding the global optimum (instead of only finding a local optimum), e.g. simulated annealing.

In [60] we argued that the cost function for the contour (Equation 5.28) is neither differentiable nor even continuous. The movement of texture contour points with respect to parameters is analytically differentiable, silhouette movement is piecewise differentiable (although generally not analytically), and inner contour movement is discontinuous (with 3D solids, contours appear, disappear, split and merge under parameter changes [61]).

Therefore, in [60] we used the *Downhill Simplex Method* [62, 55], a method known to be slow but robust. It uses function evaluations only and makes no assumption on the smoothness of the function, and therefore deals well with ill-behaved functions. However, since the gradient is not used, the method requires many more function evaluation than other optimization methods and consequently is very slow, too slow for our purpose.

When the gradient is available for the optimization, the naïve approach would be to use *steepest descent* (or gradient descent) [63]: Starting from the current estimate it searches the minimal point in the direction opposite to the gradient, uses this point as the new estimate and iterate until a (local) optimum is found. Steepest descent is known for very slow convergence close to the optimum. When the cost function space is badly scaled (i.e. the problem has a high condition number), the algorithm makes many small steps with many evaluations of the function and the gradient. *Stochastic gradient descent* (SGD) approximates the true gradient by the sum of the gradients of a small random subset of all vertices. This method is much faster (compared to gradient descent) and it was used in [1, 38] to fit the Morphable Model to an input image. SGD is very robust, it avoids local minima by adding noise to the gradient estimate [1, 64].

In [17] a stochastic version of Newton's algorithm (similar to stochastic gradient descent) is developed and used to fit the model to an image. While doing so, the second derivatives are computed numerically from the first derivatives. Similar to SGD, stochastic Newton algorithm avoids getting trapped in a local minimum by introducing a random error on the gradient.

When the cost function is designed differently, in particular, when it is smoother and has less local minima, non-stochastic optimization algorithms can be used. For instance in [2, 14] a method proposed by Levenberg and Marquardt (LM) was used as optimization method for fitting. This method is often considered the best optimization algorithm for least-squares cost functions, i.e. when the cost function is given as the sum of squares of deviations or residuals $\mathbf{r} : \mathbb{R}^{\bar{n}} \rightarrow \mathbb{R}$ ($\bar{n}$ is the number of parameters in $\mathbf{p}$)

$$F(\mathbf{p}) = \frac{1}{2} \sum_j \mathbf{r}_j^2(\mathbf{p}) \,. \tag{5.31}$$

The LM algorithm makes use of the special form of least-squares cost functions and approximates the Hessian matrix $\nabla^2 F$ using the Jacobian matrix $\mathbf{J} : \nabla^2 F \approx \mathbf{J}^T \mathbf{J}$, ignoring second order terms. For large-residual problems, i.e. problems in which the residual is large at the minimum, the second order-terms are too large to be excluded, and therefore, the performance of LM is poor in these cases [54]. In our fitting system, the intensity cost function (Equation 5.21) and the contour cost function (Equation 5.28) are such large-residual problems.

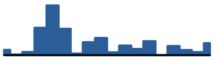
Optimization methods have been developed that accumulate information gained during previous iteration steps, such as *conjugate gradient* and *Quasi-Newton* algorithms. Like steepest descent, Quasi-Newton methods require only the gradient and the cost function in each iteration. They approximate the Hessian matrix of second derivatives $\nabla^2 F$ by a matrix that is updated in each iteration by analyzing the last gradient vectors. Compared to gradient descent, the increase in speed is dramatic and they are sometimes even faster than Newton's method since the computation of the second derivative is avoided [54].

Here, we use one of the most popular quasi-Newton algorithms, L-BFGS, the memory-limited version of the method proposed by Broyden, Fletcher, Goldfarb and Shanno. It is designed for large-dimensional problems when the approximation of the Hessian matrix $H \in \mathbb{R}^{\bar{n} \times \bar{n}}$ is too large to store. Instead, only the history of past p updates of $\mathbf{p}$ and $\nabla F(\mathbf{p})$ is stored [65]. It is known that the BFGS method has very effective self-correction properties. If the approximation of $\nabla^2 F$ is incorrect, it will correct itself within a few steps [54]. These properties are important since in each iteration the vertices over which the cost function is evaluated might change. For the contour term (Equation 5.28) in each iteration it is recalculated which model vertices are on the contour. For the pixel intensity cost function the visibility of the vertices is newly determined in each iteration.

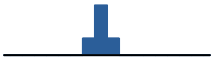
5.4 Stepwise Fitting System

The fitting system fits the parameters step by step to the input image. For this purpose, several modules are used that run consecutively. In Figure 5.2 the fitting system is illustrated. In Section 5.6 we will describe the different modules with all relevant details. In the following, we give a overview of the fitting system.

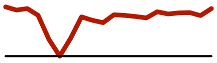
(a)



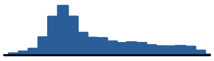
(b)



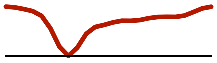
(c)



(d)



(e)



5.4.1 Multiresolution

The application of fast optimization methods requires the cost functions to be well behaved, i.e. they should be locally smooth and should not have too many local minima. In the fitting system we use several cost functions (F^{lm}, F^{cnt} and F^{int}). In particular, the intensity cost function has many local minima, but so does the contour cost function, when the contours or edges are extracted from an image with an overly sensitive edge detector. For this reason we combine the approach of using several cost functions with a traditional multi-resolution approach for the cost functions F^{int} and F^{cnt}, as illustrated in the plots in the margin. Again, (a) is the input image, (b) the rendered model and (c) is the intensity cost function with many local minima. If the input image is smoothed (d), the cost function (e) also becomes smooth and the minimum of this cost function can be found more easily. This is used as the initial guess for searching the (global) optimum in the original cost function (c). In most cases, this procedures also significantly speeds up the optimization.

5.4.2 Multi-Level Fitting

This principle is now used for the fitting. The cost function (Equation 5.20) is evaluated from two inputs, the input image ($\widetilde{\mathbf{I}}$) and the image generated by the

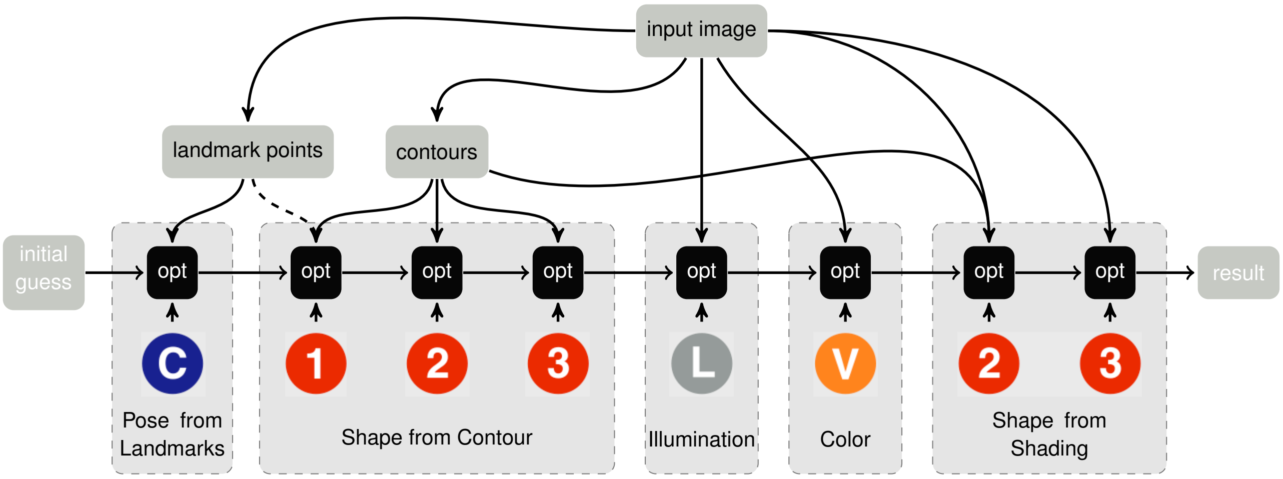


Figure 5.2: The fitting system consists of several modules that run consecutively. Each module updates the parameter of the corresponding model and gives the parameter to the next modules as initial guess. Model C is the camera model, 1,2,3 are the global and the local components of the G2L model, L is the illumination model and V is the per-vertex surface color model.

model (**I**). The input image is smoothed and downsampled, and the image gradients are computed from the downsampled images. Accordingly, in case of the contour cost function, the edges are extracted on downsampled images and different parameters for the edge detector are used, such that more or less detailed edges are used for the fitting.

For the model term, however, we make use of the multi-level structure of the G2L model and three levels of the G2L model are fitted one after the other. At first, the global shape model (level 1) is fitted to a low resolution version of the image, then the local components on level 2 are fitted to the higher resolution image, and finally the local components on level 3 are fitted to the original image. Usually, the very fine details on level 4 are neglected, since experiments have demonstrated (e.g. the experiment shown in Figure 5.6) that these components are not relevant for the fitting.

5.4.3 2-Stage Fitting

In addition to the multi-level fitting, the optimization problem is separated into two sub-problems, as illustrated in Figure 5.3. The 2-stage multilevel fitting system is detailed in Table 5.3.

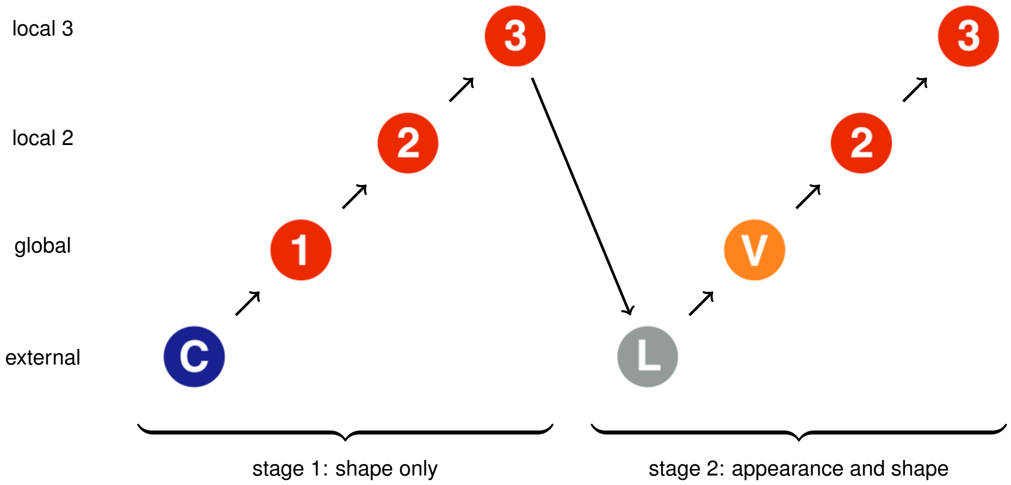


Figure 5.3: The fitting system consists of two stages: In Stage 1 the shape parameter ρ (camera model C) and α (G2L shape model level 1,2,3) are estimated without considering the appearance. In Stage 2, the approaches parameters λ (illumination model) and β (surface color model) are estimated and the shape parameter are refined. Both stages use a coarse-to-fine strategy, where first the external model parameters are estimated, then the global model parameters and then the local model parameters.

Stage 1

The position of landmarks and contours in an image does not depend on the illumination of the scene. For this reason, these features can be extracted without having estimated the illumination. These features are then used to estimate the camera and shape coefficients. Therefore, in the first stage of the fitting, we can fit the camera and the shape coefficients, while the illumination and color model parameters are neither estimated in the first stage, nor even needed for the cost functions F^{lm} (Equation 5.26) and F^{cnt} (Equation 5.28). The result is an approximation of the surface that matches these features very well, but lacks some of the internal features of the face. For example, for faces in frontal view the shape of the nose is only poorly reconstructed up to now, see Figure 5.5.

Stage 2

The estimated shape from Stage 1 is accurate enough to fit the illumination model and then the color model. Then, the estimated shape is used as an initial guess for the intensity cost function F^{int} (Equation 5.21) to refine the shape parameter of the local models on level 2 and 3.

5.5 SGD and MF Fitter

The fitter that we present in this chapter is different in many ways from the fitters used in our group before to fit a PCA model [1, 2]. Using the cost function from

Step	Module	Cost Function	Shape				Appearance	
			C	1	2	3	L	V
1	Pose from Landmarks	F^{lm}	★	⊘	⊘	⊘	╱	╱
2a	Shape from Contours	$F^{\text{lm}} + \eta^{\text{cnt}} F^{\text{cnt}}$	★	★	⊘	⊘	╱	╱
2b	Shape from Contours	$F^{\text{lm}} + \eta F^{\text{reg}}$	⊡	★	⊘	⊘	╱	╱
3	Shape from Contours	$F^{\text{cnt}} + \eta F^{\text{reg}}$	⊡	⊡	★	⊘	╱	╱
4	Shape from Contours	$F^{\text{cnt}} + \eta F^{\text{reg}}$	⊡	⊡	⊡	★	╱	╱
5	Illumination	F^{int}	⊡	⊡	⊡	⊡	★	⊘
6	Surface Color	F^{int}	⊡	⊡	⊡	⊡	⊡	★
7	Shape from Shading	$F^{\text{int}} + \eta^{\text{cnt}} F^{\text{cnt}}$	⊡	⊡	★	⊡	⊡	⊡
8	Shape from Shading	$F^{\text{int}} + \eta F^{\text{reg}}$	⊡	⊡	⊡	(★)	⊡	⊡

★	model parameters are optimized
⊘	model parameters are fixed to zero (mean)
⊡	model parameters are fixed to previous estimate
╱	not considered

Table 5.3: To fit the G2L model to images, a fitting system with different modules is used. In principle these modules could be combined in any order. This table gives an overview of the steps and the order, in which they are usually used for fitting. Thereby, either Step 2a or 2b are used and Step 6 is optional. Model C is the camera model, 1,2,3 are the global and the local components of the G2L model, L is the illumination model and V is the per-vertex surface color model.

Section 5.2, in this section we explain the similarities and dissimilarities in more detail.

5.5.1 SGD Fitter

In [1], the cost function is a combination of the intensity term f_i^{int} (Equation 5.20) and regularization term F^{reg} (Equation 5.30)

$$F = \eta \sum_{i \in S^{\text{rand}}} f_i^{\text{int}} + F_\alpha^{\text{reg}} + F_\beta^{\text{reg}} + F_\rho^{\text{reg}} \quad . \tag{5.32}$$

In addition to the regularization for the shape parameters $F_\alpha^{\text{reg}} = \|\boldsymbol{\alpha}\|^2$, a similar regularization for the surface color parameters is used ($F_\beta^{\text{reg}} = \|\boldsymbol{\beta}\|^2$) and an ad-hoc estimate of the camera model parameters $F_\rho^{\text{reg}} = \sum_j \frac{(\rho_j - \bar{\rho}_j)^2}{\sigma_{\rho,j}^2}$. In contrast to the intensity term F^{int} (Equation 5.21), the intensity term in [1] is computed as a sum over triangles. The fitting is initialized using five to seven manually set landmark points (we expect these landmarks to be detected automatically in the future, see Section 6.3).

To avoid local minima, in [1] two strategies are used. Firstly, stochastic gradient descent is used, a very robust, but not very fast optimizer (see Section 5.3):

the cost f_i^{int} is evaluated over S^{rand}, a random subset of 40 triangles that is chosen in each iteration, and the probability of selecting a triangle is proportional to the image area covered by this triangle.

Secondly, the following coarse-to-fine strategies are used to avoid local minima:

- In the first iterations, the resolution of the input image and the PCA model is reduced.
- At the beginning of the optimization, only the first few statistical model parameters α_j and β_j are optimized.
- The regularization parameter η is increased over time, reducing the weight of the prior probability.
- In the final steps of the optimization, the PCA model is broken down into segments. The coefficients α_j and β_j are optimized independently and the final surface is obtained by blending the segments.

5.5.2 Multi Feature Fitter

In [2] a different approach to the local minima problem (in the intensity cost function) is presented: In addition to the intensity cost function, various image features are used, yielding a cost function

$$F = \eta_{\text{int}} F^{\text{int}} + \eta_{\text{cnt}} F^{\text{cnt}} + \eta_{\text{spc}} F^{\text{spc}} + \eta_{\text{reg}} (F_\alpha^{\text{reg}} + F_\beta^{\text{reg}}) + \eta_{\text{tc}} F^{\text{tc}} \tag{5.33}$$

where the η_i's are the feature weighting factors. The specular highlight feature F^{spc} is based on the idea that the specular highlight can be easily detected (the pixels are saturated) and that on these points the surface normal is given by the half-vector between the view vector and the light direction. The texture constraint feature F^{tc} enforces the model color to be in a valid range between $[0..1]$. This combined cost function is smoother, and therefore it is not needed to use stochastic optimization. Instead, Levenberg-Marquardt optimization is used. The fitting is initialized using five to seven manually set landmark points.

We use this fitter as the reference system and compare our results with these fittings (see Section 5.8). In [2] the `MPI` model was used. However, this fitter is also tuned for the use with the `BFM` in [37], i.e. this fitter uses a PCA model that was trained from the same scans that are used to train the G2L model.

5.6 Fitting Modules

The full fitting system consists of several modules, namely modules for landmark fitting, contour fitting, illumination fitting, and shape-from-shading fitting. In Figure 5.2 the fitting system is illustrated. Additionally to the cost function and the gradient, each module needs a renderer that generates an image from the parameter $\mathbf{p}$ to be compared with the input image. Some modules perform some preprocessing on the input image. The modules are implemented in a way that they can easily be interchanged. Table 5.3 gives an overview of the steps in that order that is usually used for fitting. Each component takes a parameter vector $\mathbf{p} = (\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\beta}, \boldsymbol{\rho})^T$ as input and updates a subset of these parameters. The resulting parameters $\mathbf{p}'$ are then used as input for the next component. In the following we will describe the different modules with all relevant details.

5.6.1 Module 1: Pose from Landmarks

The contour fitting requires a rough initial guess of the pose [60]. The pose and the other camera parameters (3D translation, internal camera parameters) can be estimated quite accurately using a few landmark points. Here, the shape parameters are kept fixed, see Table 5.3. As cost function, the landmark cost function (Equation 5.26) is used. The derivatives with respect to the camera model parameters are given in Appendix 7.1.1.

To estimate the camera parameters, we typically use a set (depending on their visibility) of seven landmarks: The center of the eyes, the tip of the nose, the left and right corner of the mouth and the earlobes. Currently, these landmarks are either manually labeled (in case of photos) or synthesized (in case of the synthetic test sets). The reason for the selection of these landmarks is that we expect these landmarks to be detected automatically in the future, see Section 6.3. An alternative could be to first fit an Active Appearance Model [24, 66]. With this approach, at least for frontal and near-frontal images, one would get a higher number of feature points. When more landmark points are available, it is also reasonable to fit the first few shape components. However, in the typical case with 5-7 landmark points, these points are only used to estimate the camera parameters. An example of such a fitting is shown in Figure 5.4.

Since this is typically the first step, this component is initialized with the mean face in frontal pose, with a focal length of $500px$ and centered:[3]

$$\begin{aligned} \alpha_i &= 0 \quad \forall i && (5.34)\\ \varphi &= \vartheta = \gamma = t_x = t_y = t_z = o_x = o_y = 0 && (5.35)\\ f &= 500\,px. && (5.36) \end{aligned}$$

---

[3] with a camera distance $\mathfrak{d} = 5000mm$ and $r_o = t_0 = 100\,px \cdot mm$ and $\mathfrak{n} = 1mm$

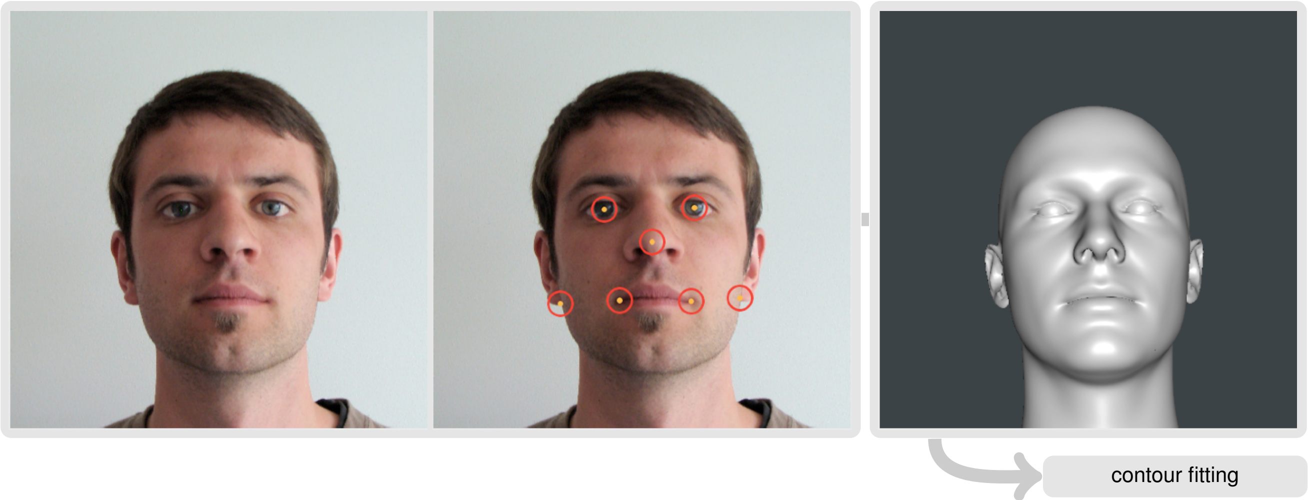


Figure 5.4: In Step 1 of the fitting system, the external camera parameters (pose and 3D translation) are estimated from the (manually labeled) landmarks (middle) on the input photo (left). The internal camera parameters (focal length, 2D offset of the principle point) are computed from the EXIF data of the input image.

Digital Camera

In cases where the input image is taken with a digital camera, the focal length f and the offset of the principle point $\vec{o}$ can be computed. The image file format used by digital cameras contains specific meta data tags. The EXIF (Exchangeable Image File Format) standard covers information such as the camera model (therefore we know the ccd size) and the focal length. In the original image we manually select the (quadratic) region of interest, crop the file and resample it to a resolution of $512px \times 512px$. The offset of the principle point o_x, o_y can be computed and the focal length is given by

$$f = 2\frac{r_0}{\mathfrak{n}} \frac{w_{\text{ccd}}}{f_{\text{camera}}} \frac{w_{\text{img}}}{512px} = 2\frac{t_0}{\mathfrak{n}} \frac{h_{\text{ccd}}}{f_{\text{camera}}} \frac{h_{\text{img}}}{512px} \tag{5.37}$$

where $w_{\text{ccd}}, h_{\text{ccd}}$ are the width and height of the ccd chip (in mm) and $w_{\text{img}}, h_{\text{img}}$ are the width and height of the camera picture (in pixel).

5.6.2 Module 2: Shading from Contour

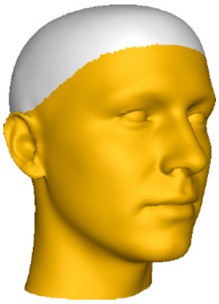
Different authors often think of different things when they use the term *contour*. Sometimes, contours are *image contours* (e.g. in the context of Active Contour Models [24]), a synonym of what we call *edges*. In the context of shapes, contour is often used interchangeably with *silhouette*. Other definitions of contour are implied by terms such as *inner contours*, *texture contours*, or *shadow contours*. We define contour as the collection of those edges whose image locations are invariant to lighting. Contours then fall into two classes: those originating in the geometry of the solid (*occlusion contours*), and those originating in material properties (*texture contours*). For example, a line drawing of a face would characteristically consist of just these edges.

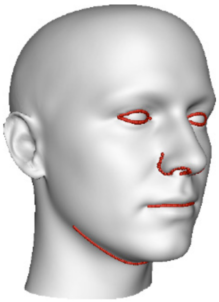
- Occlusion Contours
  Assuming a smooth solid shape [61], occlusion contours can be formally defined as the projection of those points $\vec{x}$ of the surface that are visible (i.e. that are not occluded by other parts of the object) and whose surface normal $\vec{n}$ is perpendicular to the (normalized) viewing direction $\hat{v}$ from the eye to the point $\vec{x}$, i.e. $\langle \vec{n}, \hat{v} \rangle = 0$. This definition encompasses both *outer* contours (silhouettes) and *inner* contours (e.g. on the nose), where inner contours only appear on non-convex solids.

- Texture Contours
  Texture Contours are permanent edges on texture maps. For faces, such contours are found at the lips, the eyes, the eyebrows, as well as at hair boundaries and within the hair.

Which contour features are accessible for matching depends on the application context. For example, when reconstructing faces from a shadow play, only silhouettes are available. When the input is a photograph, all types of contours are in principle accessible.

In order to match the contours generated from the model to the contours or edges in the image, two tasks need to be done: render the contours from the model, and compare them with the contours in the input image.





Rendering Function

The contour of the model is generated by determining front- and back-facing polygons on the mesh. Edges between a back and a front facing polygon are contour candidates. Hidden contours are eliminated through z-buffering. Since hair is not covered by the model, we do not consider points on the back of the head. For this reason, we do not use contour points outside the mask shown in the margin. In addition to the occlusion contours, we marked some texture contours in the face, namely the line around the eyes, the nostrils and the mouth line, and the chin. These contours are always rendered when they are visible, i.e. when they are not occluded by other parts of the face. The choice of these contours is based on the observation of what kind of contours can typically be detected in the image by the edge detector we use. Depending on the viewing direction, occlusion contours are generated at the same positions as some texture contours. In particular, the texture contour at the chin is hidden (and removed by the z-buffering) and replaced by the occlusion contour when the person looks down. However, when the person looks up, no occlusion contour is generated at the chin. Finally, the remaining edges are projected onto an image.

Distance Transform

For the contour cost, we need to compute the difference between the rendered contour and the extracted contour. Edge comparison is a research topic in its own right with many interesting approaches, such as elastic matching, Fourier descriptors, or shock graphs [67], to name only a few. All of them are computationally expensive and require a relatively noise-free input. As long as we are dealing with unreliable input from the edge detector, we choose the more primitive yet robust approach of distance transforms, which solves correspondence implicitly, however, not necessarily well.

The contour cost function (Equation 5.28) is defined as the sum over the distance of the model contour vertices (projected onto the image) to the nearest edge detected in the image. To compute the cost efficiently, we use the Euclidean distance transform [68] of the input contour image. This is a scalar field $d : \mathbb{R}^2 \rightarrow \mathbb{R}$ where $d(x, y)$ is the Euclidean distance of the pixel at (x, y) to the nearest contour pixel.

The contour cost function is fairly robust against unmatched edges (i.e. edges that are detected in the image, but are not generated by the model, e.g. edges caused by cast shadow), although such edges can create an undesirable potential. On the other hand, contours not present in the input image will create large error terms, and optimization will drive them towards completely unrelated edges. To prevent this, the gradient far away from input edges should be small or zero. This can be achieved by using a threshold for the distance transform as in [60]. However, this leads to unwanted plateaus when several distance transforms are combined (see below). For this reason we compute the distance transform as in [69]

$$d(x, y)' = \frac{d(x, y)}{d(x, y) + \kappa} \tag{5.38}$$

where κ is a parameter that controls the width of the valley (for an example, see Figure 5.5).

To extract the edges from the image, we use a general-purpose edge detection algorithm, the Canny edge detector [70], that is designed to work on a wide range of images. It has several adjustable parameters, in particular the size of the Gaussian filter that is used to smooth the input image and two thresholds for the intensity gradients. If the thresholds are set too high, only a few edges are detected and important information is missed, if they are set too low, irrelevant information and noise is falsely detected as edges. It is not only difficult to find a generic value that works well on all images, it is even difficult to find one for a specific image that works well on all regions.

In order not to tune the Canny-thresholds by hand, we use a distance transform that integrates the information over a range of edge thresholds (similar to [69]).

$$\bar{d}(x, y) = \sum_{\theta} \frac{d_\theta(x, y)}{d_\theta(x, y) + \kappa} \tag{5.39}$$

where θ is the parameter (i.e. the lower threshold) of the Canny edge detector. This approach has the advantage that weak edges (i.e edges that only appear for low Canny-thresholds) only have an influence if they are far away from strong edges. In addition, the gradient of the distance transform far away from detected edges is neglectable.

Cost Function

The distance transform codes for each pixel in the image the cost to the next contour. Accordingly, we use the contour cost function (Equation 5.28) that runs over the contour vertices of the model and evaluates the distance transform at the position of the projected vertices. The derivatives with respect to the shape model are given in Appendix 7.1.2. To prevent overfitting, we use regularization (Equation 5.30). This yields a combined cost function

$$F^{cnt/reg} = F^{\mathrm{cnt}} + \eta F^{\mathrm{reg}} \tag{5.40}$$

with the regularization parameter η. In Figure 5.5 the distance transform and the cost function are visualized, where the sum runs over the vertices in dark green.

Usually, in this module the parameters of the shape model are estimated, and the parameters of the camera model are kept fixed, see Table 5.3. Depending on the accuracy of the landmark points, it might be reasonable to refine the estimated pose parameters in the shape from contour module while fitting the components on level 1 (global). We implemented this as an optional step in the fitting, see Table 5.3. In this case, the cost function is combined from the contour term and the landmark term

$$F^{cnt/lm} = F^{\mathrm{cnt}} + \eta F^{\mathrm{lm}} \,. \tag{5.41}$$

Regularization is not needed here, since this is only used for the components on level 1.

Multi Level

The model components are fitted using the coarse-to-fine strategy (see Section 5.4.2), for both the model (different levels) and the input image. For the model,

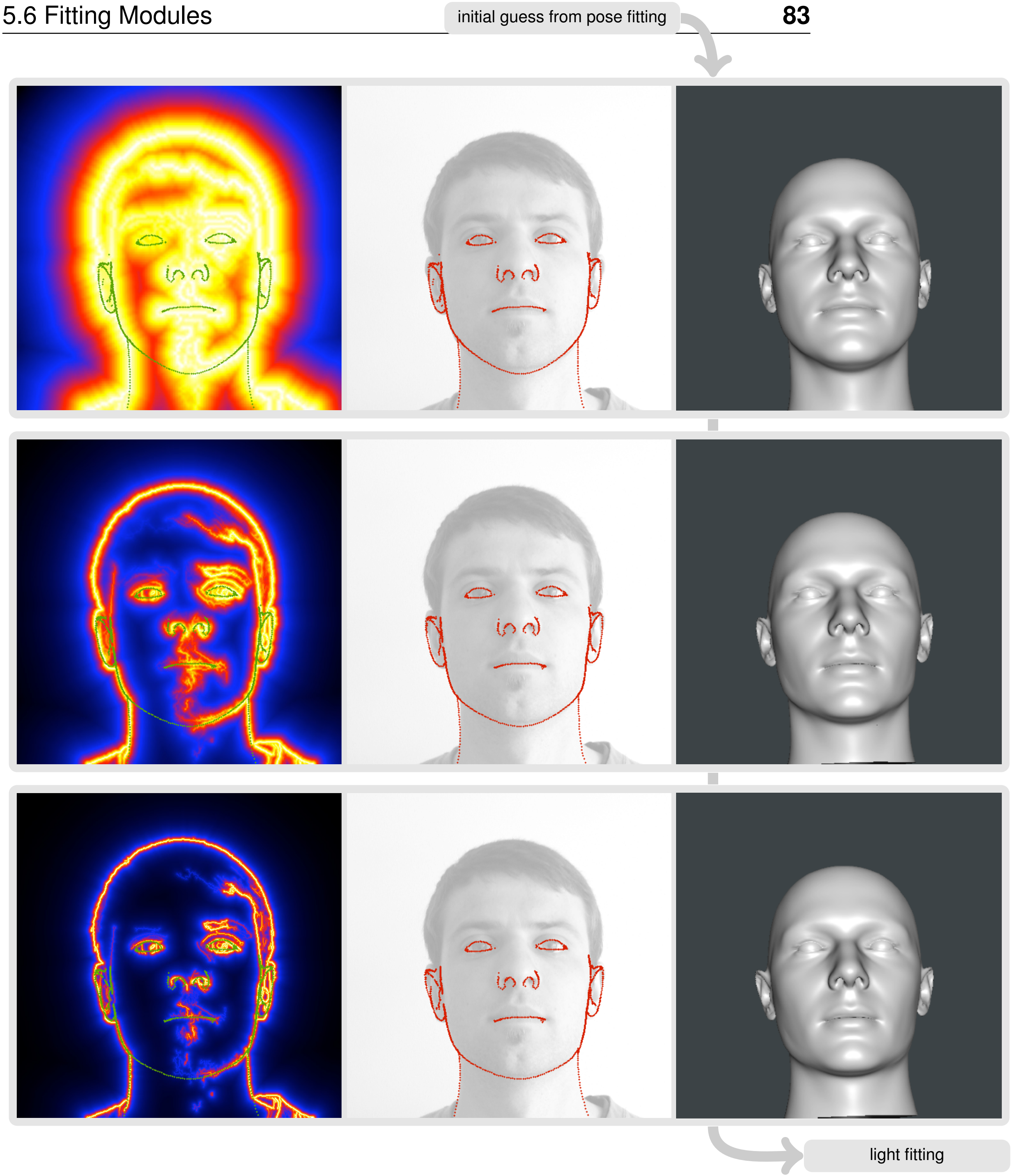


Figure 5.5: In Step 2/3/4 of the fitting system (module shape from contour), the shape parameters are estimated given the distance transform extracted from the image (left column, white is the minimal value of the distance transform, and black is the maximal value). The right column shows the estimated shape and the middle column shows the contour (texture and occlusion contours) of the shape model (red) rendered into the original image (gray). In the first row, the global shape parameters (level 1) are estimated from a coarse distance transform. In the second row, local level 2 parameters are estimated and in the third row level 3 parameters.

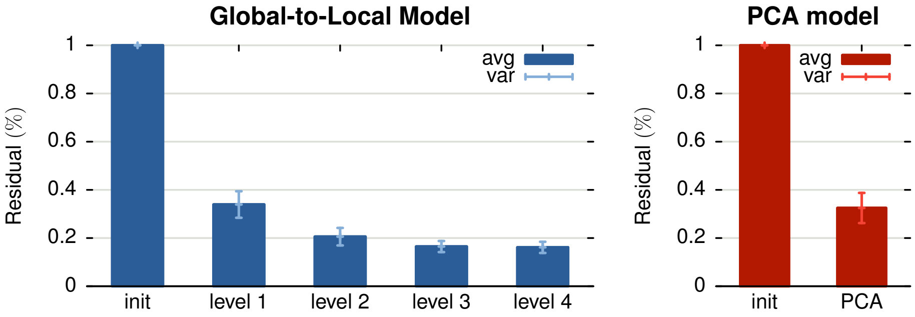


Figure 5.6: The plot shows the evaluation of the contour fitting in a synthetic test environment where the input images are computed from the 3D geometry of the faces in the test set (`UniBS-A test`, 10 images per subject with varying pose and illumination). Since the numerical value of the cost function differs depending on the size of the head and the pose (the number of vertices at the contour is different), the error is normalized with the error after the pose from landmark step that is used as initial guess for the contour fitting. The left plot shows the error for the G2L model after fitting level 1,2,3 and 4.

this means that the parameters of the already fitted levels are kept fixed. The edge detection is done on a different resolution of the image and with different parameters κ for the distance transform (for an example, see Figure 5.5).

level	parameter	image size
level 1 (global)	$\kappa = 100$	$2^7 = 128$
level 2	$\kappa = 30$	$2^8 = 256$
level 3	$\kappa = 15$	$2^9 = 512$

While doing so, we are interested in the question, how far it is reasonable to fit higher levels of the model to edges, i.e. whether the error increases using more levels. For this, we use a synthetic test environment, where the distance transform is extracted from images that are rendered from the 3D geometry of the faces in the test set with varying pose and illumination, similar to the experiments we have used in [60]. The result of this experiment is shown in Figure 5.6. Even tough level 1 (global) has the largest influence, local levels 2 and 3 further reduce the fitting error. In particular, the final fitting error for the G2L model is much smaller than the fitting error for the PCA model. Both the G2L model and the PCA model are trained from the same dataset (`UniBS-A` training) and the same set of distance transform (generated from `UniBS-A test`) was used.

5.6.3 Module 3: Illumination from Intensity

After having completed the first stage, the result of the contour fitting is a quite accurate fit of the shape that lacks some internal features of the face, but is accurate enough to start with the second stage where the appearance of the face is used, i.e. from now on modules are used in which the cost function is (mainly) the intensity cost function (Equation 5.21). Similar to the first stage, we start with estimating the parameters of the external model, i.e. the illumination model, while all other parameters are kept fixed, see Table 5.3. The derivatives with respect to the parameters of the illumination model is given in Appendix 7.1.3.

In contrast to the contour fitting, fitting the parameters of the illumination model is much more straightforward. Here we use an illumination model with ambient light and one directional light source and color correction (see Section 5.1.3), which was also used by [1]. This simple illumination model is sufficient for a wide range of images, for an example see Figure 5.7. As one can see in Figure 3.1, the first components of the surface color model are heavily influenced by the illumination. For this reason, we have separated fitting the illumination from fitting the surface color model. However, this might not be optimal. Using a different color model and fitting both components together is possible and might improve the result.

Rendering Function

For the first time during the fitting, we now need to render the full image: for each vertex its visibility, its position in the 2D image, and its color need to be computed. Here we use the estimated shape together with the vertex color of the mean face, again see Figure 3.1. In this step, only the color of each vertex changes, but not its position in the image.

Cost Function

To speed up the computation, we approximate the pixel intensity cost function and the gradient by running the sum over every $10th$ vertex. The selection is not chosen randomly, we simply select a fixed set. Our experiments have demonstrated that for the intensity module, this approximation hardly affects the result (however, this is not true for the shape from the shading module, see below).

5.6.4 Module 4: Surface Color from Pixel Intensity

In this module, the parameters of the surface color module are estimated while all other parameter are kept fixed, see Table 5.3. This module can be used when the parameters of the illumination model are estimated. As cost function, the pixel intensity cost function is used and the derivatives with respect to the model parameters are given in Appendix 7.1.4. We use this module optionally, and usually we are not interested in the estimated color, since this is replaced by the extracted texture. However, fitting the color (instead of using the color of the mean) improves the outcome of the next module, the shape form shading module.

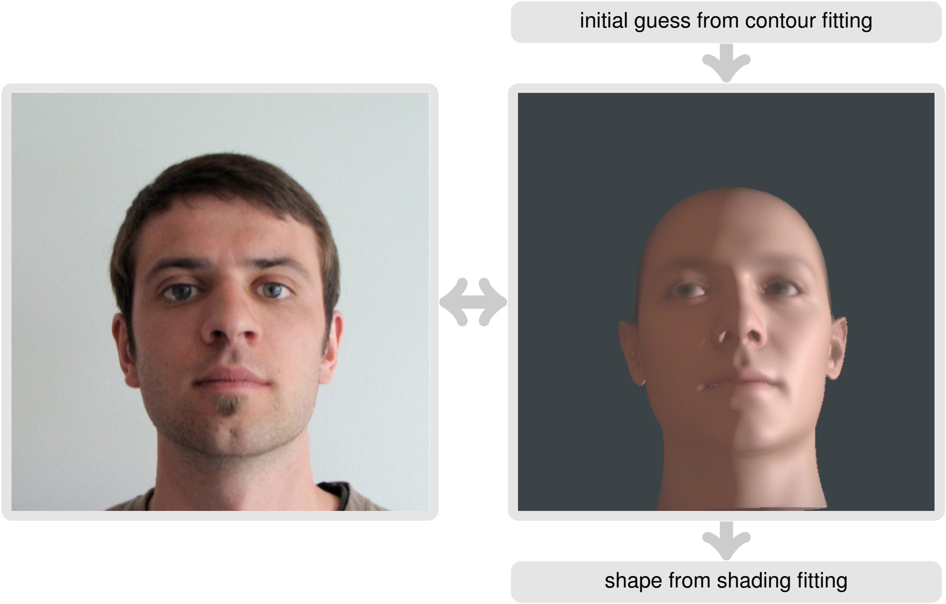

initial guess from contour fitting
shape from shading fitting

Figure 5.7: In Step 5 of the fitting system (module illumination from intensity), the parameters of the illumination model are estimated using the initial guess from the contour fitting and the color of the mean face.

5.6.5 Module 5: Shape from Shading

The estimated shape after Module 2 (Shape from Contour) fits the input features well but lacks some of the internal features of the face. This happens since some features like the width of the nose (in frontal pose) are not represented by the edge feature and are therefore not modeled by the contour model. Another reason is that due to bad illumination conditions in the input image some features might not be detected by the edge detector. In the Shape from Shading module, the parameter of the local shape components are refined, see Table 5.3. The global shape components (level 1) only affect the overall shape of the face. These parameters are already determined by the contour fitting and need no refinement.

The estimated shape after Module 2 is good enough to be used as the initial guess for the pixel intensity cost function. The derivatives with respect to the shape model are given in Appendix 7.1.2, where an approximation of the gradient is used, see Section 5.7.3. The challenge in this module is to keep the vertices at the contour fixed and fit the remaining flexibility. For this, we investigated four approaches:

Refining the Shape Parameters

The most straightforward idea is to use the estimated shape as initial guess and refine the shape parameters using the pixel intensity cost function (Equation 5.21). One could assume that the contours are not changed anymore, since the contours implicitly take part in the pixel intensities. However, our experiments have

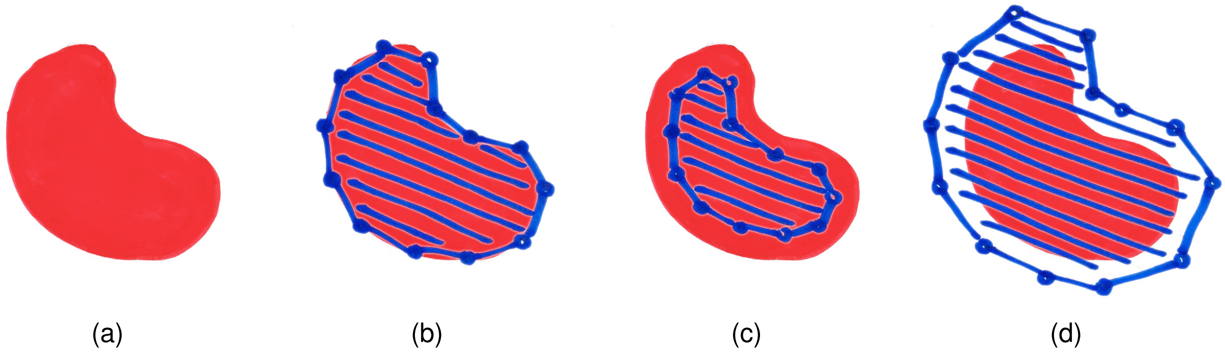


Figure 5.8: The input image (red patch in (a)) is compared with the rendered model (blue). The cost function is defined as the integral over the area of the rendered model. Thus, when the model is changed, the area where the cost function is evaluated also changes. For (d), the cost is large, since the background is compared with the model. However, for both (b) and (c) the cost is small. In practice, this causes the optimizer to shrink the model and destroys the contour fit.

demonstrated that the rendered model tends to shrink. The reason for this is, that by modifying the model parameters, the area over which the cost function is evaluated changes, since the cost function is defined as the sum over the visible vertices. For this reason, it also almost never happens that the rendered model expands, as is illustrated in Figure 5.8, since in this case, the background is compared with the rendered model.

Modeling the Remaining Flexibility

In [71] we presented a method to model the remaining flexibility when some part of a statistical shape model is fixed. When statistical models are fitted to partial data like the contour, many different reconstructions are possible, and we are not only interested in a plausible reconstruction but also the remaining flexibility within the model and the reliability of the reconstruction.

The remaining flexibility is modeled by flexibility components that can be fitted to the input image in the same way as the components of a statistical model. Our experiments have shown that this is possible in principle. However, the other two approaches show better results. The reason for this is twofold: Firstly, the vertices at the contour are fixed, i.e. they cannot move along the contour line, even though the contour cost would remain the same. Even worse, the vertices are not only fixed in 2D, but also in 3D, i.e. motion of the vertices along the viewing ray in 3D that generated the same 2D position is prohibited. Secondly, the flexibility components have global support, even though they are computed from a G2L model.

Combined Cost Function

Another approach to keep the vertices at the contour fixed and fit the remaining flexibility is to fix the 2D projection of the (contour) vertices. By using a combined cost function

$$\eta^{\mathrm{int}} F^{\mathrm{int}} + \eta^{\mathrm{cnt}} F^{\mathrm{cnt}} \tag{5.42}$$

we additionally allow changes of the position of the vertices along the contour and penalize when the vertices are moved away from the contour. Our experiments have demonstrated that this is the best approach to fit the relatively large local components on level 2. An example can be seen in Figure 5.9, first row (Step 6). In contrast to Module 3 (Illumination from Intensity, Section 5.6.3, here the cost function and the gradient are evaluated using all vertices.

Selection of the Local Models

The local components on level 3 have a smaller support than the components on level 2 (see Figure 3.11), and in fact, many components do not touch the contour. For this reason, here a different approach can be used that directly exploits the local support of the components.

The local components that are intersected by the outer contour (i.e. by the silhouette) are called *outer components* and the other components are called *inner components*. The partition in outer in inner components is computed based on the current fitting after having fitted on level 2. The outer components are already determined by the contour and need no refinement. The inner components are not determined, except those that are already fitted to the texture contours and the self-occlusion contours. On the other hand, changing these components does not influence the area in which the cost function is computed and there the problems discussed above do not appear. The outer components are fixed and inner components estimated. In doing so, regularization is needed, yielding the cost function

$$F^{\mathrm{int}} + \eta F^{\mathrm{reg}} \,. \tag{5.43}$$

The effect of this step is usually clearly visible in the shape. The shape of nose, lips and eyes changes considerably, since these features are not sufficiently determined by the contour used in the early fitting stages, for an example see Figure 5.9, second row, Step 7),

5.6.6 Texture Extraction

The result of the fitting is a very good estimate of the shape, the surface color, however, is not as well estimated. There are two reasons for this. Firstly, for the surface color we use a holistic PCA model (in Section 6.4 we discuss how the idea of the

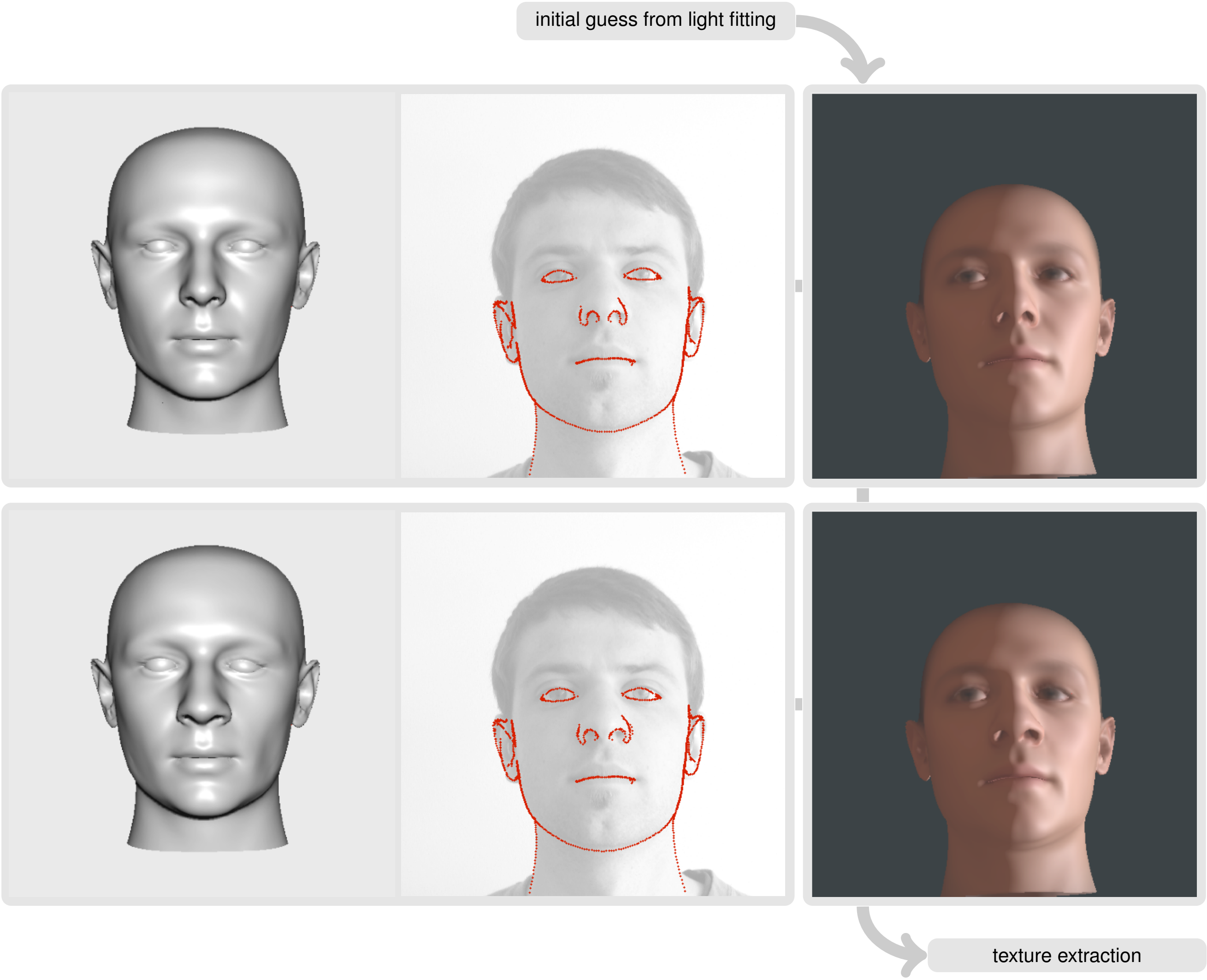


Figure 5.9: In Step 6/7 (module shape from shading) of the fitting system, the shape parameters are refined, in the first row on local level 2 and in the second row on local level 3. The left column shows the estimated shape in frontal pose, the middle column shows the contour (texture and occlusion contours) of the shape (red) rendered into the original image (gray). The right column the illuminated shape (with the mean color) as it is compared with the input image.

G2L model can be transferred to color models). Secondly, we cannot expect the high frequency details like stubble or freckles (and even wrinkles) to be properly registered.

To improve the visual quality of the model, we extract the texture from the input image, given the estimated shape. This step is also important in order to evaluate the quality of the shape estimation. When the shape is not properly estimated, some regions are badly aligned. This leads to artifacts when the estimated shape together with the extracted texture is rendered with a novel pose. Figure 5.10

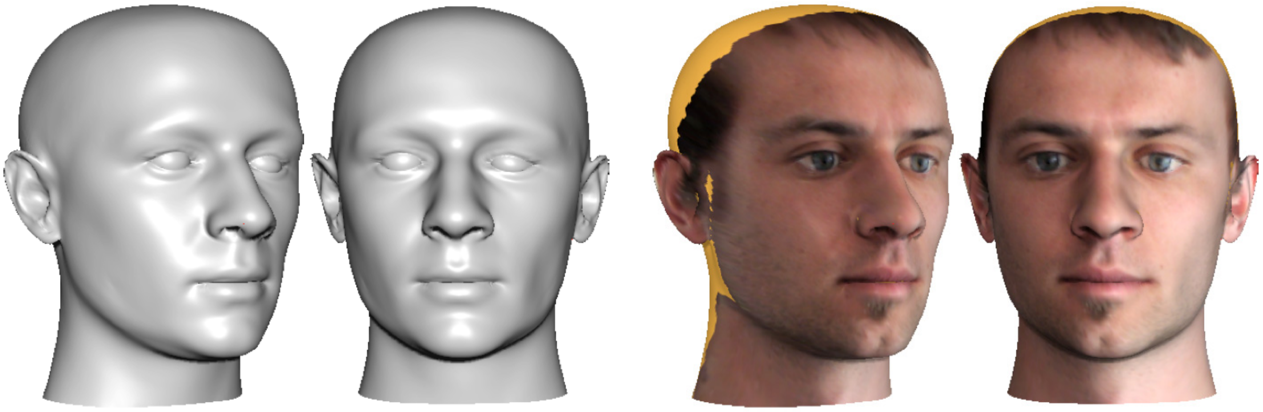


Figure 5.10: The G2L model (`UniBS-A`) is fitting to a single input photo (shown in Figure 5.4). The extracted texture demonstrates the highly accurate correspondence between estimated shape and input photo. The resulting shape is very similar compared with the (registered) scan of the same individual (`FLX`), see Figure 2.3

shows the estimated shape and extracted texture of example `FLX` under two novel poses and demonstrates that all regions are properly aligned and no artifacts are visible.

The extracted texture, however, is only of limited use and two problems can be observed. Firstly, the texture is incomplete, in particular when the input photo shows the person from the side (for an example, see Figure 5.13). Secondly, the extracted texture shows the appearance of the face including the illumination. To render the face with a different illumination, however, we would like to reconstruct the pure albedo, see future work, Section 6.5.

5.7 Efficiency

With the focus on accuracy, time efficiency was not out primary goal. On the other hand, the long run time of previous fitting methods was not only one of the main reasons that hindered the widespread application of these models, it also hinders the development, testing, and evaluation of the fitting system.

Computing a full fitting takes $\sim 30\ s$ on recent consumer hardware (see Table 5.4). This is much faster compared to all previous fitting algorithms. In [2] a run time of $\sim 70\ s$ on a 3.0 Ghz Intel®Pentium®IV computer was reported for the MMF/LM fitting and $\sim 4.5\ min$ on a 2.0 Ghz for the SNO algorithm. For a direct comparison with this method, it took $\sim 86\ s$ to compute the fitting shown in Figure 5.13, $3rd$ row, on the same hardware (Table 5.4) and with a model built from the same data. The run time could easily be further reduced at the cost of the accuracy (see future work, Section 6.3).

CPU	Intel®Core™2 Duo 2.40GHz (cache size: 4096 KB)
RAM	3GB
GPU	NVIDIA GeForce 7300 with OpenGL version 2.1.2
OS	Linux `fc9.x86_64` (64bit)
c++	gcc version 4.3.0
flags	`-O3 -funroll-loops -mssse3 -march=core2 -mfpmath=sse`
BLAS	Intel®Math Kernel Library 10.0.011
OpenCV	1.0.0

Table 5.4: The computer used for experimentation is a standard desktop PC.

Memory efficiency is not that much an issue. The whole model is loaded into the memory at the beginning of the fitting and the fitter does not use much more memory. So the memory footprint of the fitting algorithm is mainly determined by the size of the model.

The run time of the optimization is given by the number of function evaluations and the cost per iteration. The internal cost of the optimizer can be neglected (at least for L-BFGS). In order to develop an efficient fitting implementation, one has to choose an optimizer that requires few functions and gradient evaluations and to implement the evaluations efficiently. Strategies for achieving these efficiency goals are discussed in the following sub-sections.

5.7.1 Multiresolution, Optimization and Cost Function

By using the multiresolution approach (with multiple levels of the statistical model and multiple modules, see Section 5.4.2), the cost function in the first modules is smooth, and in the last modules we start with a quite good initial guess. Therefore, it is not necessary to use stochastic optimization algorithms, which are robust but slow. Instead, we can use very efficient optimization algorithms that require only a few evaluations of the function and the gradient, but have the disadvantage that they are sensitive to local minima in the cost function.

With the multiple modules strategy, not all parameters of all models are optimized at the same time and therefore not all gradients are needed all the time. Accordingly, only the gradients with respect to parameters that are not fixed are evaluated. The computational cost for evaluating the gradients differs dramatically for the different cost functions used. Especially the derivatives of the intensity cost function (Equation 5.22) are much more expensive than the derivatives of the other cost functions.

5.7.2 Implementation

On the other hand, the runtime of the fitting can be reduced by reducing the runtime per iteration. The cost of the function evaluation is mainly given by the cost of computing the linear combination of the model (see Equation 5.1) and rendering the image. The linear combination is efficiently computed using BLAS Level 2 (matrix-vector operations), and OpenGL could be used to compute the image. However, for all cost functions used, we need the 2D positions of the vertices in the image, not only the pixel intensities that can be read from the OpenGL framebuffer. For this reason, we compute the 2D positions (i.e. 4x4-matrix × 4-vector) using BLAS. OpenGL, however, is used for determining which vertices are visible and which are occluded by other parts of the object. This is done by using the OpenGL z-buffer. Thus, we avoid rasterization of the triangles on the CPU.

5.7.3 Elimination of irrelevant terms of ∇

A rule of thumb for software development says that 80% of the runtime of an algorithm are spent in 20% of the functions [72]. By profiling the implemented fitting system using `valgrind`, we checked which functions are expensive to evaluate, and we checked experimentally whether these terms are relevant for the result. In this way, we prevent the system from wasting time on terms that are not needed.

For the illumination model ($\boldsymbol{\lambda}$), the computation of the derivatives of the rendered image with respect to the parameters of the illumination model $\frac{\partial \mathbf{I}}{\partial \lambda}$ (Equation 7.19) are expensive especially for the light direction l (Equation 7.23). Here we approximate the true gradient by a gradient that is computed using 10% of the visible vertices. Experiments have demonstrated that this makes (almost) no difference in the fitting result.

For the shape model ($\boldsymbol{\alpha}$), however, a similar approximation reduces the quality of the fittings, therefore it is preferable to evaluate the cost function using all visible vertices. When we take a close look at the gradient (Equation 5.22 with $\xi = \alpha_j$) for the shape parameters,

$$\frac{\partial F^{\text{int}}}{\partial \alpha_j} = \sum_i \frac{\partial f_i}{\partial \alpha_j} = \sum_i \sum_{c \in \{R,G,B\}} 2(I_i^c - \widetilde{I}_i^c)(\frac{\partial I_i^c}{\partial \alpha_j} - \frac{\partial \widetilde{I}_i^c}{\partial \alpha_j}) \tag{5.44}$$

two additional terms need to be evaluated:

- The derivative of the input image with respect to the shape parameter $\frac{\partial \widetilde{I}_i^c}{\partial \alpha_j}$: changing the shape parameter changes the position in the image onto vertex i is projected to. Thus, the pixel intensity changes, which is compared with the color of vertex i.

- The derivative of the rendered image $\frac{\partial I_i^c}{\partial \alpha_j}$: changing the shape parameter changes the shape and therefore the normal of vertex i and hence color of vertex i due to illumination.

Our experiments have shown that $\frac{\partial \widetilde{I}_i^c}{\partial \alpha_j} \gg \frac{\partial I_i^c}{\partial \alpha_j}$, at the same time $\frac{\partial I_i^c}{\partial \alpha_j}$ is much more expensive to evaluate, see Appendix 7.1.5 and Appendix 7.1.6 (i.e. Equation 7.19, Equation 7.37 to Equation 7.39, and Equation 7.29 to Equation 7.36). For this reason, we approximate Equation 5.22 by

$$\frac{\partial F^{\text{int}}}{\partial \alpha_j} = \sum_i \frac{\partial f_i}{\partial \alpha_j} = \sum_i \sum_{c \in \{r,g,b\}} 2(I_i^c - \widetilde{I}_i^c)(-\frac{\partial \widetilde{I}_i^c}{\partial \alpha_j}) \,, \tag{5.45}$$

with little to no adverse effects on the fitting result.

5.8 Results and Comparison

It is quite difficult to evaluate the quality of a fitting result. This was the motivation for the evaluation in a controlled environment, see Chapter 4. Nevertheless, fitting results need to be evaluated, even if this evaluation is done visually. The same two points as in Chapter 4 need to be considered: Firstly, how well are the features of the face in the image matched? This is important in particular for the contours, but also for other features where the estimated shape and the input need to be in correspondence. This can be evaluated by extracting the texture and rotating the face into a novel view. Secondly, how well the model generalizes, i.e. how similar are the estimated shape and true 3D real shape of the person?

To compare fittings of the G2L model with the PCA model, it would be easy to use the fitter presented here with a PCA model. However, this comparison is unfair, since the fitter is not tuned to PCA models. In particular, due to the missing multiresolution of the PCA model, the fitter almost immediately gets stuck in a local minimum. For this reason, it is more meaningful to use the fitter from [14] as a reference system, which is optimized to the `BFM` PCA model [37].

5.8.1 Experiment C

In this experiment the model is fitted to photos of subjects from the `UniBS-A` test set. 3D scans of these subjects are available and hence we can compare the estimated surfaces with the captured surfaces. The G2L model is trained from the `UniBS-A` training set, but the subjects of the test set are not in the training set.

In Figure 5.10 and Figure 5.13 example fits are shown. The surfaces with extracted textures rendered in a novel view confirm that the features are matched

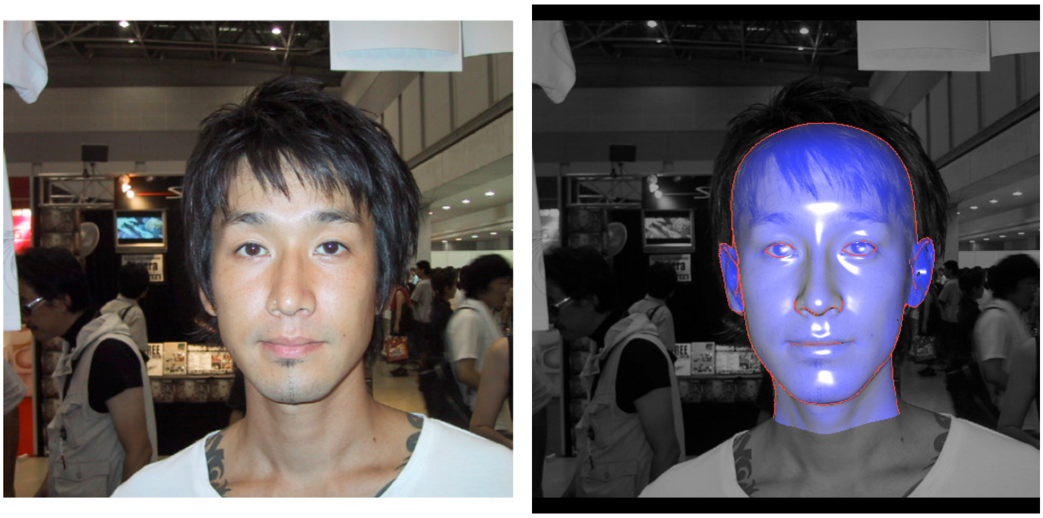


Figure 5.11: The photograph (left) from Figure 1.1 is fitted with the G2L model (`UniBS-A`). In the right image, the fitted surface (blue) and its occlusion contours and texture contour for the eyes (red) are rendered into the input image (gray), demonstrating that the accuracy at the edges (in particular at the chin) has been improved. The estimated shape can be seen in Figure 5.12.

accurately. Comparing the estimated surface with the captured surface of these subjects (shown in Figure 2.2 and Figure 2.3) shows that the surface is estimated well.

In Figure 5.13, $3rd$ row, the segmented PCA model (`BFM`) is fitted to the same input image of subject `PPY` using the fitter [2]. Subject `PPY` is not in the training set for the `BFM` model either (however, subject `FLX` is in that training set, and for this reason, we skip fitting the `BFM` to the `FLX` image).

In Figure 5.11 and Figure 5.12 we show the result of fitting the G2L to the photograph we used in the motivation, demonstrating the improvements in accuracy at the edges. However, this comparison is slightly unfair, since in Figure 1.1, the PCA model was trained from the `MPI-TÜ` dataset and the texture is not extracted from the photograph. On the other hand, Figure 1.1 is a good example of a state-of-the-art-fitting before the beginning of this work.

5.8.2 Experiment D

For this experiment, we use photos from a publicly available face database, the PIE database [73]. In Experiment D, the G2L and the PCA model are based on the same 200 scans: The G2L model is trained from the `UniBS-B` dataset and the PCA model is the `BFM`, see Section 2.1.2.

In Figure 5.14 and Figure 5.15 two examples of fittings of the G2L model and the (segmented) PCA model are shown. To compare the accuracy of the fits, we rendered the estimated surface in the estimated pose into the original image. The estimated surface is blue and semi-transparent, and its occluding contours and the texture contours at the eyes are marked in red. This demonstrated that the G2L

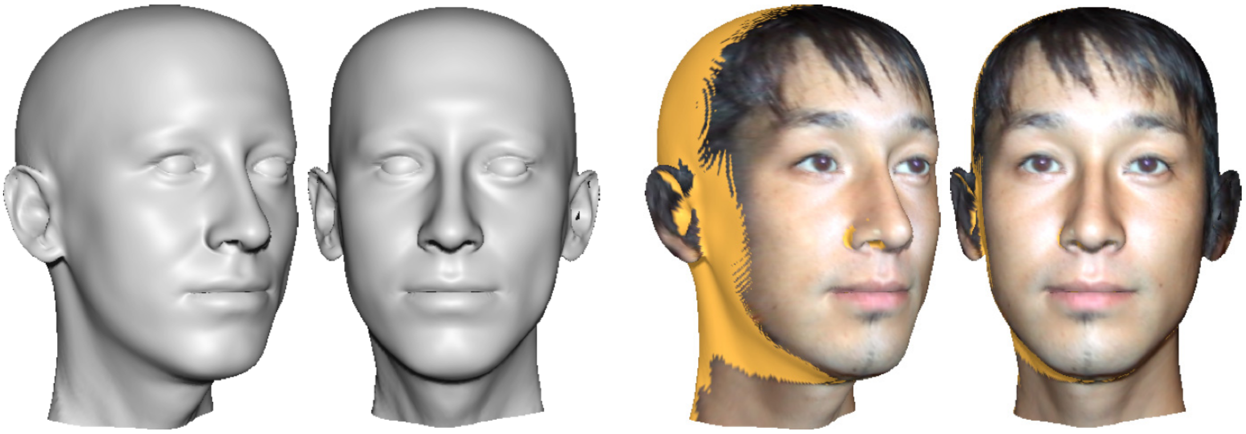
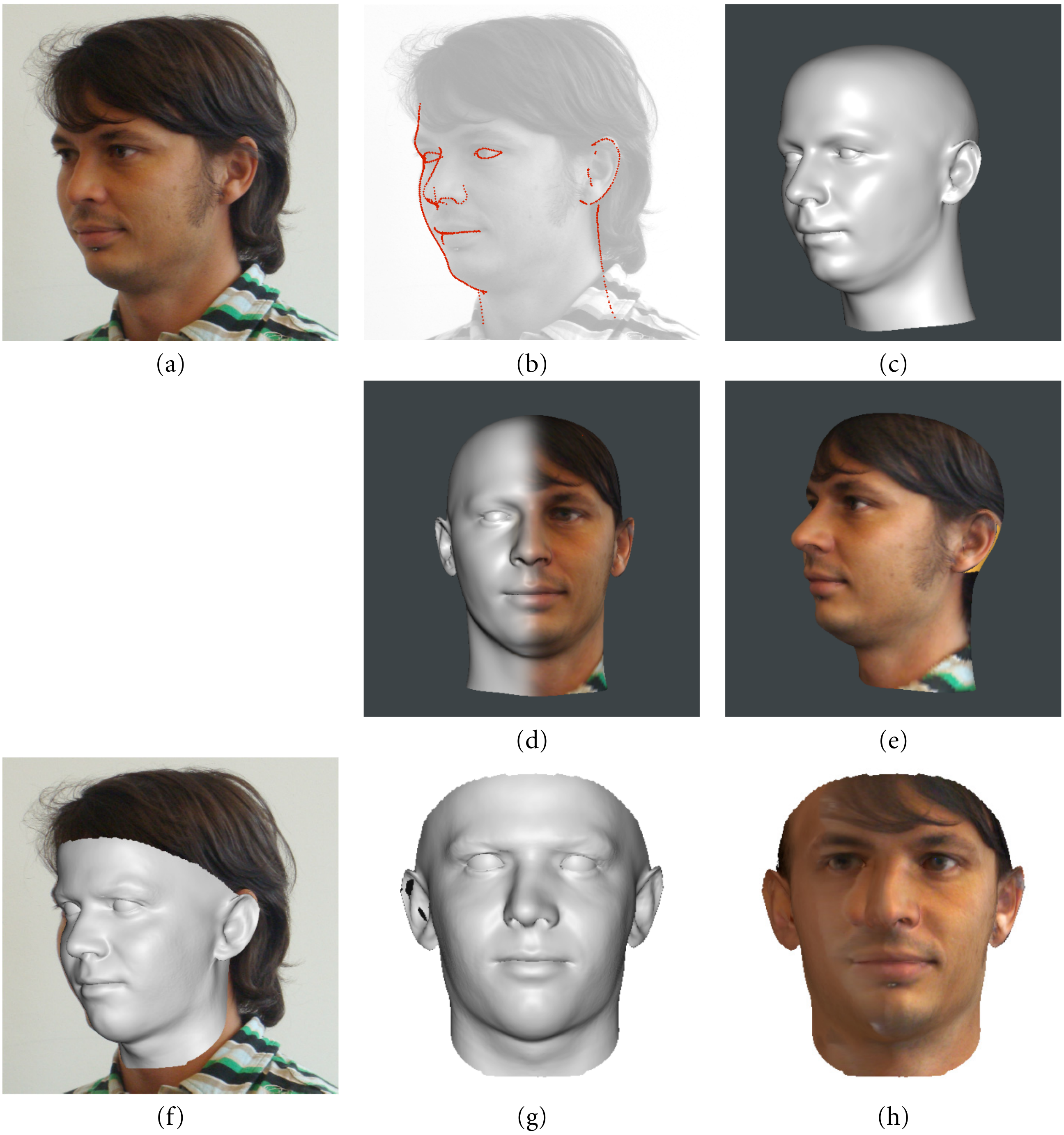


Figure 5.12: The photograph from Figure 5.11 is fitted with the G2L model (`UniBS-A`). The extracted texture demonstrates the highly accurate correspondence between estimated shape.

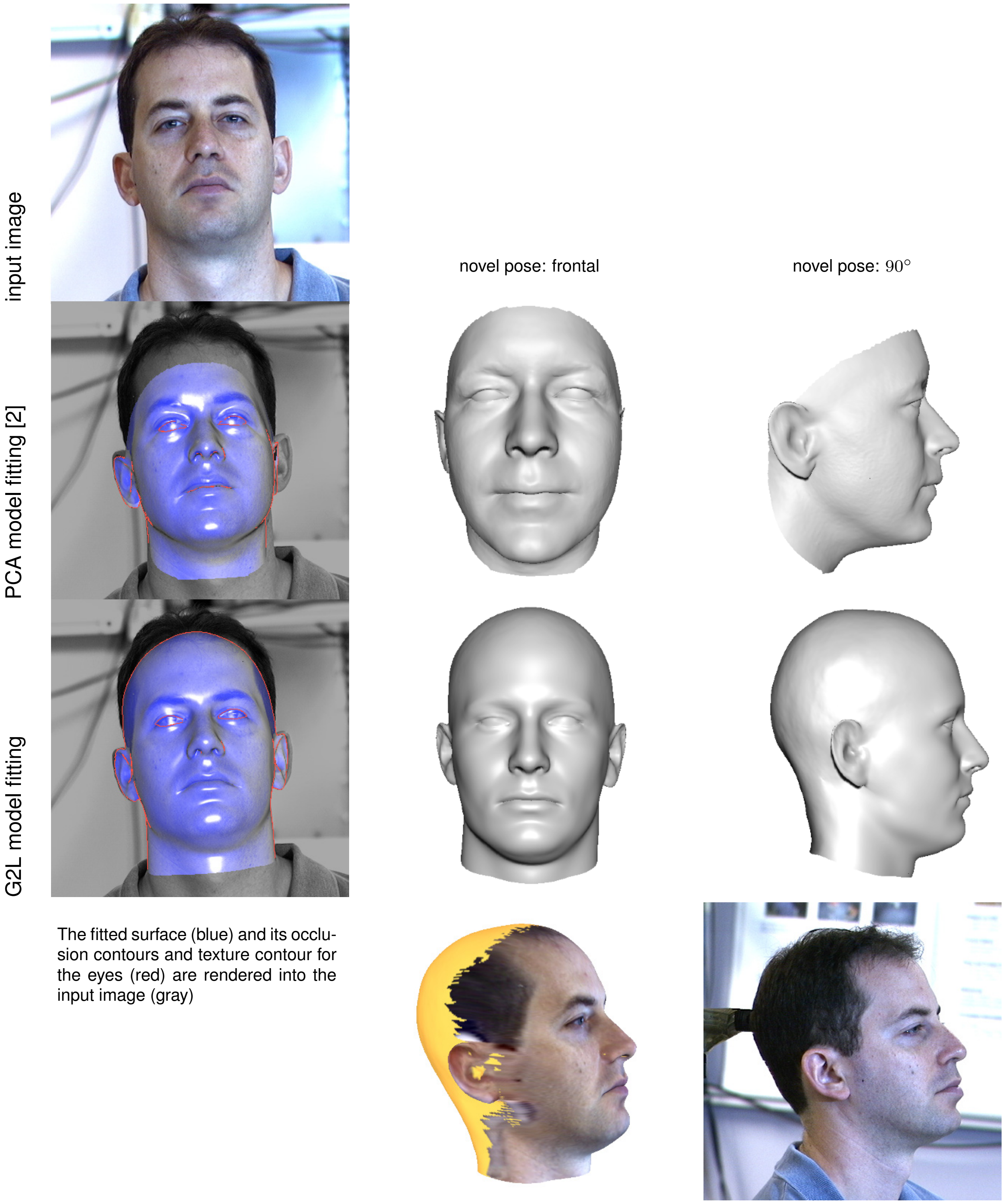
model improves the accuracy, in particular the eyes are in correspondence and most contours are matched correctly. In Figure 5.14 a typical problem shows up, where the contour of the left ear is matched to the inner contour of the ear.

Since for the PIE database the 3D surface of the subjects is not available, it is difficult to judge the generalization of the models. However, the PIE database contains images from different directions taken simultaneously. Hence, we can fit the pose of the estimated surface to an image of the same subject taken from a different direction (using Module 1: pose from landmarks). In Figure 5.14 the estimated shape in novel pose is very similar to the second photo. In Figure 5.15, however, the shape of the nose is a little too large (although much better compared with the PCA model fit). The reason for this is that in the photo in near frontal view, we can only make assumptions on the shape of the nose based on prior knowledge about the ethnic origin of the subject. However, the model cannot do that, since the dataset contains (almost) only scans of Caucasians.



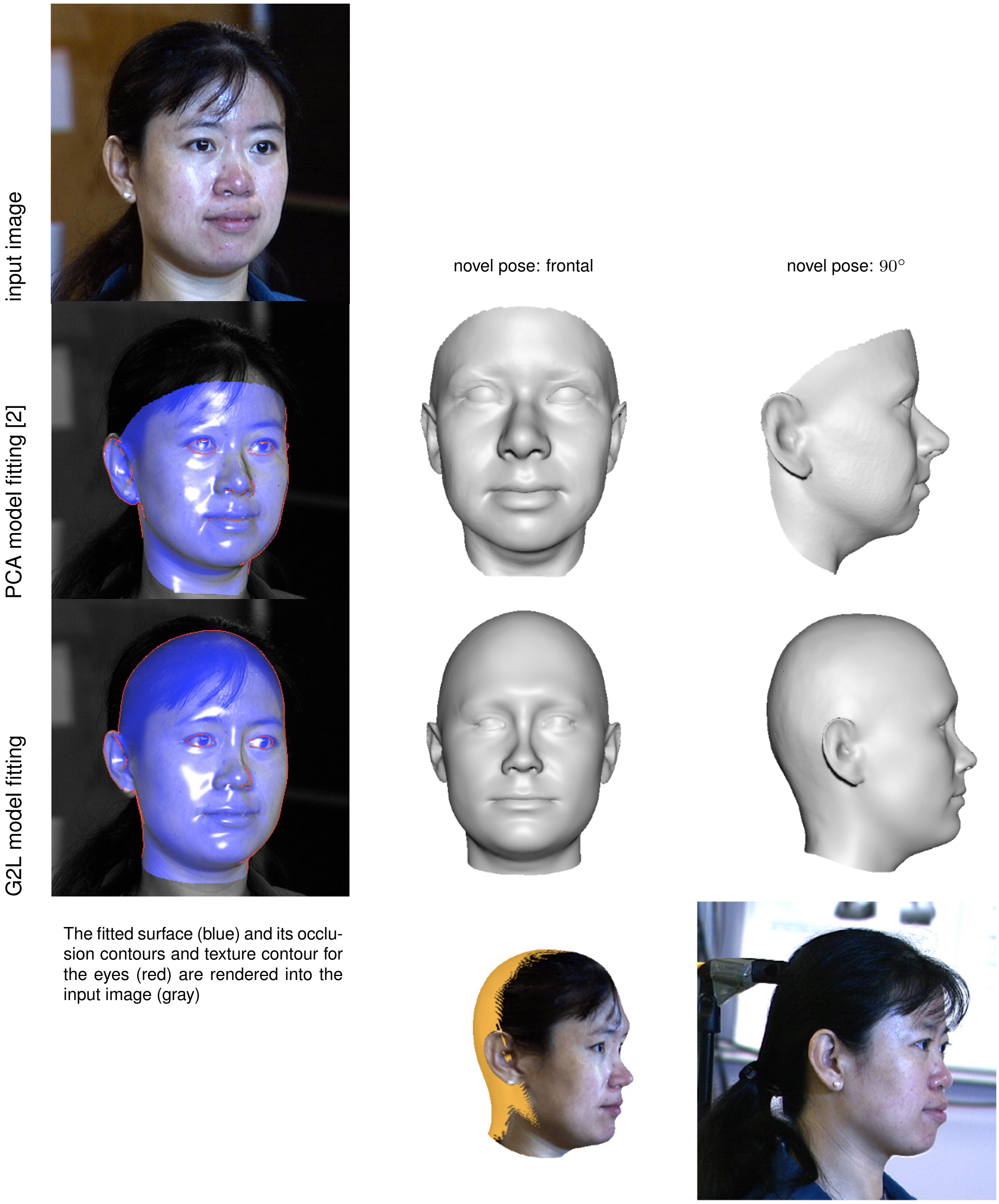
(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.13: The figure shows the G2L model (`UniBS-A`) fitted to a single input photo (a). The extracted texture (d,e) and the contour of the estimated shape rendered into the photo (b) demonstrate the highly accurate correspondence between estimated shape and input photo. The resulting shape (c) is very similar to the (registered) scan of the same individual (`PPY`), see Figure 2.3. In the $3rd$ row, we show a fitting of the same input image (a) with the Multi Feature Fitter [2] and the PCA model (`BFM`, trained from the same set of scans then the G2L model (`UniBS-A`). Figure (f) and (g) show the reconstructed shape in the fitted and the frontal pose, respectively. Figure (h) shows the extracted and estimated texture.



The fitted surface (blue) and its occlusion contours and texture contour for the eyes (red) are rendered into the input image (gray)

Figure 5.14: The PCA model (`BFM`, $2nd$ row) and the G2L model (`UniBS-B`, $3rd$ row) are fitted to the input image ($1st$ row, PIE `4007/27/07` [73]). In the $4th$ row, the fitted surface (G2L model) with the extracted texture is shown in novel pose and a photo of the same subject for comparison (PIE `4007/22/14`, this photo was not used for fitting).



The fitted surface (blue) and its occlusion contours and texture contour for the eyes (red) are rendered into the input image (gray)

Figure 5.15: The PCA model (`BFM`, $2nd$ row) and the G2L model (`UniBS-B`,$3rd$ row) are fitted to the input image ($1st$ row, PIE `4000/05/11` [73]). In the $4th$ row, the fitted surface (G2L model) with the extracted texture is shown in novel pose and a photo of the same subject for comparison (PIE `4000/22/22`, this photo was not used for fitting).

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we have presented a novel analysis method for training a statistical model from data.

The components of the well-known PCA model all have global support, i.e. each model coefficient affects the full shape. The G2L model is based on a multiresolution analysis of the shape and consists of a hierarchy of global and local components with limited support. The shape of different parts of the face can be modified without affecting the rest of the shape.

In order to evaluate the quality of the model, we have used a controlled fitting environment, where the surface of faces is reconstructed from a sparse set of 3D landmark points. We have demonstrated that, in contrast to the PCA model, the G2L model is able to locally adapt to the landmarks. At the same time, the G2L model has a slightly better generalization error than the PCA model.

We have also demonstrated that this model can be fitted to a single photograph and can reconstruct the 3D surface of a face. In the fitting system, first the contours are fitted. In a second step, the contours are kept fixed, and the remaining flexibility of the shape model is fitted using the color pixel intensities. We use a multiresolution approach for fitting that, in combination with an efficient Quasi-Newton method (L-BFGS), results in a run-time for the full fitting of $\sim 30\ s$ on standard hardware.

The G2L model has decisive advantages compared to the previous PCA model, in particular the model is locally more flexible and can therefore match features in the image much more accurately.

However, the current model has some restrictions. The most important challenges for future work are to develop a representation for facial expression, especially those with open mouth and eye movement, as well as hairstyle, facial hair like beards, and texture details like freckles, stubble, wrinkles, birthmarks and scars. The second challenge is to fit these models to images. Also methods to improve the robustness of the fitting are needed. In the following sections, some approaches to tackle these tasks are discussed.

6.2 Expression Model

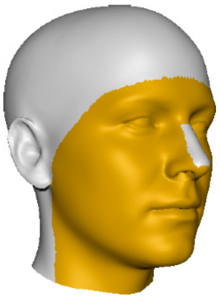
The model we have used in this work separates shape and surface color information from pose and illumination. However, the model does not separate identity and expression.

The statistical shape models used in this work are trained from faces of different individuals in neutral expression. Therefore, they can generate faces with different identities, but only faces with neutral expression. For this reason, these models can only be fitted to images with persons with neutral expression. Especially when the person smiles, smirks or opens the mouth, the fitting fails. Making the fitting robust with respect to facial expression would expand the application area of statistical face models dramatically and is one of the most important challenges for future work.

In [19] we have demonstrated that expression invariant 3D face recognition can be done by fitting an identity/expression separating PCA model to 3D shape data. We use one model for all types of expressions, and each set of coefficients describes one specific expression. The combined identity/expression model is then fitted to the data (3D or images) and the expression is normalized by removing the expression components.

For the `UniBS-B` dataset 709 scans are available where the subject shows one of the basic emotions joy, fear, disgust, anger, sadness and surprise [44], see Section 2.1.2. In [19] we presented a statistical expression model where the statistical model for the identity (i.e. the PCA model as in [19] or the G2L model) is expanded with components that describe the expression, see Figure 6.1. For each expression scan $\mathbf{x}_{i,r}$, $r \in \{\mathrm{joy}, \mathrm{fear}, \mathrm{disgust}, \mathrm{anger}, \mathrm{sadness}, \mathrm{surprise}\}$, the expression vector is calculated as the difference

$$\mathbf{y}_{i,r} = \mathbf{x}_{i,r} - \mathbf{x}_i \tag{6.1}$$



between the expression scan and the neutral scan $\mathbf{x}_i$ of that specific person. This approach shows the problem that due to deficits in the registration, unnatural movements of the shape of the back of the head occur. For this reason, we mask out regions of which we assume that they do not change due to expressions, see the mask $\mathbf{m}$ in the margin. Hence, the expression vector is given as

$$\mathbf{y}'_{i,r} = b(\mathbf{x}_{i,r}, \mathbf{x}_i, \mathbf{m}) - \mathbf{x}_i \tag{6.2}$$

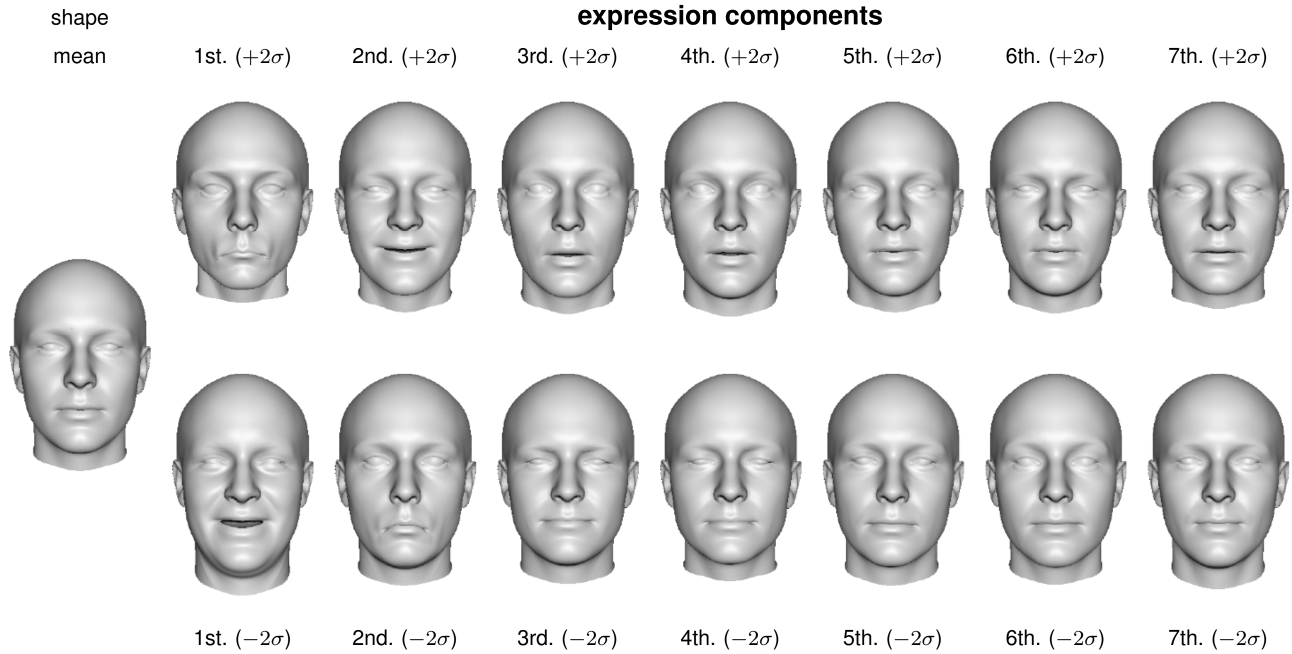


Figure 6.1: The mean of the `UniBS-B` data set (see Section 2.1.2) together with the variation of first seven expression components. The model is computed from 709 pairs of neutral and expression scans.

where b is the multi-scale blending using the Gaussian- and Laplacian Pyramids (similar to the texture blending in Appendix 7.3). These differences are already mode-centered, when the neutral expression is the natural mode of the expression data [19]. Therefore, it is not reasonable to subtract the arithmetic mean from expression vector $\mathbf{y}'_{i,r}$. For the expression, we use a simple PCA model, i.e. the m_{EXP} expression vector are stacked into the data matrix

$$\mathbf{Y}_{\text{EXP}} = [y'_1, \dots, y'_{m_{\text{EXP}}}] = \mathbf{U}_{\text{EXP}}\mathbf{W}_{\text{EXP}}\mathbf{V}^T_{\text{EXP}} \tag{6.3}$$

and decomposed with SVD. The outcome of this is the expression model

$$\mathcal{M}_{\text{EXP}} = (\boldsymbol{\mu}, \boldsymbol{\sigma}_{\text{EXP}}, \mathbf{U}_{\text{EXP}}). \tag{6.4}$$

An example of such an expression model can be seen in Figure 6.1. This model only modifies the expression of a face, but not the identity. A combined model for identity and expression

$$\mathcal{M}_{\text{G2LX}} = (\boldsymbol{\mu}, [\boldsymbol{\sigma}_{\text{G2L}}\boldsymbol{\sigma}_{\text{EXP}}], [\mathbf{U}_{\text{G2L}}\mathbf{U}_{\text{EXP}}]) \tag{6.5}$$

is therefore given by the combined model consisting of the components of the identity model and the components of the expression model.

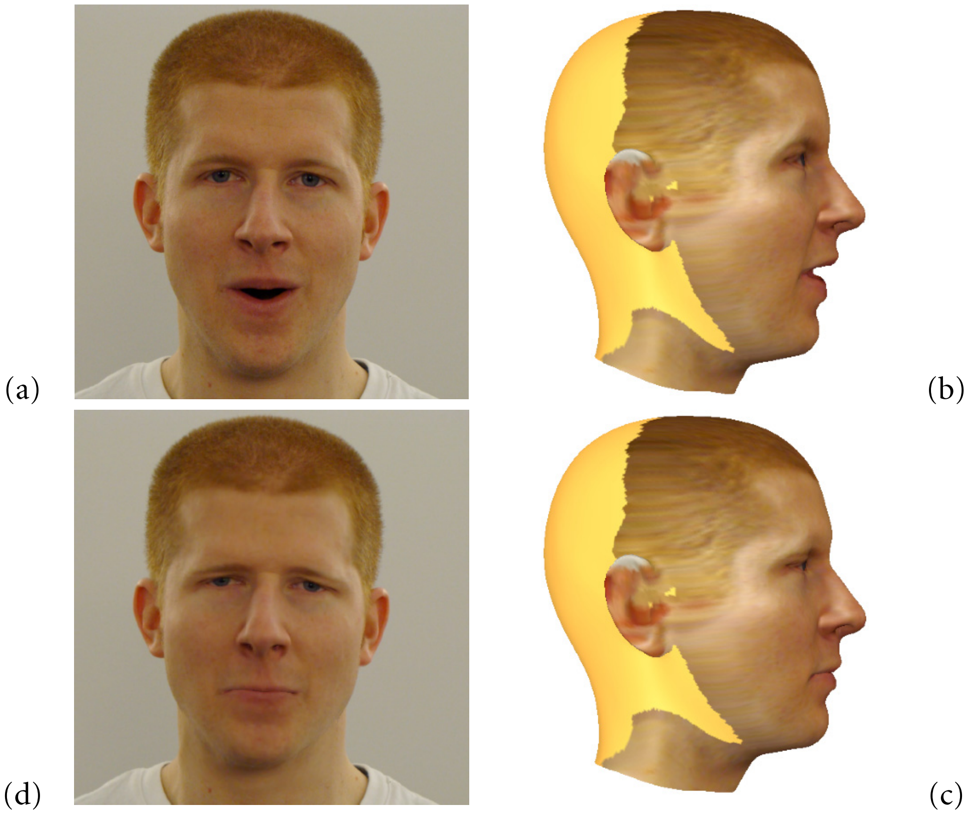


Figure 6.2: Using the expression model, the open mouth of the person in the input image (a) is correctly recovered, as it can be seen in the rendering in novel pose (b, with extracted texture). To fit the model (expression model and G2L model) to the image, a manually guided fitting system with 19 clicked landmark points and manually labeled contours was used [74]. By removing the expression components, the expression can be neutralized (c). Image (d) shows the input image (a) with rendered neutralized model (c).

Currently, we are not able to fit such a model automatically to an image. The main problem is that the fitting system does not open the mouth, even when components with open mouth are available. The reason for this is that the optimization starts with an initial guess with closed mouth and gets trapped in a local minimum where both the upper and the lower lip are assigned to the same contour.

As a proof of concept, in the master thesis of Felix Gorny we developed a manually guided fitting system [74]. In this work, 19 landmark points are used, in particular a landmark points at the upper lip and one at the lower lip, forcing the mouth to open. Moreover, the contours are defined manually, so that the upper and lower lip are fitted correctly, for an example see Figure 6.2.

6.3 Automatic Contour Extraction

A fitting algorithm should be fast, precise, robust and work fully automatically. In the following, we discuss how to improve the presented fitting system.

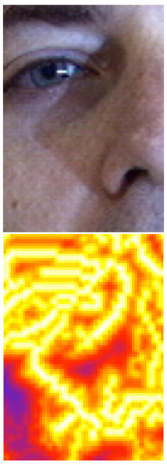
The presented fitting algorithm works very precise, i.e. if the features in the face are detected, the fitting algorithm can match them. At the same time, the fitting is very cost-efficient, we reduced the run time of the fitting algorithm from $\sim 70\ s$ reported in [2] to $\sim 30\ s$.

Further acceleration can be achieved easily by reducing the number of iterations (with more aggressive stop parameters in L-BFGS), using models with fewer vertices (e.g. level 6 with 23888 vertices instead of level 7 with 97577 vertices), evaluating the cost function and its gradient using a subset of the vertices or using fewer model components in each fitting module. However, all these approximations affect the accuracy of the model negatively and were therefore not an option for the work presented here.

The fitting system is implemented using C++, BLAS and OpenGL. In algorithms like the fitting system that require massive vector operations, using the GPU's ability to operate on large matrices in parallel can yield several orders of magnitude higher performance than an implementation on a conventional CPU. We did some early experiments implementing some of the derivatives of the fitting on the GPU using CUDA. The experiments have confirmed that the acceleration is significant, e.g. Tobias Maier achieved for the derivatives $\frac{\partial \vec{c}_l}{\partial c^i_d}$ (Equation 7.22) a speed-up of factor 17 (using an Intel®Core™2 Duo E8400 (3GHz) with a Geforce G8600 GT, speed-up including transfer time). However, developing on the GPU is quite time-consuming in terms of developing time, and therefore implementing the full fitting system on the GPU remains a long-term goal.

The fitting system does not work fully automatically, i.e. the user has to manually set five to seven landmark points in the image. We expect that in the future these landmarks can be detected automatically. M. Rätsch reports in his PhD [75] that he is able to detect the five inner feature points (eyes, nose, corner of the mouth) for faces in frontal pose using a variant of his face detector [76]. However, currently, this method only works robustly for frontal or near-frontal images.

The robustness of the fitting system depends on its weakest module. In particular, the preprocessing step of contour extraction offers the possibility to improve the robustness of the full fitting system. The distance transform (see Section 5.6.2) is based on contours detected with Canny edge detection. It works well to detect a wide range of edges in images. However, it is not specially designed to detect contours in face images. As a result, the distance transform contains many false positives and also false negatives, an example of a distance transform can be seen in Figure 6.3. At the eyes (similar things happen e.g. at the ears) multiple parallel edges are detected by the Canny edge detector and, depending on the initialization, it can happen that the model contour is fitted to the wrong edge.



One particular challenge is illustrated in the figure in the margin, where Canny edge detection also detects edges that are caused by cast shadows. One approach to overcome this is to classify the edges according to their being caused by a change in shading or by a change in albedo using classifiers like in [77].

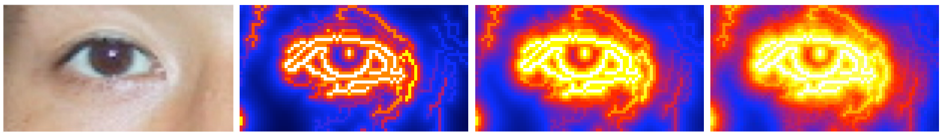

(a) (b) $\kappa = 10$ (c) $\kappa = 20$ (d) $\kappa = 30$

Figure 6.3: From the input image (a), the distance transform (Equation 5.39) can be computed using different thresholds.

A different solution could be to use prior knowledge from the model already in the edge detection process, e.g. by first fitting an Active Appearance Model [24, 66] to the input image. From this fitting we could directly derive the distance transform (and also landmark points).

The process of extracting edges from the image could also be avoided by learning locally ideal cost functions for face model fitting [59]. An interesting approach could be to combine their work with a 3D model.

In particular, this problem has to be solved when the model (and the expression model) should be fitted to images with persons showing expressions. When the mouth of the person is opened, it is important for the fitting to find the upper and lower contour of the lips and exclude edges caused by the teeth. It has to be assured that the upper (and lower) mouth line of the model comes into correspondence with the upper (and lower) mouth line detected in the image. This is in particular a challenge when the optimization is started with an initial guess with a closed mouth, since it easily happens that both mouth lines get fitted to either the upper or the lower mouth line. In the example fit in Figure 6.2 we solved this problem by using landmarks at the upper and lower lip, and therefore forced the mouth to open. When using model-based contour detection, the explicit correspondence has to be used in a similar way.

6.4 Statistical Color Models

In this work, we focused on statistical shape models only. Solely as an optional step in the fitting algorithm (Module 4: Surface Color from Pixel Intensity, see Section 5.6.4) we use a simple per-vertex color PCA model. An important question is, how the idea of the G2L model can be transferred to color models. In this section, we discuss some challenges in this process.

By using vertex color, fine details like freckles and stubble get lost, even though the resolution of the scanner is high enough to capture them (see Section 2.2). So, at first, it seems to be reasonable to use texture mapping and thereby increase the resolution of the color information (see Section 2.4.2). However, these fine details are not in correspondence, with the consequence that these details get lost when

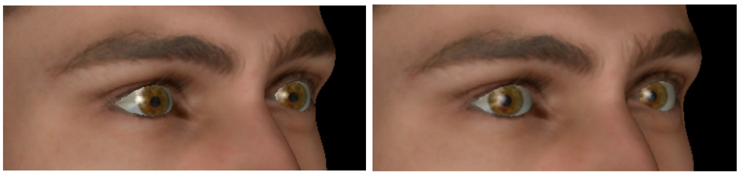


Figure 6.4: In the Bachelor Thesis of Christian Horisberger [78], we modeled 3D geometry of the human eye by a sphere with the texture map of the iris and sclera. Thus, the eyes can be rotated.

building affine combinations of multiple high resolution textures. One reason for the missing correspondence of these details is that color information is not used during the registration process. However, it is even unclear, how to define correspondence for fine details like freckles in a reasonable way.

For this reason, it makes little sense to use linear models (like the PCA model or the G2L model) for high resolution color data. One approach could be a model that is able to generate skin with freckles, birthmarks and scars, and that is trained from (skin) patches that are not in correspondence.

A similar challenge is a color model for the eyes: here, the registration does not take into account that the iris has to be round. In addition, for the eyes, there is no correspondence between shape and texture. In particular, this causes problems when the person moves the eyes to the right and left (the shape of the eyes in the template cannot be rotated). For this reason, all scanned persons look straight ahead, however, still the boundary of the iris in the mean is blurry. For the fitting, it would be preferable that the model can generate persons looking aside, e.g. by rotating the eyes explicitly in the 3D geometry as in Figure 6.4.

6.5 Color Correction

Usually, the visual quality of fittings can be dramatically improved by mapping the pixel intensities of the input image to the reconstructed surface (see Section 5.6.6). Since the color model is not able to represent all small details that are important to capture the individual albedo of a face (see above), with this model-free approach, photo-realistic renderings can be created.

In doing so, we are confronted with three challenges: Firstly, the input image is illuminated, i.e. the input images does not show the pure albedo, and shading and cast shadows are also responsible for the appearance. Secondly, the texture is incomplete, since we use a single image and parts of the head are occluded. Thirdly, some parts of the head might be occluded, e.g. by hair, and these artifacts are visible in the extracted texture.

To separate the pure albedo from the shading, in [1] it is suggested to correct the extracted texture by the difference between the estimated model color (not

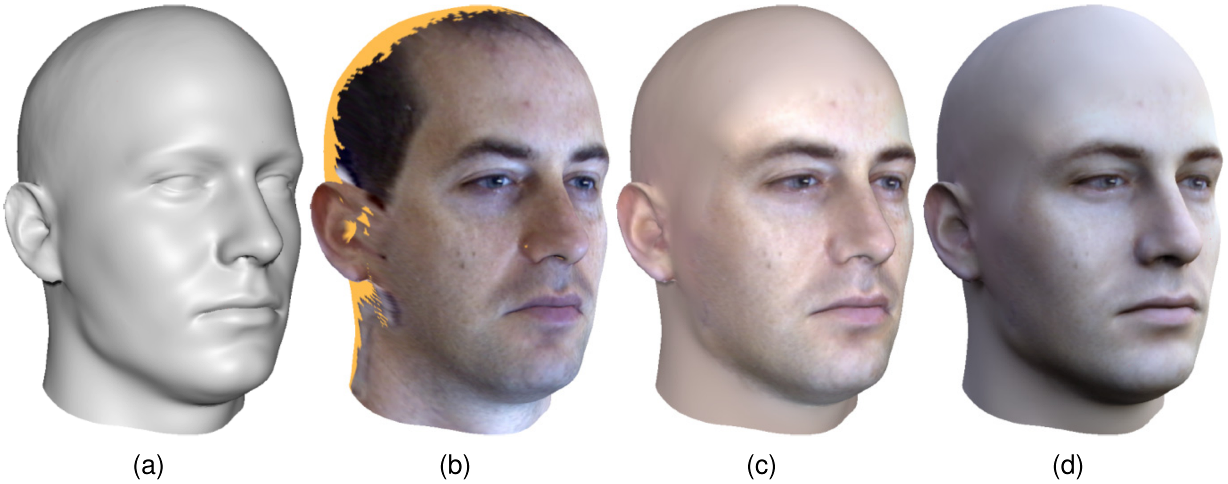
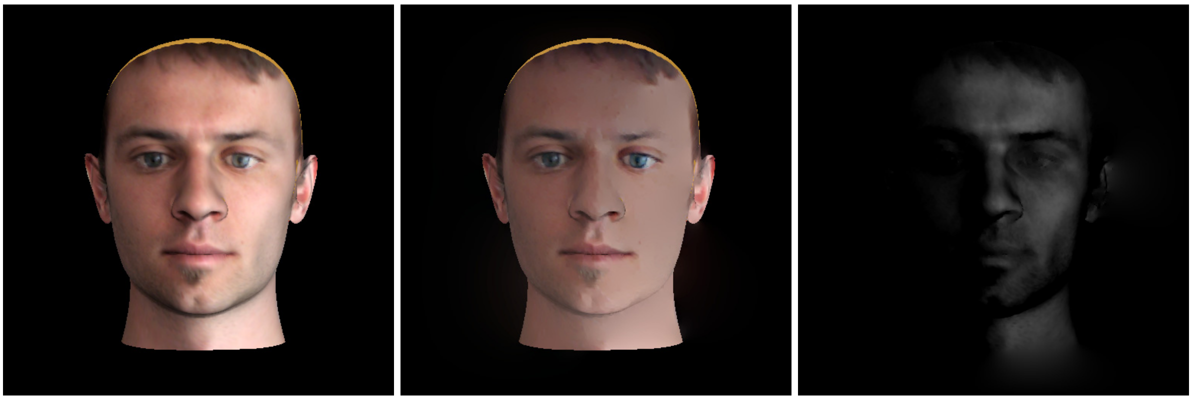


Figure 6.5: For the reconstructed surface (a), the color of the input photo (see Figure 5.14) is mapped to the surface, where occluded regions are filled in with yellow (b). This color is corrected by the difference between the model color and the illuminated model color (c). Missing regions and the area outside the facial mask is replaced by the estimated model color. This surface color can be used to re-illuminate the scene using a more complex global illumination model (d). Here, environmental mapping using a light probe from [79] was used.

illuminated) and the illuminated model color, an example can be seen in Figure 6.5. However, this correction shows only good results when the illumination is well estimated. This is only the case when the shortcoming caused by simplifications of the illumination model (in particular that the illumination model uses ambient light and one directed light source, see Section 5.1.3) are not too severe. In [77] a method was proposed to decompose an image into two intrinsic images, one for the reflectance (albedo) and one for the shading. In the master thesis of Marcel Arheit [80], we tried to apply this method to face images, result are shown in Figure 6.6.

To fill in missing regions in the extracted texture, the estimated model color can be used, see Figure 6.5 for an example. However, this works only well for persons in frontal or near frontal pose. Since the model color is only given in low resolution, freckles and stubble is missing on the other side. One approach could be to mirror the extracted texture. However, in doing so, all birthmarks, scars and wrinkles are reproduced symmetrically which is not correct. Another approach is to estimate the missing texture parts using super-resolution or image analogies [81]. For regions where the high-resolution texture is missing, at least the low-resolution surface color is available (from the model). The high-resolution texture for these regions is estimated based on corresponding patches of low-resolution surface color and extracted high-resolution texture from the visible areas of the surface. In [82] Christian Horisberger is investigating in this approach.

In this chapter, we discussed some topics for future work. All of them are mo-



(a) (b) (c)

Figure 6.6: The extracted color mapped to the reconstructed surface (a) is decomposed into two intrinsic images, one for the reflectance (b) and one for the shading [80].

tivated by practical problems, in particular with the goal to make the fitting system more robust and reliable. Developing a fitting system that works well on a wider range of input images would increase the application area. However, theoretical questions follow, e.g. how to compute statistical models that do not require (local) correspondence of the training data (like a model for skin). This is in particular a challenge, when this model-based approach is transferred to other object classes than face.

Chapter 7

Appendix

7.1 Derivatives

7.1.1 Derivatives of the Camera Model

In this section we give the partial derivatives of the image projection with respect to parameters of the camera model $\vec{\rho}$. We start with the 3D translation $\vec{t} = (t_x t_y t_z)^T$.

$$\frac{\partial P_x}{\partial t_j} = -\frac{w}{2}\frac{\mathfrak{n}f}{r_0}(\delta_{xj}e_z - \delta_{zj}e_x)/e_z^2; \quad j \in \{x, y, z\} \tag{7.1}$$

$$\frac{\partial P_y}{\partial t_j} = -\frac{h}{2}\frac{\mathfrak{n}f}{t_0}(\delta_{yj}e_z - \delta_{zj}e_y)/e_z^2; \quad j \in \{x, y, z\} \tag{7.2}$$

The partial derivatives with respect to the rotation angles γ, φ and ψ are given by

$$\frac{\partial P_x}{\partial \theta} = -\frac{w}{2}\frac{\mathfrak{n}f}{r_0}\sum_{k\in\{x,y,z\}} x_k \left[\left(\frac{\partial \mathbf{R}}{\partial \theta}\right)_{xk}\frac{1}{e_z} - \left(\frac{\partial \mathbf{R}}{\partial \theta}\right)_{zk}\frac{e_x}{e_z^2}\right] \qquad \theta \in \{\gamma, \varphi, \psi\} \tag{7.3}$$

$$\frac{\partial P_y}{\partial \theta} = -\frac{h}{2}\frac{\mathfrak{n}f}{t_0}\sum_{k\in\{x,y,z\}} x_k \left[\left(\frac{\partial \mathbf{R}}{\partial \theta}\right)_{yk}\frac{1}{e_z} - \left(\frac{\partial \mathbf{R}}{\partial \theta}\right)_{zk}\frac{e_y}{e_z^2}\right] \qquad \theta \in \{\gamma, \varphi, \psi\} \tag{7.4}$$

with

$$
\begin{aligned}
\frac{\partial \mathbf{R}}{\partial \psi} &= \begin{pmatrix} 0 & -\cos\psi\sin\varphi\cos\gamma + \sin\psi\sin\gamma & \sin\psi\sin\varphi\cos\gamma + \cos\psi\sin\gamma \\ 0 & -\cos\psi\sin\varphi\sin\gamma + \sin\psi\cos\gamma & \sin\psi\sin\varphi\sin\gamma - \cos\psi\cos\gamma \\ 0 & \cos\psi\cos\varphi & -\sin\psi\cos\varphi \end{pmatrix} \\
\frac{\partial \mathbf{R}}{\partial \varphi} &= \begin{pmatrix} -\sin\varphi\cos\gamma & -\sin\psi\cos\varphi\cos\gamma & -\cos\psi\cos\varphi\cos\gamma \\ -\sin\varphi\sin\gamma & -\sin\psi\cos\varphi\sin\gamma & -\cos\psi\cos\varphi\sin\gamma \\ \cos\varphi & -\sin\psi\sin\varphi & -\cos\psi\sin\varphi \end{pmatrix} \\
\frac{\partial \mathbf{R}}{\partial \gamma} &= \begin{pmatrix} -\cos\varphi\sin\gamma & \sin\psi\sin\varphi\sin\gamma - \cos\psi\cos\gamma & \cos\psi\sin\varphi\sin\gamma + \sin\psi\cos\gamma \\ \cos\varphi\cos\gamma & -\sin\psi\sin\varphi\cos\gamma - \cos\psi\sin\gamma & -\cos\psi\sin\varphi\cos\gamma + \sin\psi\sin\gamma \\ 0 & 0 & 0 \end{pmatrix}
\end{aligned}
$$

The partial derivatives with respect to the focal length f is given by

$$\frac{\partial P_x}{\partial f} = -\frac{w}{2}\frac{\mathfrak{n}e_x}{r_0 e_z}; \qquad \frac{\partial P_y}{\partial f} = -\frac{h}{2}\frac{\mathfrak{n}e_y}{t_0 e_z} \tag{7.5}$$

and finally the partial derivatives with respect to the offset $\vec{o} = (o_x\ o_y)^T$ are trivial.

$$\frac{\partial P_x}{\partial o_x} = 1; \qquad \frac{\partial P_y}{\partial o_x} = 0; \qquad \frac{\partial P_x}{\partial o_y} = 0; \qquad \frac{\partial P_y}{\partial o_y} = 1 \tag{7.6}$$

7.1.2 Derivatives of the Shape Model

In a linear model with n' components, each vertex $\vec{x}_i$ of the model is given as the linear combination, see Equation 3.3

$$\vec{x}_i = \vec{\mu}_i + \mathbf{S}_i \boldsymbol{\alpha} \tag{7.7}$$

where $\vec{\mu}_i \in \mathbb{R}^3$ is the 3D position of the mean for the ith vertex and $\mathbf{S}_i \in \mathbb{R}^{3\times n'}$ denote the corresponding rows of the weighted linear model matrix

$$\mathbf{S} = \mathbf{U}\mathrm{diag}(\sigma_1 \ldots \sigma_{n'})\ . \tag{7.8}$$

We pick one particular vertex and drop the index i. Thus one entry x_j of the vertex $\vec{x} = (x_x\ x_y\ x_z)^T$ is written as

$$x_j = \mu_j + \sum_k S_{jk}\alpha_k \tag{7.9}$$

and hence one entry e_j of the vertex in eye coordinates can be written as

$$e_j = \left[\mathbf{R}\vec{x} + \vec{t}\right]_j = \sum_k R_{jk}x_k + t_j = \sum_k R_{jk}\left(\mu_k + \sum_l S_{kl}\alpha_l\right) + t_j \tag{7.10}$$

$$\begin{aligned} \frac{\partial P_x}{\partial \alpha_r} &= -\frac{w}{2}\frac{\mathfrak{n} f}{r_0}\left(\frac{\partial e_x}{\partial \alpha_r} e_z - \frac{\partial e_z}{\partial \alpha_r} e_x\right) / e_z^2 && (7.11) \\ &= -\frac{w}{2}\frac{\mathfrak{n} f}{r_0}\sum_j S_{jr}\left(R_{xj}\frac{1}{e_z} - R_{zj}\frac{e_x}{e_z^2}\right) && (7.12) \\ \frac{\partial P_y}{\partial \alpha_r} &= -\frac{h}{2}\frac{\mathfrak{n} f}{t_0}\left(\frac{\partial e_y}{\partial \alpha_r} e_z - \frac{\partial e_z}{\partial \alpha_r} e_y\right) / e_z^2 && (7.13) \\ &= -\frac{h}{2}\frac{\mathfrak{n} f}{t_0}\sum_j S_{jr}\left(R_{yj}\frac{1}{e_z} - R_{zj}\frac{e_y}{e_z^2}\right) && (7.14) \end{aligned}$$

7.1.3 Derivatives of the Illumination Model

In this subsection, the partial derivatives of the rendered image $\vec{I}$ (i.e. the color intensities of the ith vertex, we drop the vertex index here) with respect to the paramters of the illumination model (see Table 5.2) are given. The derivatives for the color gain are given by

$$\frac{\partial \vec{I}}{\partial c_g^i} = \vec{\epsilon}_i \left[\mathbf{M}_c \begin{pmatrix} c_l^R \\ c_l^G \\ c_l^B \end{pmatrix}\right]; \qquad i \in \{R, G, B\} \tag{7.15}$$

where here $\vec{\epsilon}_i$ is the ith unit vector, i.e. $\vec{\epsilon}_R = (1,0,0)^T$ etc. The derivatives for the color contrast is given by

$$\frac{\partial \vec{I}}{\partial c} = \begin{pmatrix} c_g^R & & \\ & c_g^G & \\ & & c_g^B \end{pmatrix} \frac{\partial}{\partial c}\mathbf{M}_c \vec{c}_l \tag{7.16}$$

$$\text{with} \qquad \frac{\partial}{\partial c}\mathbf{M}_c = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} - \begin{pmatrix} 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \end{pmatrix} \tag{7.17}$$

and for the color offset by

$$\frac{\partial \vec{I}}{\partial c_o^i} = \vec{\epsilon}_i; \qquad i \in \{R, G, B\}\,. \tag{7.18}$$

The derivatives with respect to the ambient and diffuse light color ($\vec{c}_a, \vec{c}_d$) and the direction ($\vec{l}$) of the light have the form

$$\frac{\partial \vec{I}}{\partial \xi} = \begin{pmatrix} c_g^R & & \\ & c_g^G & \\ & & c_g^B \end{pmatrix} \mathbf{M}_c \begin{pmatrix} \frac{\partial c_l^R}{\partial \xi} \\ \frac{\partial c_l^G}{\partial \xi} \\ \frac{\partial c_l^B}{\partial \xi} \end{pmatrix} \tag{7.19}$$

where the derivatives for not corresponding ambient and diffuse color, resp., are zero

$$\frac{\partial c_l^i}{\partial c_a^j} = \frac{\partial c_l^i}{\partial c_d^j} = 0; \qquad i, j \in \{R, G, B\}, i \neq j \tag{7.20}$$

and the derivatives for the ambient color are given by

$$\frac{\partial c_l^i}{\partial c_a^i} = a^i; \qquad i \in \{R, G, B\} \tag{7.21}$$

and the derivatives for the diffuse color are given by

$$\frac{\partial c_l^i}{\partial c_d^i} = a^i \langle \hat{n}', \hat{l} \rangle + s \langle \hat{r}, \hat{v} \rangle^{\nu}; \qquad i \in \{R, G, B\} \tag{7.22}$$

and the derivatives for the light direction are given by

$$\frac{\partial \vec{c}_l}{\partial l_k} = \mathrm{diag}(\vec{a}) \cdot \vec{c}_d \langle \hat{n}', \frac{\partial \hat{l}}{\partial l_k} \rangle + \tag{7.23}$$

$$s \vec{c}_d \nu \langle r, \hat{v} \rangle^{\nu - 1} \left\langle \langle \hat{n}', \frac{\partial \hat{l}}{\partial l_k} \rangle \hat{n}' - \frac{\partial \hat{l}}{\partial l_k}, \hat{v} \right\rangle; \; k \in \{x, y, z\} \tag{7.24}$$

$$\text{where} \qquad \frac{\partial \hat{l}}{\partial l_k} = \widetilde{L}(\delta_{ik} - \widetilde{L}^2 l_i l_k) \tag{7.25}$$

$$\text{with} \qquad \widetilde{L} = \frac{1}{\sqrt{l_x^2 + l_y^2 + l_z^2}}. \tag{7.26}$$

7.1.4 Derivatives of the Surface Color Model

Similar to the shape model, for a vertex i, the surface color is given as the linear combination

$$\vec{c}_{a,i} = \vec{\mu}_{t,i} = \mathbf{B}_i \boldsymbol{\lambda} \tag{7.27}$$

where $\mathbf{B}_i \in \mathbb{R}^{3\times n'_a}$ are the rows of the weighted model matrix $\mathbf{B} = \mathbf{U}_a \mathrm{diag}(\sigma_{a,1} \dots \sigma_{a,n'_a})$ for vertex i (and again, we drop the index i). The derivatives of the image color with respect to the texture model coefficients $\frac{\partial \vec{I}}{\partial \beta}$ have the form of Equation 7.19 with

$$\frac{\partial \vec{c}_l}{\partial \beta_k} = \mathrm{diag}(\vec{b}_k)(\vec{c}_a + \vec{c}_d \langle \hat{n}', \hat{l} \rangle) \tag{7.28}$$

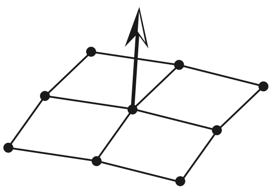
where $\vec{b}_k \in \mathbb{R}^3$ is the k-th column of $\mathbf{B}_i$.

7.1.5 Derivatives of the Surface Normals

In this section we will derive the derivatives of the surface normals. We will use them to compute the derivatives of the illumination model in the next section.

We assume that the surface normals for each vertex $\vec{x}_i$ are computed given the neighboring vertices:

$$\hat{n} = \frac{(\vec{x}_r - \vec{x}_l) \times (\vec{x}_t - \vec{x}_b)}{\|(\vec{x}_r - \vec{x}_l) \times (\vec{x}_t - \vec{x}_b)\|} = \frac{\vec{n}}{\|\vec{n}\|} \tag{7.29}$$



where $\vec{x}_r, \vec{x}_l, \vec{x}_t, \vec{x}_b$ are the neighboring vertices in the right, left, top and bottom direction, resp., using the regular structure of the mesh, as shown in the figure. Writing each of the vertices as a linear combination (see Equation 7.7) and using the abbreviations for horizontal (H) and vertical (V) pairs of vertices

$$\vec{\mu}^H = \vec{\mu}_r - \vec{\mu}_l;\ \vec{\mu}^V = \vec{\mu}_t - \vec{\mu}_b; \qquad \vec{\mu}^H, \vec{\mu}^V \in \mathbb{R}^3 \tag{7.30}$$

$$\mathbf{S}^H = \mathbf{S}_r - \mathbf{S}_l;\ \mathbf{S}^V = \mathbf{S}_t - \mathbf{S}_b; \qquad \mathbf{S}^H, \mathbf{S}^V \in \mathbb{R}^{3\times n'} \tag{7.31}$$

we can write the unnormalized direction of the surface normal as

$$\vec{n} = \vec{\mu}^H \times \vec{\mu}^V + (\mathbf{S}^H \boldsymbol{\alpha}) \times \vec{\mu}^V + (\mathbf{S}^V \boldsymbol{\alpha}) \times \vec{\mu}^H + (\mathbf{S}^H \boldsymbol{\alpha}) \times (\mathbf{S}^V \boldsymbol{\alpha}) \tag{7.32}$$

in order to compute the derivative, we can simplify this to

$$\begin{aligned} \vec{n} \ &= \ \vec{\mu}^H \times \vec{\mu}^V + (\mathbf{S}^H \boldsymbol{\alpha}) \times \vec{\mu}^V - (\mathbf{S}^V \boldsymbol{\alpha}) \times \vec{\mu}^H + \\ &\qquad \epsilon_{i,j,k} \left(\sum_{l<m} \left[S^H_{jl} S^V_{km} + S^H_{jm} S^V_{kl} \right] \boldsymbol{\alpha}_l \boldsymbol{\alpha}_m + \sum_l S^H_{jk} S^V_{kl} \boldsymbol{\alpha}_l^2 \right) \end{aligned} \tag{7.33}$$

where $\epsilon_{i,j,k}$ is the Levi-Civita Symbol. As usual, S_{ij} denotes the (i,j)th entry of the matrix $\mathbf{S}$. The derivative of the normal can now be computed by

$$\begin{aligned} \frac{\partial n}{\partial \alpha_s} \ &= \ \vec{s}^H_s \times \vec{\mu}^V + \vec{s}^V_s \times \vec{\mu}^H + \\ &\qquad \epsilon_{i,j,k} \left(\sum_{l \neq s} \left[S^H_{jl} S^V_{ks} + S^H_{js} S^V_{kl} \right] \alpha_l + \sum S^H_{js} S^V_{ks} \alpha_s \right) \end{aligned} \tag{7.34}$$

where $\vec{s}_i = (s_{1j}\; s_{2j}\; s_{3j})^T$ denotes the ith column of matrix $\mathbf{S}$. However, we need to consider the normalized normal

$$\hat{n} = \frac{1}{\|\vec{n}\|}\vec{n} = \frac{1}{\sqrt{n_1^2 + n_2^2 + n_3^2}}\vec{n} \tag{7.35}$$

and the derivative of the normalized normal can be computed using the derivative of the normal by

$$\frac{\partial \hat{n}_i}{\partial \alpha_s} = \frac{1}{\|\vec{n}\|}\left(\frac{\partial n_i}{\partial \alpha_s} - \frac{1}{\|n\|^2} n_i \langle \vec{n}, \frac{\partial n}{\partial \alpha_s}\rangle\right) \tag{7.36}$$

7.1.6 Derivatives of the Shape Model (Color Intensity)

The derivatives of the image color with respect to the shape model coefficients $\frac{\partial \vec{I}}{\partial \alpha}$ have the form of Equation 7.19 with (we omit the color channel i here)

$$\frac{\partial c_l}{\partial \alpha_k} = ac_d\langle \frac{\partial \hat{n}'}{\partial \alpha_k}, \hat{l}\rangle + sc_d\nu\langle \hat{r}, \hat{v}\rangle^{\nu-1}\left[\langle \frac{\partial \hat{r}}{\partial \alpha_k}, \hat{v}\rangle + \langle ., \frac{\partial \hat{v}}{\partial \alpha_k}\rangle\right]\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{7.37}$$

where $\langle ., \frac{\partial \hat{v}}{\partial \alpha_k}\rangle$ is neglected, as in [38], and the derivative of the reflectance ray $\hat{r}$ (Equation 5.15) is given by

$$\frac{\partial \hat{r}}{\partial \alpha_k} = 2\left[\langle \frac{\partial \hat{n}'}{\partial \alpha_k}, \hat{l}\rangle \hat{n}' + \langle \hat{n}', \hat{l}\rangle \frac{\partial \hat{n}'}{\partial \alpha_k}\right] \tag{7.38}$$

and the derivative of the surface normal in eye coordinates

$$\frac{\partial \hat{n}'}{\partial \alpha_k} = \mathbf{R}\frac{\partial \hat{n}}{\partial \alpha_k} \tag{7.39}$$

is computed by rotating the derivatives of the (normalized) surface normal $\frac{\partial \hat{n}}{\partial \alpha_k}$, Equation 7.36.

7.2 Computation of the Segments

In Section 3.4 segments are used to mask the detail components $\mathbf{y}^l$. On different levels of detail it is reasonable to use a different segmentation. In principle, we could define these segments manually. Here, we describe how we obtained these segments automatically. For this purpose, we use a variant of the K-means clustering. K-means partitions a dataset of observations of a random variable into K clusters. For instance, it can be used to segment an image into regions with homogeneous appearance [83]. Our goal is similar, however, we want to segment the vertices of the 3D shape into regions where the vertices are correlated. For this reason, we modify the K-means clustering by exchanging the way, the similarity is computed.

The data set $\{\mathbf{x}_1, \dots \mathbf{x}_n\}$ contains the 3D surfaces of n faces. We now consider this as a collection of m 3-dimensional random variables $\vec{x}_i$

$$\mathbf{x} = (\vec{x}_1, \dots, \vec{x}_m) \tag{7.40}$$

and the dataset contains n observations for each random variable.

The goal is to partition the vector $\mathbf{x}$ into K clusters, such that the variables inside a cluster have a larger inter-variable covariance compared with the variables outside the cluster.

Similar to K-means [83], we need a set of 3-dimensional vectors $\vec{\mu}_j$, where $1 \leq j \leq K$, in which $\vec{\mu}_j$ is the prototype for cluster j. Let $m_{i,j} \in \{0, 1\}$ be a binary indicator variable (where $1 \leq j \leq K$), and $m_{i,j} = 1$ if variable i is assigned to cluster j and $m_{i,j} = 0$ otherwise.

To compute the clustering, a two stage optimization is used: First, the binary indicator variables $m_{i,j}$ for the segments are updated

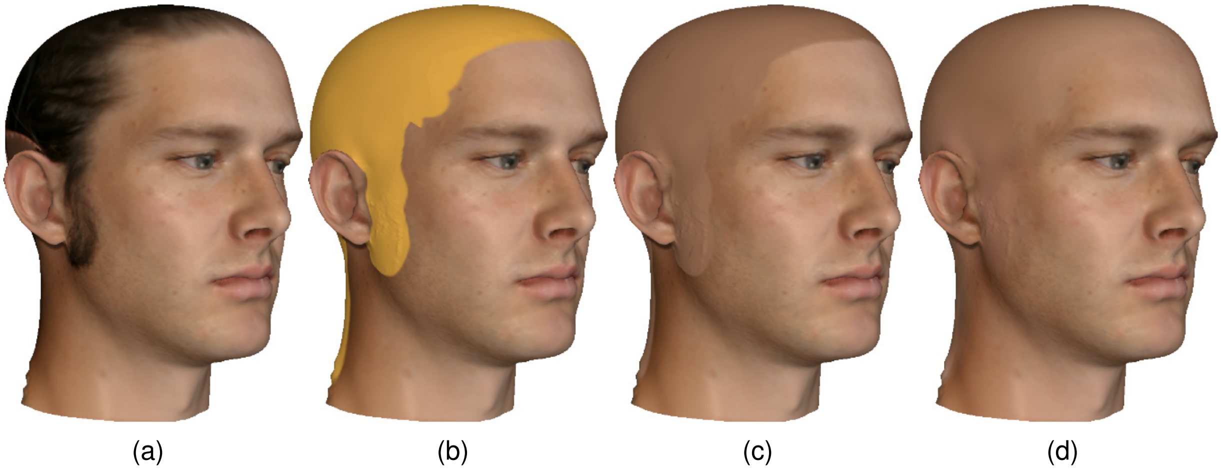
$$m_{i,j} = \begin{cases} 1 & \text{if } j = \arg\min_l \operatorname{cov}(\vec{x}_i, \vec{\mu}_l) \\ 0 & \text{otherwise} \end{cases} \tag{7.41}$$

then the prototype for clusters $\vec{\mu}_j$ are updated

$$\vec{\mu}_j = \frac{1}{\sum_i m_{i,j}} \sum_i m_{i,j} \vec{x}_i \, . \tag{7.42}$$

This process is repeated until convergence. K-means clustering does not guaranty that the clusters or patched are connected, and in fact, neigher connection nor neighborhood information is even used. Each variable is treated independently. For this reason, is sometimes happens (in particular for higher levels) that the patches are not connected, a fact that we want to avoid for the G2L model.

Thus, we split the K-means clusters into segments that are not connected. The outcome of the K-means clustering is not unique, it depends on the random initialization and the number of segments K. However, at least for level $\ell = 2$, in most cases the algorithm converges to the same clustering. In Figure 3.11 the segmentation computed by K-means clustering is shown for level $\ell = 2, 3$ and 4.



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

Figure 7.1: The texture extracted by the scanner contains hair and artifacts (a). These are masked out manually (b) and filled with the average color (c). To avoid edges, the two textures are blended using Gaussian and Laplacian Pyramids (d).

7.3 Reconstruction and Blending of the Texture

To blend two images, three pyramids are used, one Laplacian Pyramid for each of the two input images and one Gaussian Pyramid for the blend mask. Now, each level of the pyramids is blended, one level at a time. Finally the blended Laplacian Pyramid is collapsed, giving the result. Using this multiresolution blending, image features are blended across a transition zone that is proportional in size to the spatial frequency of the features [48].

We have used this technique to reconstruct the color of the skin for the back of the head as shown in Figure 7.1. The shape of hair can not be captured by the scanner and the shape of the head is estimated during the registration. To have a complete texture, the hair and other artifacts are manually removed from the captured textures and the color of the skin is estimated by the average color of the texture. When the six texture maps of the mesh (see Section 2.4.2) are blended one at the time, artifacts at the boundaries of the charts appear. For this reason it is necessary to use Gaussian and Laplacian Pyramids that are implemented directly in the parametrization. These texture pyramids are implemented analogously to the Gaussian and Laplacian Shape Pyramids discussed in Section 3.3.

Acknowledgments

First and foremost to all my colleagues and PhD students for countless discussions and their enthusiasm.

To Prof. Thomas Vetter for his ongoing support and encouragement.

To Brian Amberg for our shape retrieval experiments and the expression model and to Thomas Albrecht for our experiments on modeling the remaining flexibility.

To Pascal Paysan for scanning hundreds of people and to Brian Amberg using all the computers to put them in correspondence.

To Beat Röthlisberger for his help implementing the fitting system and calculating the derivatives.

To our bachelor- and master- students for their exceptional hard work and their contributions in their theses: To Marcel Arheit separating shading from reflectance, to Felix Gorny manually fitting the expression model, to Christian Horisberger for modeling the eyes, to Michael Keller for developing an early contour fitter, to Tobias Maier implementing parts of the fitter on the GPU and to Frank Preiswerk for fitting AAMs.

Thanks also to Thomas Albrecht, Brian Amberg, Lothar Knothe and Sandro Schönborn for carefully reading the manuscript.

And to the best parents a kid could hope for, Ingrid and Lothar... for everything.

This work was supported by a grant from the Swiss National Science Foundation 200021-103814.

Bibliography

[1] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3D faces,” in *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1999, pp. 187–194.

[2] S. Romdhani and T. Vetter, “Estimating 3D shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 986–993 vol. 2, June 2005.

[3] V. Blanz, K. Scherbaum, and H.-P. Seidel, “Fitting a morphable model to 3D scans of faces,” in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007.

[4] F. B. ter Haar and R. C. Veltkamp, “A 3D Face Matching Framework,” in *SMI'08, Shape Modeling International*, 2008, pp. 103–110.

[5] ——, “3D Face Model Fitting for Recognition,” in *ECCV 2008*, 2008.

[6] B. Allen, B. Curless, and Z. Popović, “The space of human body shapes: Reconstruction and parameterization from range scans,” *ACM Trans. Graph.*, vol. 22, no. 3, pp. 587–594, 2003.

[7] M. Fleute and S. Lavallee, “Building a complete surface model from sparse data using statistical shape models: Application to computer assisted knee surgery,” *Medical Image Computing and Computer-Assisted Intervention-MICCAI*, vol. 98, pp. 880–7.

[8] G. Zheng, K. Rajamani, and L. Nolte, “Use of a dense surface point distribution model in a three-stage anatomical shape reconstruction from sparse information for computer-assisted orthopaedic surgery: a preliminary study,” *ACCV*, vol. 6, pp. 52–60.

[9] G. Subsol, J. Thirion, and N. Ayache, “A scheme for automatically building three-dimensional morphometric anatomical atlases: application to a skull atlas,” *Medical Image Analysis*, vol. 2, no. 1, pp. 37–60, 1998.

[10] E. Dam, P. Fletcher, and S. Pizer, "Automatic shape model building based on principal geodesic analysis bootstrapping," *Medical Image Analysis*, 2007.

[11] D. Shen, E. Herskovits, and C. Davatzikos, "An adaptive-focus statistical shape model for segmentation and shape modeling of 3-D brain structures," *IEEE Transactions on Medical Imaging*, vol. 20, no. 4, pp. 257–270, 2001.

[12] T. Albrecht, M. Luthi, and T. Vetter, "A statistical deformation prior for non-rigid image and shape registration," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[13] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," in *EuroGraphics*, 2003, pp. 641–650.

[14] S. Romdhani, "Face image analysis using a multiple features fitting strategy," *Dissertation, Universität Basel*, 2005.

[15] J.-S. Pierrard and T. Vetter, "Skin detail analysis for face recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[16] D. A. Leopold, A. J. O'Toole, T. Vetter, and V. Blanz, "Prototype-referenced shape encoding revealed by high-level aftereffects," *Nature Neuroscience*, vol. 4, no. 1, pp. 89–94, 2001.

[17] V. Blanz and T. Vetter, "Face recognition based on fitting a 3D morphable model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1063–1074, 2003.

[18] B. Amberg, R. Knothe, and T. Vetter, "SHREC'08 entry: Shape based face recognition with a morphable model," in *SMI'08, Shape Modeling International*, New York, NY, USA, 2008.

[19] ——, "Expression invariant 3D face recognition with a morphable model," in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2008.

[20] "Face recognition as a search tool "Foto-Fahndung" final report," *Bundeskriminalamt, Wiesbaden*, 2007.

[21] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 399–458, 2003.

[22] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–591, Jun 1991.

[23] D. Cooper, T. Cootes, C. Taylor, and J. Graham, “Active shape models - their training and application,” *Computer Vision and Image Understanding*, no. 61, pp. 38–59, 1995.

[24] A. Lanitis, C. Taylor, and T. Cootes, “Automatic interpretation and coding of face images using flexible models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 743–756, Jul 1997.

[25] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[26] P. Penev and J. Atick, “Local feature analysis: A general statistical theory for object representation,” *Neural Systems*, vol. 7, pp. 477–500, 1996.

[27] R. Knothe, S. Romdhani, and T. Vetter, “Combining pca and lfa for surface reconstruction from a sparse set of control points,” in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2006, pp. 637–644.

[28] H. Schneiderman, “Learning statistical structure for object detection,” in *Images and Patterns (CAIP)*. Springer-Verlag, 2003.

[29] B. Heisele, T. Serre, and T. Poggio, “A component-based framework for face detection and identification,” *International Journal Computer Vision*, vol. 74, no. 2, pp. 167–181, 2007.

[30] C. Davatzikos, X. Tao, and D. Shen, “Hierarchical active shape models, using the wavelet transform,” *Medical Imaging, IEEE Transactions on*, vol. 22, no. 3, pp. 414–423, March 2003.

[31] A. Bhalerao and R. Wilson, “Local shape modelling using warplets,” *Lecture Notes in Computer Science*, vol. 3540, pp. 439–448, 2005.

[32] D. Nain, S. Haker, A. Bobick, and A. Tannenbaum, “Multiscale 3-d shape representation and segmentation using spherical wavelets,” *Medical Imaging, IEEE Transactions on*, vol. 26, no. 4, pp. 598–618, April 2007.

[33] R. Knothe, “Waveletbasierte Kompression des Morphable Models,” *Diplomarbeit, Universität Freiburg*, 2004.

[34] M. Lounsbery, T. D. DeRose, and J. Warren, “Multiresolution analysis for surfaces of arbitrary topological type,” *ACM Transactions on Graphics*, vol. 16, no. 1, pp. 34–73, 1997.

[35] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal processing using hidden markov models," *IEEE Transactions on Signal Processing*, 1997.

[36] V. Blanz and T. Vetter, "Reconstructing the complete 3D shape of faces from partial information," *it+ti Oldenburg Verlag*, vol. 44, no. 6, pp. 295–302, 2002.

[37] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3D face model for pose and illumination invariant face recognition," *6th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009.

[38] V. Blanz, "Automatische rekonstruktion der dreidimensionalen form von gesichtern aus einem einzelbild," *Dissertation, Eberhard-Karls-Universität Tübingen*, 2000.

[39] "USF humanid 3D face dataset, courtesy of Sudeep Sarkar, Univ. of South Florida, Tampa, 2005."

[40] C. Basso, P. Paysan, and T. Vetter, "Registration of expressions data using a 3D morphable model." in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2006, pp. 205–210.

[41] (2008) Cyberware website. [Online]. Available: http://www.cyberware.com/

[42] (2008) ABW-3D website. [Online]. Available: http://www.abw-3d.de/

[43] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid ICP algorithms for surface registration," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[44] P. Ekman, "Basic emotions," *Handbook of Cognition and Emotion, John Wiley & Sons, Ltd.*, 1999.

[45] M. Lüthi, A. Lerch, T. Albrecht, Z. Krol, and T. Vetter, "A hierarchical, multi-resolution approach for model-based skull-segmentation in MRI volumes," *Conference Proceedings 3D-Physiological Human*, December 2008.

[46] N. Litke, M. Droske, M. Rumpf, and P. Schröder, "An image processing approach to surface matching." in *Symposium on Geometry Processing, Vienna, Austria*, 2005, pp. 207–216.

[47] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. COM-31,4, pp. 532–540, 1983.

[48] (2008) hugin - panorama photo stitcher / enblend. [Online]. Available: http://hugin.sourceforge.net/, http://enblend.sourceforge.net/

[49] D. Zorin, P. Schröder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens, "Subdivision for modeling and animation," *SIGGRAPH 2000 Course notes*, 2000.

[50] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, vol. 10, no. 6, pp. 350–355, 1978.

[51] T. Akenine-Möller and E. Haines, "Real-time rendering," *A.K. Peters Ltd., Natick, Massachusetts*, 2002.

[52] C. M. Bishop, "Neural networks for pattern recognition," *Clarendon Press, Oxford*, 1995.

[53] C. Dorai and A. K. Jain, "COSMOS - a representation scheme for 3D free-form objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1115–1130, 1997.

[54] J. Nocedal and S. Wright, "Numerical optimization," *Springer, New York*, 2006.

[55] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes: The art of scientific computing," *Cambridge University Press*, 2007.

[56] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision," *Cambridge University Press*, 2004.

[57] D. Shreiner, M. Woo, J. Neider, and T. Davis, "Opengl(r) programming guide: The official guide to learning opengl, version 2," *Addison Wesley Longman, Reading, Massachusetts*, 2005.

[58] M. Segal and K. Akeley, "The opengl(r) graphics system: A specification," *Silicon Graphics, Inc.*, 2006.

[59] M. Wimmer, F. Stulp, S. Pietzsch, and B. Radig, "Learning local objective functions for robust face model fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1357–1370, Aug. 2008.

[60] M. Keller, R. Knothe, and T. Vetter, "3D reconstruction of human faces from occluding contours," *Proceedings of the Mirage 2007*, March 2007.

[61] J. J. Koenderink, "Solid shape," *The MIT Press*, 1990.

[62] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[63] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in c / the art of scientific computing," *Cambridge University Press*, 1999.

[64] M. J. Jones and T. Poggio, "Multidimensional morphable models: A framework for representing and matching object classes," *International Journal Computer Vision*, vol. 29, no. 2, pp. 107–131, 1998.

[65] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Math. Program.*, vol. 45, no. 3, pp. 503–528, 1989.

[66] B. Amberg, A. Blake, and T. Vetter, "On compositional image alignment, with an application to active appearance models," 2009.

[67] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.

[68] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," *Technical Report*, 2004.

[69] B. Amberg, A. Blake, A. Fitzgibbon, S. Romdhani, and T. Vetter, "Reconstructing high quality face-surfaces using model based stereo," *IEEE 11th International Conference on Computer Vision (ICCV)*, Oct. 2007.

[70] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[71] T. Albrecht, R. Knothe, and T. Vetter, "Modeling the remaining flexibility of partially fixed statistical shape models," in *2nd MICCAI Workshop on the Mathematical Foundations of Computational Anatomy*, 2008.

[72] S. Meyers, "Effective c++: 55 specific ways to improve your programs and designs," *Addison-Wesley Professional Computing*, 2005.

[73] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression (PIE) database of human faces," The Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-01-02, January 2001.

[74] F. Gorny, "Enhancing automatic fitting procedures of human heads by manual interaction," *Master thesis, Universität Basel*, 2009.

[75] M. Rätsch, "Wavelet frame accelerated reduced vector machine for efficient image analysis," *Dissertation, Universität Basel*, 2008.

[76] M. Rätsch, G. Teschke, S. Romdhani, and T. Vetter, "Wavelet frame accelerated reduced support vector machines," *Image Processing, IEEE Transactions on*, vol. 17, no. 12, pp. 2456–2464, Dec. 2008.

[77] M. Tappen, W. Freeman, and E. Adelson, "Recovering intrinsic images from a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1459–1472, Sept. 2005.

[78] C. Horisberger, "I see you - 3d head and gaze simulation for visual tracking," *Bachelor thesis, Universität Basel*, 2006.

[79] P. Debevec, "Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *ACM SIGGRAPH 2008 classes*, 2008.

[80] M. Arheit, "Separating shading from reflectance in face images," *Master thesis, Universität Basel*, 2009.

[81] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2001, pp. 327–340.

[82] C. Horisberger, "Texture synthesis for morhable face models," *Master thesis, Universität Basel*, 2009.

[83] C. M. Bishop, "Pattern recognition and machine learning," *Springer*, 2006.

Curriculum Vitae

persönliche Daten

- geboren am 11. Februar 1978 in Freiburg im Breisgau.
- Staatsangehörigkeit: deutsch
    - Vater: Dr. Lothar Knothe, Dipl.-Chem.
    - Mutter: Dr. med. Ingrid Knothe, Dipl.-Chem.

Schule

- 1984-1988: Johann-Peter-Hebel-Schule Gundelfingen
- 1988-1997: Albert-Schweitzer-Gymnasium Gundelfingen
- Juni 1997: Abitur - allgemeine Hochschulreife

1997/98: Zivildienst im St. Josefskrankenhaus Freiburg

Studium

- Oktober 1998 - März 2001: Grundstudium der Informatik mit Nebenfach Physik an der Albert-Ludwigs-Universität Freiburg
- April 2001 - August 2003: Hauptstudium der Informatik mit Nebenfach Mathematik an der Albert-Ludwigs-Universität Freiburg.
- Oktober 2001-März 2002: ERASMUS-Stipendium an der ETH Zürich.
- September 2003-Februar 2004: Diplomarbeit an der Universität Basel bei Herrn Prof. Thomas Vetter.
  Thema: Waveletbasierte Kompression des Morphable Models.
- März 2004: Diplom in Informatik der Albert-Ludwigs-Universität Freiburg.
- seit Juni 2004: wissenschaftlicher Assistent am Departement Informatik der Universität Basel bei Herrn Prof. Thomas Vetter.