Editing Faces in Videos

Inauguraldissertation
zur Erlangung der Würde eines Doktors der Philosophie.
Der Philosophisch-Naturwissenschaftlichen Fakultät der
Universität Basel
vorgelegt von

Brian Amberg

aus Deutschland
Basel, 2011

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät auf Antrag von

Prof. Dr. Thomas Vetter, Universität Basel, Dissertationsleiter
Dr. Andrew Fitzgibbon, Microsoft Research Cambridge, Korreferent

Basel, den 14.12.2010 Prof. Dr. Martin Spiess, Dekan

Contents

1 Introduction 1

Part I The Generative Face Model

2 Introducing 3D Morphable Models 7
1 Camera Model 7
2 Training Morphable Models 8

3 Establishing Correspondence: Registration 11
1 Introduction 11
2 A Deformation Energy, Useful for Mesh Editing 12
3 Registration 26
4 Conclusion 32

Part II Fitting a 3D Morphable Model to a Video Sequence

4 3D Morphable Model Fitting 39
1 Problem Formulation 39
2 Fitting Strategy 40
3 The Imaging Model 41
4 Modelling the Posterior 42
5 Optimization Method 48
6 Alternatives 48

5 Interactive Interest Point Tracking 51
1 Introduction 51

2 Method 52
3 Efficient Optimization 55
4 Occlusions 58
5 Candidate Selection 60
6 Accuracy 62
7 Feature Extraction 62
8 User Interface 63
9 Experiments 63
10 Conclusion 66

6 Modelling the background for accurate shape extraction 69
1 Why should one use a silhouette term? 69
2 How to detect the silhouette 70
3 Results 75
4 Conclusion 75

7 Feature Contour Detection with Active Appearance Models 79
1 Active Appearance Models 81
2 Fitting an Active Appearance Model 82
3 Efficient Composition 84
4 The Incremental Warp 86
5 Experiments 86
6 Conclusion 90

Part III Face Editing with 3D Morphable Models

8 Video Manipulation 95
1 Adding Virtual Objects to the Scene 95
2 Face Exchange 97
3 Expression Transfer 99
4 Identity Transfer 99

9 Conclusion and Future Work 103

Bibliography 105

Thanks

Here is the place to give a huge thank you to my wife, who did not only bear with me during these years and especially during the final push, but encouraged and helped me wherever possible. I know what I have found in you, and I am very happy about that.

Apart from that I am want to thank my supervisor Thomas Vetter for the impulses and the freedom to shape my research to my liking, and to Andrew Blake and Andrew Fitzgibbon for the great discussions and their corrections of my blatant stupidity. If there are parts of this thesis which are not cryptic, then it can be solely attributed to the education in clarity given to me by Andrew Blake.

And I guess this is also the place where one puts a quote displaying the wit and literacy of the writer, here it comes:

> "I love deadlines, they make such a nice whooshing noise as they pass by."
>
> Douglas Noël Adams, *11 March 1952, † 11 May 2001

Abstract

Editing faces in movies is of interest in the special effects industry. We aim at producing effects such as the addition of accessories interacting correctly with the face or replacing the face of a stuntman with the face of the main actor.

The system introduced in this thesis is based on a 3D generative face model. Using a 3D model makes it possible to edit the face in the *semantic* space of pose, expression, and identity instead of pixel space, and due to its 3D nature allows a modelling of the light interaction. In our system we first reconstruct the 3D face, which is deforming because of expressions and speech, the lighting, and the camera in all frames of a monocular input video. The face is then edited by substituting expressions or identities with those of another video sequence or by adding virtual objects into the scene. The manipulated 3D scene is rendered back into the original video, correctly simulating the interaction of the light with the deformed face and virtual objects.

We describe all steps necessary to build and apply the system. This includes registration of training faces to learn a generative face model, semi-automatic annotation of the input video, fitting of the face model to the input video, editing of the fit, and rendering of the resulting scene.

While describing the application we introduce a host of new methods, each of which is of interest on its own. We start with a new method to register 3D face scans to use as training data for the face model. For video preprocessing a new interest point tracking and 2D Active Appearance Model fitting technique is proposed. For robust fitting we introduce background modelling, model-based stereo techniques, and a more accurate light model.

— Chapter 1 —

Introduction

Editing faces in movies is of interest to the special effects industry, where one might want to replace the face of a stuntman with that of the main actor, or when doing lipsyncing, where it is necessary to manipulate the actor's face such that she seems to speak a different sentence. Performing such editing tasks directly in the image by 2D editing methods is a daunting task.

In this thesis a method to perform such manipulations in videos is presented. This method requires only a small amount of manual interaction and results in high quality output. Faces are edited by reconstructing the 3D shape, texture and lighting situation of all the frames of a video. The scene is described in terms of a generative face model. The face can then be manipulated in this low parametric model space consisting of a few thousand parameters, instead of in the high dimensional space of video pixels, consisting of millions of parameters. The model is constructed such that the parameters for lighting and camera, the pose of the face, the albedo of the face, the part of the shape which describes the identity, and the shape deformations describing the expressions are all separated. Each of these parameter sets can therefore be edited independently, such that we can keep pose and lighting and change only the expressions or the identity. The face generated by the edited parameters can then be seamlessly combined with the original sequence, because lighting and camera have been estimated. A 3D Morphable Model (3DMM) as described by Blanz and Vetter (1999) is used for face reconstruction and editing. We follow the approach to video editing introduced in Blanz et al. (2003) which consists of fitting the video, changing the model parameters and rendering the resulting 3D scene back into the video, but derive novel methods on the way which are also useful in other contexts. This includes new techniques for model learning, interest point tracking and model fitting.

The problem can be broken down into three parts, which are reflected in the structure of this thesis. First, we describe the generative model and how it is learned from examples. This happens in part one, where 3D Morphable Models are introduced. They are a generative description of faces and are learned from

3D shape and texture data. The novelty in this part is the registration algorithm, a nonrigid ICP method with a new deformation energy.

The second part of the problem is that of reconstructing the generating model parameters given a video sequence. It is addressed in part II, where we describe how to find the most likely model parameters given a video sequence. This part draws from the work of Blanz and Vetter (1999); Blanz et al. (2003); Romdhani. and Vetter (2005); and Knothe (2009). The number of parameters which need to be estimated for a video is immense, even though the 3D Morphable Model is a very compact representation of face images. And it is a very difficult problem, as the posterior which we maximize is non-convex, and defined in a high dimensional space. We approach this problem by guiding a local optimization with semi automatically annotated landmarks and automatically detected contours of the eyes, the nose, and the lips.

To this end, we introduce a fast and accurate interest point tracking method, which finds the position of landmark points that were manually marked in some frames in the full video. Setting the landmarks is the only manual work involved in our pipeline. This interaction can be performed conveniently, as the interest-point tracking method runs at interactive rates. Here, we extended the method of Buchanan and Fitzgibbon (2006) with a background model to give more reliable tracking results and introduce a more efficient search for the globally optimal track.

Extracting the contours of eyes, nose and lips is done fully automatically based on the landmarks and 2D appearance models of the respective parts of faces. The appearance models were trained from the same data used to learn the 3D model. The novelty here is an appearance model fitter which is nearly as fast as the fastest previously available method while not using the approximations which made previous fast methods brittle.

We are fitting the whole video simultaneously, while previous systems treated the frames independently. This allows us to perform what we would like to call ‘temporal, model based stereo’. We are observing a deforming face under different views. We reconstruct the shape of the face by modelling the probability of the deformation throughout the video and finding deformations which make the appearance of the same point in the face similar in all frames. This makes it possible to reconstruct the shape even without a texture model.

3D Morphable Model (3DMM) and Active Appearance Model (AAM) fitters have a tendency to shrink. That is, that the contour of the best fit is smaller than the true contour. We propose to model the whole video by simultaneously segmenting the video into foreground and background while fitting the face to improve the fitting results.

The third part of the problem is that of using the reconstructed scene to generate new videos. Part three describes how the model parameters are edited, and how the resulting 3D scene is merged back into the original video such that the resulting sequence looks natural. We show examples of adding virtual objects to the scene, and of pasting the expression, identity or complete face from one scene into another scene.

Related Work

In this introduction we mention only the literature which is related to the overall system. Throughout the thesis we will describe how each of the methods solving a partial problem is related to more specific prior work.

The system proposed here is most closely related to Blanz et al. (2003). Our system is marker-less. For situations where markers are feasible, Vlasic et al. (2005) proposed a system to transfer expressions between individuals using a bilinear representation of the space of identities and expressions. Wang et al. (2008, 2004) and Huang et al. (2004) used data from a high resolution, high frame rate 3D scanner to accurately track a deforming face, this type of input data could also be incorporated in our system to track the expressions of a secondary actor driving the expressions in the edited video. Byun (2007) also drives the expressions of one video with another without using markers, but uses only the contours of lips, eyes, eyebrows, and nostrils, to get the overall deformations. This system cannot account correctly for the light interaction.

Wang et al. (2004) considered the temporal dynamics of speech and expressions, which is not necessary in our context but will be of interest when extending the system to be used with a text to speech generator. We try to directly map the expressions from one face to another, Stoiber et al. (2010) proposed a different parametrization of expression space to make a virtual actor perform actions which are not similar deformations, but have a similar semantics to the actions of the driving actor.

— Part I —

The Generative Face Model

— What are 3D Morphable Models — Constructing the Face Space from Registered Examples — Establishing Correspondence Between 3D Face Scans, the Algorithm and the Necessary Tweaks —

— Chapter 2 —

Introducing 3D Morphable Models

A 3D Morphable Models (3DMM) as introduced in Blanz and Vetter (1999) is a generative face model consisting of linear models of shape $\mathcal{S}(\boldsymbol{\alpha})$ and texture $\mathcal{T}(\boldsymbol{\beta})$, with a Gaussian prior over the parameters. A face is described with a 3DMM as a three dimensional tessellated surface with an associated texture. It is represented as a reference shape and texture plus a linear combination of shape and texture offsets.

$$\mathcal{S}(\boldsymbol{\alpha}) = \boldsymbol{s} + \boldsymbol{S}\boldsymbol{\alpha} \qquad\qquad \mathcal{T}(\boldsymbol{\beta}) = \boldsymbol{t} + \boldsymbol{T}\boldsymbol{\beta} \tag{2.1}$$

where $\boldsymbol{s} \in \mathbb{R}^{3N_v}$ are the stacked vertices of the sample mean of the training faces and $\boldsymbol{S} \in \mathbb{R}^{3N_v \times N_s}$ are the N_s orthogonal directions of maximal variation of the offsets from the sample mean, which were observed in the training data. The albedo of the face is described in the same way by the stacked RGB color channels $\boldsymbol{t} \in \mathbb{R}^{3N_v}$ of the mean face and the N_t orthogonal directions of maximal variation in face albedo $\boldsymbol{T} \in \mathbb{R}^{3N_v \times N_t}$. $\boldsymbol{S}$ and $\boldsymbol{T}$ are scaled such that the prior distribution over the shape and texture parameters is given by a normal distribution with unit covariance

$$p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\alpha} \mid \boldsymbol{0}, \boldsymbol{I})\mathcal{N}(\boldsymbol{\beta} \mid \boldsymbol{0}, \boldsymbol{I}) \propto \exp\{-\left\|\boldsymbol{\alpha}\right\|^2 - \left\|\boldsymbol{\beta}\right\|^2\} \quad , \tag{2.2}$$

under the assumptions that (1) the input data was already normally distributed and (2) shape and texture are independent.

1 Camera Model

These linear 3D models are extended to *nonlinear* 2D models of face images I by combining them with a deterministic camera and lighting function R into

$$I(\boldsymbol{\rho}, \boldsymbol{\iota}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = R(\boldsymbol{\rho}, \boldsymbol{\iota}, \mathcal{S}(\boldsymbol{\alpha}), \mathcal{T}(\boldsymbol{\beta})) \quad . \tag{2.3}$$

R projects the 3D face described by $(\mathcal{S}(\boldsymbol{\alpha}), \mathcal{T}(\boldsymbol{\beta}))$ according to the camera parameters $\boldsymbol{\rho}$ and illuminates it according to the lighting parameters $\boldsymbol{\iota}$.

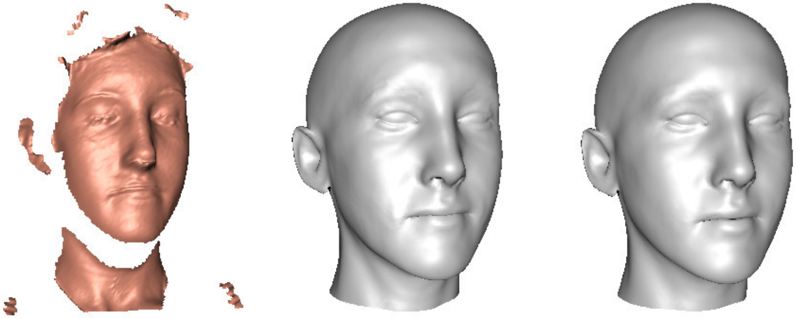
The problem of model fitting is then to find for a given image or video the parameters which generated the image. This is defined more formally in chapter 4. Now we will describe how to construct a morphable model from training examples.

2 Training Morphable Models

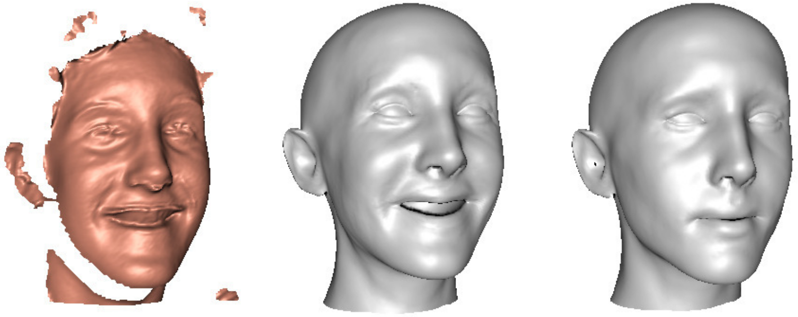
3D Morphable Models are learned from data. The training data consists of registered examples of the 3D shape and texture of faces. The faces are parametrized as triangulated meshes. In this context, *registered* means that every face is in the same parametrization, i.e. shares the same triangulation, and that semantically corresponding points such as the corners of the eye are at the same position in this parametrization, they have the same vertex number. In chapter 3 we describe how to obtain such data. Registered face scans have been shown to have the property that convex combinations of the example scans yield new valid faces, such that the registered examples span a *linear object class*. The description of the registration method used in this work occupies most of this chapter, but let us start by explaining the calculation of a 3D Morphable Model from registered examples.

From the training data the mean, offset vectors, and variances of the shape and texture model introduced in equation 2.1 have to be determined. Once the example shapes and textures are in a common parametrization this is done by applying Prinicipal Component Analysis (PCA) (Hotelling, 1933). PCA decomposes the covariance of the training data into a generative basis and the variances associated to the basis vectors under the assumption of normally distributed training data. For a very readable introduction please refer to Bishop (2007). A detailed description of the application of PCA to linear 3D Morphable Model building is given in Blanz and Vetter (1999).

For data with expressions, a model with separate coefficients for the identity and expression have been proposed in Blanz et al. (2003) and Amberg et al. (2008a). Such a separated model consists of an identity model, which is built from the registered neutral expression scans in the usual way, yielding a mean shape $\boldsymbol{s}$ and an identity basis $\boldsymbol{S}_I$ as before. The offset between the expression scans and the corresponding neutral scans is then used to determine an additional basis $\boldsymbol{S}_E$, such that the overall model has separate coefficients for identity and



a) Target b) Fit c) Normalized



a) Target b) Fit c) Normalized

Figure 2.1: A morphable model with distinct parameters for shape and expressions can be used to separate shape from expression in unseen data. The figure shows the seperation of 3D scans into a shape and an expression component. By performing recognition on the expression normalized data it is possible to achieve high recognition rates. The figure is reproduced from Amberg et al. (2008a). Here (a) is the scanned surface which is explained by the fitted model (b). Holes and scanning artifacts are removed by using a robust fitting. The pose and expression normalized faces (c) are used for face recognition.

expression

$$\begin{aligned}\mathcal{S}(\boldsymbol{\alpha}_I, \boldsymbol{\alpha}_E) &= \boldsymbol{s} + \boldsymbol{S}_I\boldsymbol{\alpha}_I + \boldsymbol{S}_E\boldsymbol{\alpha}_E \\ &= \boldsymbol{s} + \begin{bmatrix}\boldsymbol{S}_I & \boldsymbol{S}_E\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_I \\ \boldsymbol{\alpha}_E\end{bmatrix} \quad .\end{aligned} \tag{2.4}$$

Blanz et al. (2003) noted that the two bases are not orthogonal, and accordingly the same shape can be described by more than one parameter vector. In Amberg et al. (2008a) it was demonstrated that the resulting model can nonetheless be used in a recognition setting, where it is necessary to distinguish the contribution from identity and expression when doing expression invariant identity recognition. Amberg et al. (2008a) used a maximum a posteriori approach (instead of an intractable marginalization) to determine the most likely identity and expression coefficients given an example 3D scan. The maximum of the posterior is computable, even though the bases are not orthogonal, as the prior over the shape and expression coefficients results in a well defined posterior, for which the maximum can be found with standard nonlinear optimization. Separating the identity and expression coefficients has advantages not only for expression invariant recognition, but also for model fitting to videos as described in chapter 4 and video editing as described in chapter 8.

The next chapter explains the registration method used in this work to bring the example scans into a common parametrization.

— Chapter 3 —

Establishing Correspondence: Registration

Registration is the task of parametrizing one shape in a terms of another shape, such that semantically corresponding points are mapped onto each other. This reparametrization can also be seen as a deformation of the reference shape into the target shape.

Registration is usually achieved by simultaneously minimizing a measure of the irregularity of the deformation of the template and of the distance between the deformed template and the target. The measure of deformation irregularity encodes the prior assumptions about the expected deformation between the shapes while the difference measure is usually a relatively simple distance such as the average L^2 distance, but a other distance can also be employed to achieve robustness against missing data in either the template or the target.

In this chapter we make the connection between mesh-editing and registration, propose a new deformation measure, and compare it to the measures used in the mesh-editing community. We then demonstrate how to apply this measure to the problem of registering faces.

1 Introduction

To build a 3D Morphable Model one requires registered training examples. That is, the training examples need to be in a common parametrization, which is constructed such that semantically equal points, such as the corners of the eye, are at the same position in the parametrization. One can view such a parametrization as a *deformation* of the reference, or template, into the target shape. While the correspondence at salient features such as the corners of the mouth can be determined easily, it is more difficult to define the correspondences on in-between points. A registration algorithm therefore typically chooses a smooth deformation of the template which matches the surfaces and the feature points.

Smooth deformations are also of interest for mesh editing, which led us to investigate the use of the methods from that field for registration. In mesh editing one searches for a deformation of a mesh given a constraint on the position of some vertices. The deformation should be fair and intuitive, such that an artist can create new shapes by moving some vertices of an existing shape and have the surrounding vertices follow. It is quite difficult to define what a fair and intuitive deformation is in mesh editing. For some registration problems it is much easier to define what a good deformation is. When one shape is produced by another shape by physically deforming the first one, as is the case for a face with two different expressions, then the physical constraints such as the preservation of matter should be satisfied.

When registering different instances of the same class but not of the same object, e.g. the faces of two different persons, then no physically realisable deformation exists, so we need different constraints. We are in this second case, because we want to register a large database of faces from different individals. We therefore propose the use of a novel energy, which nicely solves the mesh editing problem, for registration. The energy we developed is useful for our task, because it fullfills the following two properties. (1) it is not volume or area preserving, but rather as smooth as possible and (2) it is fast to optimize such that it can be applied to high resolution meshes. We propose not only the energy but also an efficient optimization method, and show how it can be incorporated into the nonrigid ICP framework described in Amberg et al. (2007b).

The properties of the novel deformation method are demonstrated on mesh-editing tasks, where we compare it to a range of existing methods. Also registration results on a large dataset are shown. The registration method introduced here is especially well suited for datasets with a large percentage of missing data but unchanging topology, such as the face dataset.

2 A Deformation Energy, Useful for Mesh Editing

In the first part of this chapter we develop the deformation energy and optimization strategy with a view on mesh editing, because that makes it easy to demonstrate its properties. The results from the first part are then used in the second part to derive the registration method used in our system.

2.1 Method

We propose a novel deformation energy for meshes and an efficient method to determine the minimum deformation configuration of a mesh given constraints on vertex positions. The deformation energy has desirable properties for mesh editing and for use in a nonrigid registration algorithm. Already, a large number of deformation energies have been proposed. Physically based methods discretize the object into finite elements, (e.g. Wicke et al., 2007) and minimize nonlinear measures of shear, bending and scaling, such that the surface areas or volumes are preserved and the deformations are smooth. Other methods based on Poisson editing of feature vectors extracted from the meshes can be very efficient and generate pleasing deformations. They manipulate an extracted feature vector, e.g. the surface normals and solve for the mesh which best fits the manipulated vector. For an overview over these methods refer to the survey of Botsch and Sorkine (2008). As these methods typically fail to transform local details correctly they have been enhanced by decomposing the mesh into a high frequency and a low frequency part which are deformed independently and integrated afterwards (Botsch et al., 2006b). The disadvantage of this separation is, that the separation into high and low frequency components of the shape is ad hoc and fails when the scale of the high frequency components changes, or more than two levels of details are present. An example is the fine shape of the eye-brows whose orientation depends on the curvedness of the front which itself depends on the shape of the face.

Our energy and optimization method occupies its own place within the existing methods, as it does not aim to preserve volume or length, but rather finds a C^1 smooth deformation while allowing for local scaling. This is necessary for mesh editing, when the task is not to pose, i.e. deform in such a way as a real object would deform, an existing mesh, but rather to create new shapes by extruding, locally scaling or locally rotating the mesh. And it is necessary when registering different individuals of the same class, e.g. faces, for which no physically based deformation exists that transforms one individual into another.

For mesh editing and to a lesser extent for registration it is necessary to have an energy which can be optimized efficiently, in mesh editing one needs interactive responses. In a registration algorithm there is no need for interactive rates, but to process huge datasets we still need an efficient algorithm, because the deformation energy has to be minimized repeatedly. A special case of our deformation measure results in a quadratic cost which can be minimized non-iteratively. This results in very fast mesh-editing performance, while overcoming many of the problems of linear methods identified in Botsch and Sorkine (2008). The general nonlinear case results in even smoother and more intuitive deformations, while still being fast to evaluate.

The deformation energy which we consider penalizes the squared magnitude of second derivatives of the transformation function

$$T(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}^3 \tag{3.1}$$

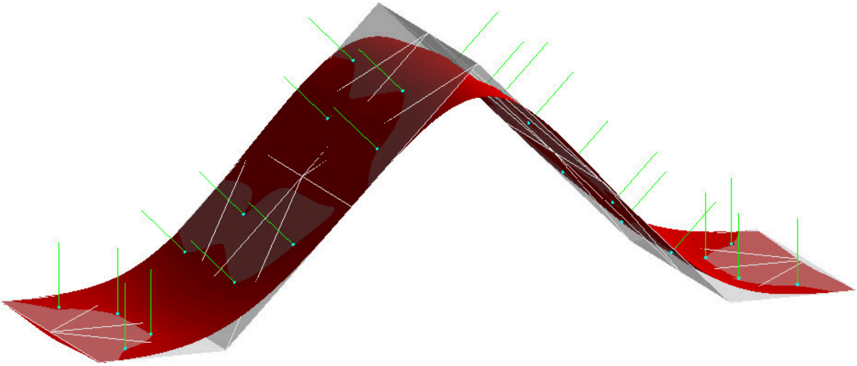


Figure 3.1: We assume that the given triangle mesh is a discrete approximation of an underlying surface which touches the mesh at the triangle barycenters and is tangential at these points.

deforming the template into a deformed target, over the surface $\mathcal{S}$

$$E_{\text{def}}(T) = \int_{\mathcal{S}} \|\nabla^2 T(\boldsymbol{x})\|^2 \,\mathrm{d}\boldsymbol{x} \tag{3.2}$$

but also incorporates the constraint that the deformation of the infinitesimal volume around the surface deforms the normals of the original surface such that they are still normal to the deformed surface. Minimization of the summed square of the second derivative yields a smooth, at least C^1 continuous surface. We observed, that adding the constraint that the space perpendicular to the surface should not shear or scale leads to a correct deformation of local details, but makes the resulting optimization nonlinear. We formalize this in the next paragraphs and propose an efficient solution to this nonlinear optimization problem. The resulting deformations are 'fair' and intuitive to handle.

We work with triangle meshes, which are interpreted as approximations of an underlying smooth surface, which touches the mesh at the barycenters of the triangles and is tangential to the mesh at these points. We do not give a constructive description of the underlying surface, it is only used conceptually, to define the bending energy. For such an underlying surface to exist, it is necessary that no edge of the mesh is shared by more than two triangles, such that the underlying surface is manifold, it resembles a two dimensional plane at each point, apart from its boundaries. This makes it possible to approximately evaluate the integral from Equation 3.2 even though the only deformations that a triangle mesh can undergo are piecewise linear. See figure 3.1 for a visualization of this concept.

We use the following notation. A triangle $\triangleright_{ijk}$ with vertices $(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k)$ has the edges $\boldsymbol{e}_{ij} = \boldsymbol{v}_j - \boldsymbol{v}_i$ and the normal $\boldsymbol{n}_{ijk} = (\boldsymbol{e}_{ij} \times \boldsymbol{e}_{ik})/\|\boldsymbol{e}_{ij} \times \boldsymbol{e}_{ik}\|$. We will denote the triangles, vertices, edges and normals in the rest state of the mesh by $\tilde{\triangleright}$, $\tilde{\boldsymbol{v}}$, $\tilde{\boldsymbol{e}}$ and $\tilde{\boldsymbol{n}}$. The deformation between a triangle $\triangleright_{ijk}$ in its rest state and its deformed state is given by a linear transformation

$$T(\boldsymbol{x}) = \boldsymbol{A}_{ijk}\boldsymbol{x} + \boldsymbol{t}_{ijk}, \tag{3.3}$$

with a 3×3 affine part $\boldsymbol{A}_{ijk}$ and a translation part $\boldsymbol{t}_{ijk}$. This deformation is constant over the triangle, but the deformation which we are reasoning over will be continously changing over the hypothetical underlying surface. The deformation for a triangle is not uniquely determined by the configurations of the triangle in its rest and deformed state, as the direction normal to the triangle can be deformed arbitrarily. We enforce deformations which keep the normal orthogonal to the triangle and do not scale it. The 3×3 matrix $\boldsymbol{A}_{ijk}(\boldsymbol{v})$ describing the affine part of the deformation is determined by the following system of linear equations, from the deformed vertices $\boldsymbol{v} = \{\boldsymbol{v}_1, \ldots, \}$:

$$\boldsymbol{A}_{ijk}(\boldsymbol{v}) \begin{bmatrix} \tilde{\boldsymbol{e}}_{ij} & \tilde{\boldsymbol{e}}_{ik} & \tilde{\boldsymbol{n}}_{ijk} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_{ij}(\boldsymbol{v}) & \boldsymbol{e}_{ik}(\boldsymbol{v}) & \boldsymbol{n}_{ijk}(\boldsymbol{v}) \end{bmatrix} \quad , \tag{3.4}$$

in terms of the edges $\tilde{\boldsymbol{e}}$ and normals $\tilde{\boldsymbol{n}}$ of the rest state and the edges $\boldsymbol{e}$ and normals $\boldsymbol{n}$ computed from the deformed vertices $\boldsymbol{v}$. As the rest state of the mesh is constant it follows that the matrix $\boldsymbol{A}_{ijk}(\boldsymbol{v})$ is a linear function of the edges and normals of the deformed mesh.

$$\boldsymbol{A}_{ijk}(\boldsymbol{v}) = \begin{bmatrix} \boldsymbol{e}_{ij}(\boldsymbol{v}) & \boldsymbol{e}_{ik}(\boldsymbol{v}) & \boldsymbol{n}_{ijk}(\boldsymbol{v}) \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{e}}_{ij} & \tilde{\boldsymbol{e}}_{ik} & \tilde{\boldsymbol{n}}_{ijk} \end{bmatrix}^{-1}}_{\text{Constant}} \tag{3.5}$$

The edges are again a linear function of the vertex positions $\boldsymbol{v}$, but the normals are a nonlinear function of the deformed vertex positions.

Now regard two adjacent triangles $\triangleright_{ijk}$ and $\triangleright_{ijl}$. We assume that the triangles are tangential to the true surface $\mathcal{S}$ and touch the surface at the triangle barycenters $\boldsymbol{x}_{ijk}$ and $\boldsymbol{x}_{ijl}$. Note that these points are defined in the rest state surface $\mathcal{S}$, so they are constant throughout the algorithm.

We know the first derivative of the deformation of the underlying surface $\mathcal{S}$ at the triangle barycenters, it is just the affine part of the deformation acting on the triangles. We do not determine the second derivative in Equation 3.2 exactly, but use a finite differences approximation of the directional derivative between $\boldsymbol{x}_{ijk}$ and $\boldsymbol{x}_{ijl}$ to approximate Equation 3.2 by

$$E_{\text{def}}(T) \approx E_s(\boldsymbol{v}) = \sum_{(i,j,k,l) \in \mathcal{N}} a_{ijkl} \left\| \frac{A_{ijk}(\boldsymbol{v}) - A_{ijl}(\boldsymbol{v})}{\|\boldsymbol{x}_{ijk} - \boldsymbol{x}_{ijl}\|} \right\|_F^2 \tag{3.6}$$

where $\mathcal{N}$ are neighbouring triangles and a_{ijkl} is the area of the discretization element as shown in figure 3.2. Here $\|\cdot\|_F^2$ denotes the squared Frobenius norm. This

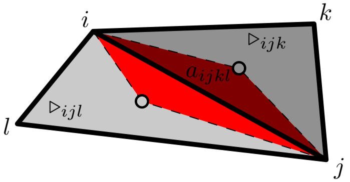


Figure 3.2: The weighting area used in the discretization of the derivative. Shown are two triangles with the triangle barycenters used for the finite differences discretization and in red the area by which this derivative is weighted. Compare Equation 3.6 for details.

is different from the continuous expression in 3.2, because the second derivative is taken only in the tangent space of the surface, not in the embedding space, but because we will be enforcing that the deformations do not scale or shear the space normal to the surface this is sufficient to generate well behaved deformations. As noted above $A_{ijk}(\boldsymbol{v})$ is a linear function in the vertices and normals of the deformed mesh, which enables us to rewrite Equation 3.6 as

$$E_s(\boldsymbol{v}) = \left\| \boldsymbol{S} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right\|^2 \quad , \tag{3.7}$$

with a suitably defined matrix $\boldsymbol{S}$.

Given the deformation energy we can now proceed to use it for mesh deformation. To this end, we introduce soft correspondences, also often called forces in the mesh editing community, that determine vertex positions. It is easily possible to generalize this to hard constraints on the positions of some vertices, which we do in section 2.2. We express the constraints between surface points and landmarks by a matrix $\boldsymbol{C}$, which contains in every row the barycentric coordinates of a surface point at the columns corresponding to the enclosing triangles vertices. The corresponding landmarks are stacked in a $N_l \times 3$ matrix $\boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_1 & \dots & \boldsymbol{c}_{N_l} \end{bmatrix}^T$. We choose the energy as a linear combination of the stiffness energy E_s and the correspondence energy E_c.

$$\begin{aligned} E(\boldsymbol{v}) &= E_s(\boldsymbol{v}) + \lambda E_c(\boldsymbol{v}) \\ E_c(\boldsymbol{v}) &= \|\boldsymbol{C}\boldsymbol{v} - \boldsymbol{c}\|^2 \\ E_s(\boldsymbol{v}) &= \left\| \boldsymbol{S} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right\|^2 \quad , \end{aligned} \tag{3.8}$$

where $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 & \dots & \boldsymbol{v}_{N_v} \end{bmatrix}^T$ is a $N_v \times 3$ matrix of vertex positions, and $\boldsymbol{n}(\boldsymbol{v}) = \begin{bmatrix} \boldsymbol{n}_1 & \dots & \boldsymbol{n}_{N_t} \end{bmatrix}^T$ are the normals of the N_t triangles, as a nonlinear function of the vertex positions.

Efficient Optimization

We derive our optimization method from the Gauss-Newton descent method, which calculates the update step as

$$\Delta \boldsymbol{v} = -(\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{J}^T \boldsymbol{r} \tag{3.9}$$

where $\boldsymbol{J}$ is the Jacobian of the sum of squares cost function, and $\boldsymbol{r}$ is the residual vector. Equation 3.8 is expensive to optimize, as it is a nonlinear function for which the Jacobian and gradient need to be calculated in each iteration of the optimizer. We propose a very efficient optimization scheme by decoupling the stiffness term and the normal estimation.

We approximate 3.8 with another function in terms of the vertices *and a new set of variables* $\bar{\boldsymbol{n}}$ *for the 'normals'* and introduce a new normal coupling cost $\bar{E}_n$ which makes $\bar{\boldsymbol{n}}$ and the true normals $\boldsymbol{n}(\boldsymbol{v})$ as similar as possible

$$E(\boldsymbol{v}) \approx \bar{E}(\boldsymbol{v}, \bar{\boldsymbol{n}}) = \bar{E}_s(\boldsymbol{v}, \bar{\boldsymbol{n}}) + \lambda^2 E_c(\boldsymbol{v}) + \kappa^2 \bar{E}_n(\boldsymbol{v}, \bar{\boldsymbol{n}}) \tag{3.10}$$

$$\begin{aligned} \bar{E}_s(\boldsymbol{v}, \bar{\boldsymbol{n}}) &= \left\| \boldsymbol{S} \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{n}} \end{bmatrix} \right\|^2 \\ \bar{E}_c(\boldsymbol{v}, \bar{\boldsymbol{n}}) &= \left\| \boldsymbol{C}\boldsymbol{v} - \boldsymbol{c} \right\|^2 \\ \bar{E}_n(\boldsymbol{v}, \bar{\boldsymbol{n}}) &= \left\| \bar{\boldsymbol{n}} - \boldsymbol{n}(\boldsymbol{v}) \right\|^2 \end{aligned} \tag{3.11}$$

This introduces some slack on the constraint that the normal should be kept. Next we approximate the normal coupling cost iteratively by ignoring the dependency of the normal direction on the changing vertex position. We substitute $\boldsymbol{n}(\boldsymbol{v})$ by the fixed normals from the previous shape estimate and approximate

$$\bar{E}_n(\boldsymbol{v}, \bar{\boldsymbol{n}}) \approx \left\| \bar{\boldsymbol{n}} - \boldsymbol{n}(\boldsymbol{v}^{t-1}) \right\|^2 \tag{3.12}$$

where $\boldsymbol{v}^{t-1}$ denotes the current estimate of the vertex positions, and $\boldsymbol{v}$ is the newly calculated position.

These approximations result in a quadratic problem to be solved in each step, which was constructed such that only the right hand side of the problem changes in each iteration. We could therefore describe the method as an alternation scheme, but I find it easier to follow through with the Gauss-Newton terminology, because this makes the approximations more explicit.

The Jacobian matrices of the three partial costs are constant

$$\boldsymbol{J}_s = \boldsymbol{S} \qquad \boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{J}_n = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.13}$$

and the complete Jacobian includes the weights λ and κ,

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_s \\ \lambda \boldsymbol{J}_c \\ \kappa \boldsymbol{J}_n \end{bmatrix} \quad . \tag{3.14}$$

As the Jacobian is constant, so is the approximate Hessian $\boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J}$. The update step is then

$$\begin{aligned}
\Delta \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{n}} \end{bmatrix} &= -(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T \left(\boldsymbol{J} \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{n}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \lambda \boldsymbol{c} \\ \kappa \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right) \qquad (3.15) \\
&= -\begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{n}} \end{bmatrix} + (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T \begin{bmatrix} \boldsymbol{0} \\ \lambda \boldsymbol{c} \\ \kappa \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \\
&= -\begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{n}} \end{bmatrix} + \lambda \boldsymbol{H}^{-1}\boldsymbol{J}_c^T \boldsymbol{c} + \kappa \boldsymbol{H}^{-1}\boldsymbol{J}_n^T \boldsymbol{n}(\boldsymbol{v})
\end{aligned}$$

The second term is constant for given correspondences, so we substitute it with $\boldsymbol{k} = \lambda \boldsymbol{H}^{-1}\boldsymbol{J}_c^T \boldsymbol{c}$. We are not interested in the update to $\bar{\boldsymbol{n}}$, as the correct normals are estimated from the vertex positions so we get rid of the corresponding rows and simplify to

$$\Delta \boldsymbol{v} = \boldsymbol{k} - \boldsymbol{v} + \kappa \boldsymbol{H}^{-1}\boldsymbol{J}_n^T \boldsymbol{n}(\boldsymbol{v}) \qquad (3.16)$$

In this way we have arrived at an update rule which is linear in the vertex positions and the current normals. Even though $\boldsymbol{H}^{-1}\boldsymbol{J}_n^T$ is constant it is dense, therefore we do not precalculate it but instead compute a sparse Cholesky decomposition of $\boldsymbol{H}$.

The partial Jacobian matrices $\boldsymbol{J}_s$ and $\boldsymbol{J}_n$ can be precomputed when the mesh is loaded, as they depend only on the rest state of the mesh. The Hessian on the other hand has to be newly decomposed whenever the choice of points to manipulate changes. When only the position of the landmarks changes, as is most often the case during mesh editing, then it suffices to recompute $\boldsymbol{k}$, which is very cheap. Also, to speed up the Cholesky decomposition necessary when the landmark vertices are chosen, we compute the reordering for the Hessian matrix using the structure of $\boldsymbol{J}_s^T\boldsymbol{J}_s + \boldsymbol{I}$ during load time.

To further speed up the calculation we implemented a multigrid approach Wesseling (1992) with a hierarchy of reduced meshes, which is also precomputed on mesh loading. We perform two coarsening and refining iterations through the multigrid hierarchy, the first with only 10 iterations per hierarchy level, the second with up to 100 iterations per level. We terminate the optimization when the position of the mesh vertices is stationary, as measured by a threshold on the difference of vertex positions in adjacent iterations.

2.2 Absolute Constraints

In the previous section we developed the optimization with soft constraints, i.e. minimizing $\lambda^2 \left\| \boldsymbol{C}\boldsymbol{v} - \boldsymbol{c} \right\|$. For mesh editing it is often desirable to have hard constraints. That is, one fixes a subset of the vertices at a certain position and calculates the position of the remaining vertices.

This can be achieved by partioning the matrices from equation 3.8 into the columns pertaining to the moving vertices, and those relating to the fixed vertices. The part corresponding to the fixed vertices is constant and can be removed from the minimization.

Denote the indices of the moving vertices by $\Updownarrow$ and the indices of the fixed vertices by $\{$, also denote the sub-matrices one gets by selecting the appropriate columns of the Jacobian and Hessian matrices with the subscripts $\{$ and $\Updownarrow$, then we get

$$\bar{E} = \left\| \boldsymbol{J}_{\Updownarrow} \begin{bmatrix} \boldsymbol{v}_{\Updownarrow} \\ \bar{\boldsymbol{n}}_{\Updownarrow} \end{bmatrix} + \boldsymbol{J}_{\{} \begin{bmatrix} \boldsymbol{v}_{\{} \\ \boldsymbol{n}(\boldsymbol{v}_{\{}) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \lambda \boldsymbol{c} \\ \kappa \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right\|^2 \tag{3.17}$$

$$\begin{aligned} \Delta \begin{bmatrix} \boldsymbol{v}_{\Updownarrow} \\ \bar{\boldsymbol{n}}_{\Updownarrow} \end{bmatrix} &= -\boldsymbol{H}_{\Updownarrow}^{-1} \boldsymbol{J}_{\Updownarrow}^{T} \left(\boldsymbol{J}_{\Updownarrow} \begin{bmatrix} \boldsymbol{v}_{\Updownarrow} \\ \bar{\boldsymbol{n}}_{\Updownarrow} \end{bmatrix} + \boldsymbol{J}_{\{} \begin{bmatrix} \boldsymbol{v}_{\{} \\ \boldsymbol{n}(\boldsymbol{v}_{\{}) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \lambda \boldsymbol{c} \\ \kappa \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right) && (3.18) \\ &= -\begin{bmatrix} \boldsymbol{v}_{\Updownarrow} \\ \bar{\boldsymbol{n}}_{\Updownarrow} \end{bmatrix} - \boldsymbol{H}_{\Updownarrow}^{-1} \left(\boldsymbol{J}_{\Updownarrow}^{T} \boldsymbol{J}_{\{} \begin{bmatrix} \boldsymbol{v}_{\{} \\ \boldsymbol{n}(\boldsymbol{v}_{\{}) \end{bmatrix} - \boldsymbol{J}_{\Updownarrow}^{T} \begin{bmatrix} \mathbf{0} \\ \lambda \boldsymbol{c} \\ \kappa \boldsymbol{n}(\boldsymbol{v}) \end{bmatrix} \right) \\ &= -\begin{bmatrix} \boldsymbol{v}_{\Updownarrow} \\ \bar{\boldsymbol{n}}_{\Updownarrow} \end{bmatrix} + \boldsymbol{k}_{\Updownarrow} + \kappa \boldsymbol{H}_{\Updownarrow}^{-1} \boldsymbol{J}_{n\Updownarrow}^{\;T} \boldsymbol{n}(\boldsymbol{v}) \\ \boldsymbol{k}_{\Updownarrow} &= \boldsymbol{H}_{\Updownarrow}^{-1} \left(\lambda \boldsymbol{J}_{c\Updownarrow}^{\;T} \boldsymbol{c} - \boldsymbol{J}_{\Updownarrow}^{T} \boldsymbol{J}_{\{} \begin{bmatrix} \boldsymbol{v}_{\{} \\ \boldsymbol{n}(\boldsymbol{v}_{\{}) \end{bmatrix} \right) && (3.19) \end{aligned}$$

One has to be careful to also select all columns with fixed normals, where a triangle has a fixed normal when all its vertices are fixed. Absolute constraints are useful in some tasks where part of the mesh is required to stay fixed, and they have the advantage that the problem size is reduced to the number of vertices which are actually moving. In a typical mesh editing task this is only a fraction of all vertices. This makes it possible to perform real time editing on even larger meshes.

2.3 Normal Slackness

By changing the slackness parameter κ one changes how strongly the deformation is normal preserving. An interesting special case occurs for $\kappa = 0$. The problem in this case is still fully constrained for non-planar surfaces, but the energy reduces to a quadratic function, which can be solved in closed form. The resulting deformation is still smooth, though the normal-preserving deformations are more intuitive. We demonstrate the effect of varying κ in figure 3.3, where

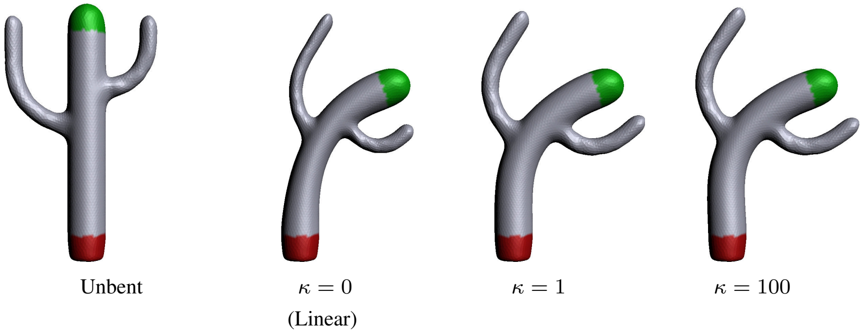


Figure 3.3: Without the normal constraint the cost reduces to a quadratic form, which can be solved exactly. Without the normal constraint the problem can become unconstrained, when the deformation consists of only a translation. We solve this by fixing not only the vertices but also the normals of triangles which have all vertices fixed. While the resulting surface deformation is smooth, it is not as intuitive as the result with increasing κ. Observe the thinning of the stem of the cactus and the deformation of the protruding arms, which are more natural with the normal constraint. The effect of increasing κ saturates, which implies that the normal constraint is solved nearly exactly for $\kappa = 100$.

the same deformation is applied to a cactus for varying values of κ. A larger κ leads to a fair and natural deformation of the cactus, and also the protruding arms of the cactus behave in the expected way. This result is shown in more detail in the movie in the additional material.

2.4 Comparison to Other Methods

A number of different deformation energies and minimization methods have been proposed. This section compares the advantages and disadvantages of the most prominent of these methods. We use the “difficult examples” from the survey paper Botsch and Sorkine (2008) for comparison. This includes results for a number of so called linear methods, like our method for $\kappa = 0$, which solve a quadratic cost and the nonlinear method *PriMo* (Botsch et al., 2006a). The results from Botsch and Sorkine (2008) are compared in figure 3.4 and figure 3.5 to our method, which we call *Normsurf*, and the linearized version of *Normsurf* with $\kappa = 0$. Only the non-linear methods can solve all the problematic cases used in this survey, and result in intuitive deformations. We should point out that *PriMo* and *Normsurf* address two different problems. While *PriMo* tries to keep the area of the surface constant and simultaneously minimizing bending and stretching, we are not trying to preserve the volume or surface area. Therefore, our method

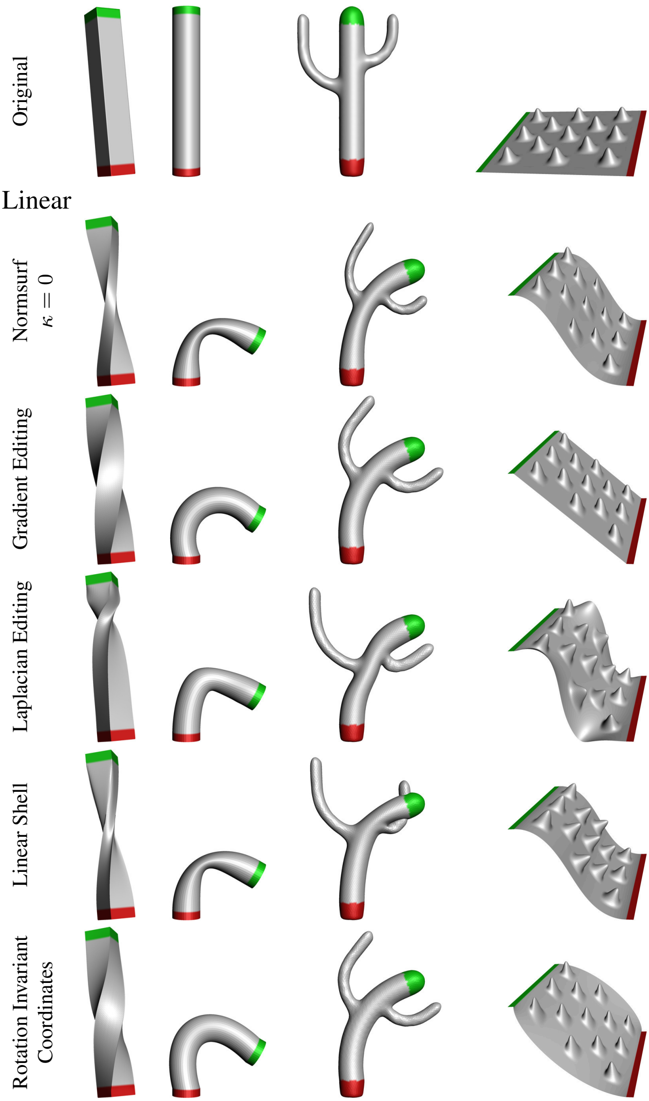


Figure 3.4: A comparison of the linear version of our method (Normsurf, $\kappa = 0$) with four linear methods demonstrates the use cases for our energy and optimization method. Our energy results in intuitive deformations for all four difficult cases from the survey of Botsch and Sorkine (2008), where each linear method fails on at least one of these cases. The next figure shows that the nonlinear version gives even more intuitive results which are still different from that of the nonlinear method proposed in the survey.

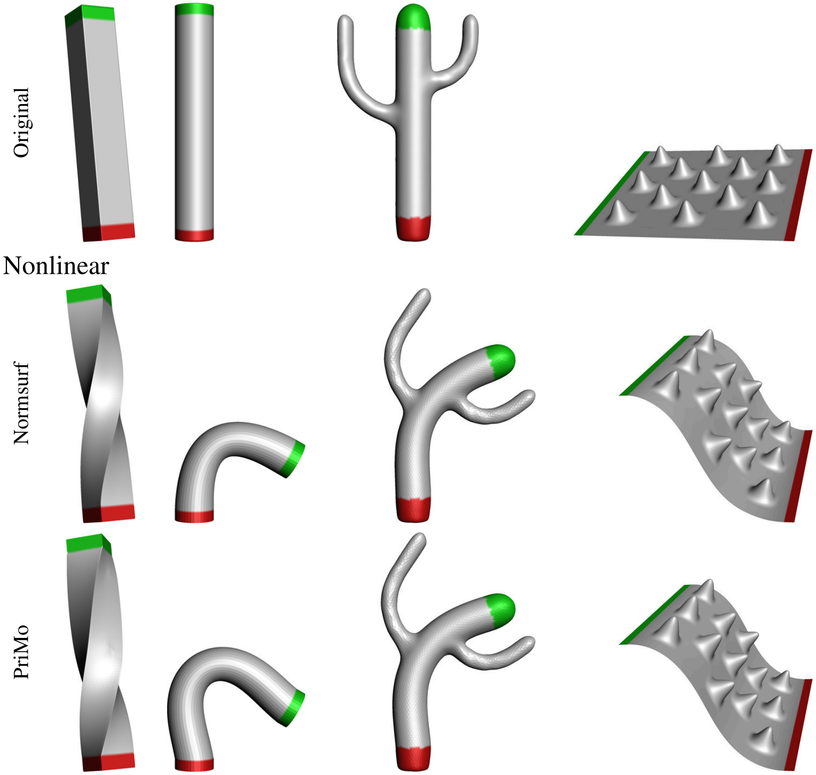
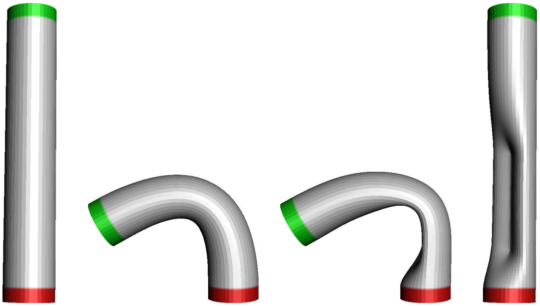


Figure 3.5: A comparison with the nonlinear method *PriMo* suggested in Botsch and Sorkine (2008). Our method addresses a different problem than the *PriMo* method, because area preservation is not the aim of this deformation. This makes the resulting deformation smoother and our energy very suited for mesh-editing and registration, but not for mesh-posing or inverse kinematics. Observe the first order discontinuity in the primo results between fixed and moving vertices.



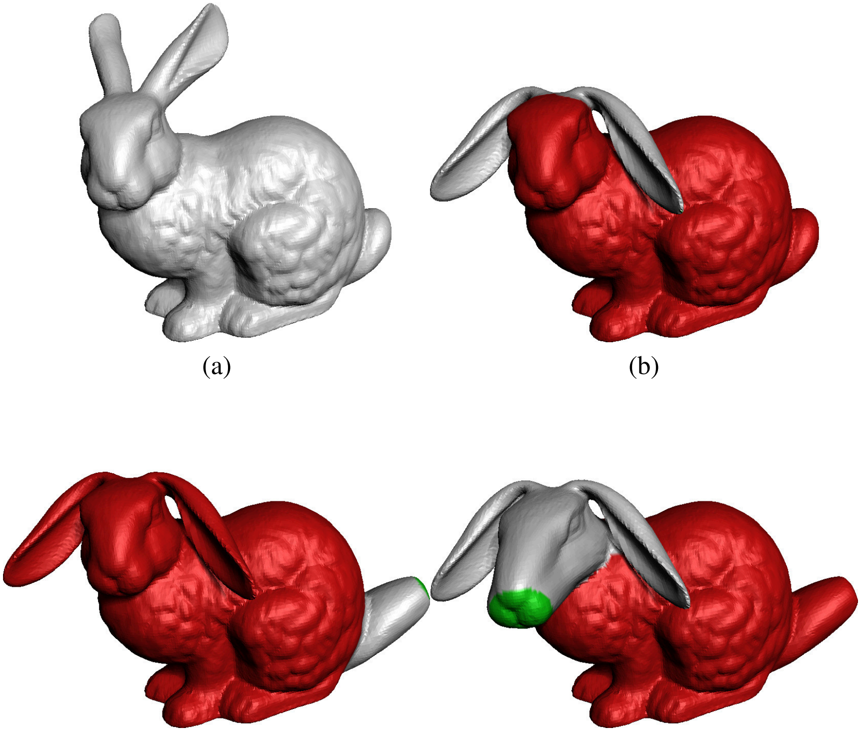
(a) (a.1) (b.1) (b)

Figure 3.6: The bending depends on the shape of the mesh. A profiled shape is harder to bend than an non-profiled shape, compare also the bunnies' ears in figure 3.7. This behaviour is intuitive and therefore useful for mesh editing. To the left is a cylinder (a) which is bent by moving the vertices marked in red and green into the shape (a.1). To the right is the same cylinder which was on its lower half deformed to have a U-shaped profile. When bending as before, the profiled part stays more rigid, resulting in the shape (b.1).

can reach smoother deformations, which is useful for registration and for mesh editing, but not for mesh posing or inverse kinematics. One can also observe on this dataset that the deformation which *PriMo* generates is not smooth between fixed and moving vertices, as opposed to that generated by our method.

2.5 Mesh Editing Results

Some mesh editing results are shown in figures 3.7, 3.6, and 3.8. Here we marked the fixed areas by red and green, where green areas are moved from their original positions. The gray vertices are unconstrained. Figure 3.7 shows a sequence of deformations applied to the scan of the stanford bunny (Turk and Levoy, 1994). Already in this figure one can observe, that a profiled shape is stiffer than an unprofiled shape, this is shown more isolated in figure 3.6. Figure 3.8 shows the results of editing the more complicated and higher resolution armadillo scan from Krishnamurthy and Levoy (1996).



(a) (b)

(c) (d)

Figure 3.7: Mesh editing with the proposed energy and optimization leads to intuitive behaviour. Red and green areas are the constrained regions, while the position of the vertices in the gray area is calculated. (a) shows the rest state of the mesh, in (b) we fixed the position of the body and of two vertices at the tips of the ears of the bunny, bending the ear. Note how the V-shaped ridge of the bunny ears leads to high stiffness, as evidenced by the bend at the base of the ears. This behaviour is consistent with the behaviour of a thin stretching shell, like stiff rubber. (c) extrudes the bunny's tail by moving a larger fixed region, and (d) makes the nose of the bunny longer. Notice the continuous deformation at the boundary between fixed and moving vertices.

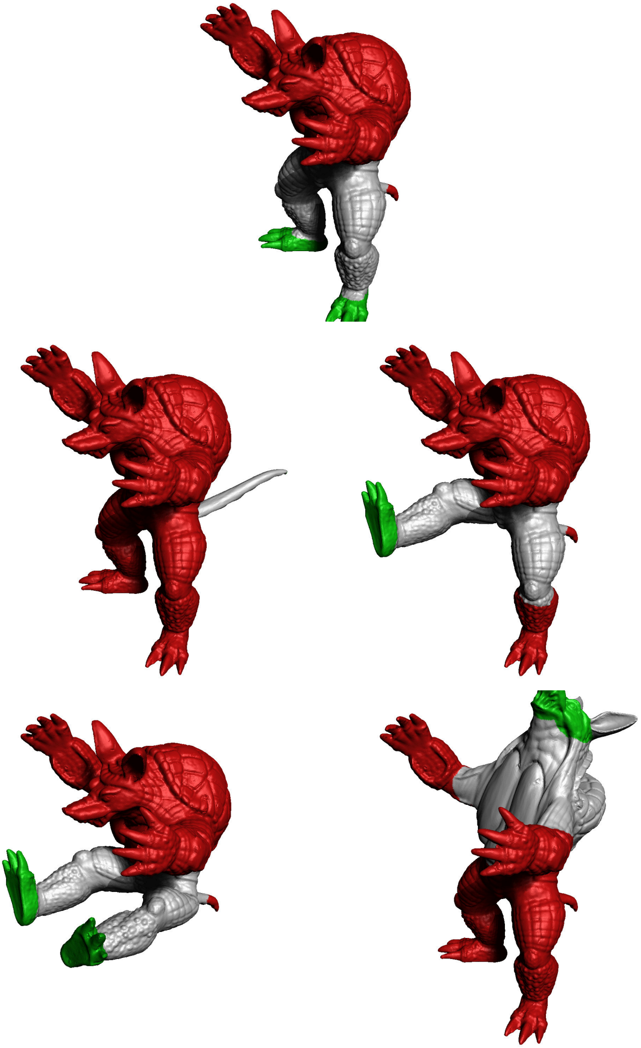


Figure 3.8: Using our deformation energy to pose a more complicated high resolution mesh. The proposed measure is shell based, and does not try to preserve volumes. The resulting deformations are intuitive when regarding the object as made from a thin shell, but not if the object is regarded as solid. Observe how the details are correctly transformed even under strong deformations. The deformations are adequate for mesh editing but for mesh posing a volume-preserving measure is better suited.

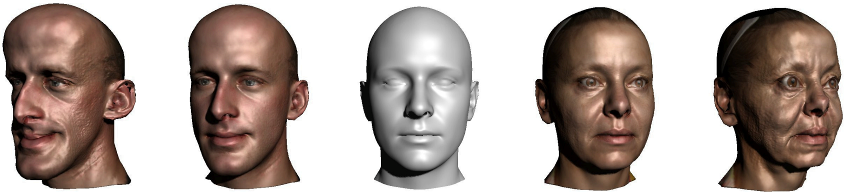
3 Registration

In this section we describe how our novel mesh deformation cost can be used to solve the registration problem between instances of 3D faces. Registration is the process of parametrizing one surface in terms of another surface. The goal is to find a mapping from the source to the target, which maps semantically corresponding points onto each other. When registering for example two faces, then the vertex on the tip of the nose of the source face should be mapped to the tip of the nose of the target face. While correspondences for salient points like the tip of the nose or the edges of the mouth could be determined by locally detecting features in both images, it is more difficult to establish a global dense correspondence between every point in the source mesh and the target surface. This can be accomplished by *regularising* the mapping between source and target, such that the overall deformation is in some sense smooth.

Registration is essential wherever statistical models are constructed (Blanz and Vetter, 1999; Allen et al., 2003) to serve as a prior in recognition or reconstruction tasks, or where comparisons between surfaces are needed.

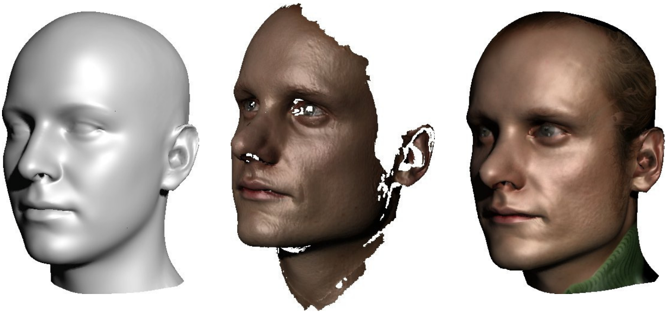
The source surface $\mathcal{S}$ and the target surface $\mathcal{T}$ are two dimensional surfaces embedded in three dimensional space. Starting from a parametrization of $\mathcal{S}$ we are searching for a parametrization of $\mathcal{T}$ such that semantically corresponding points have the same parametrization. We describe the parametrization with a transformation $T : \mathcal{S} \to \mathcal{T}$, and search for the T which minimizes the deformation energy $E_{\text{def}}(T)$ associated with the transformation (and maybe incorporates additional constraints such as landmark constraints between salient points). This can be formulated as an optimization problem over transformations $T : \mathcal{S} \to \mathbb{R}^3$ with the constraint that $T(\mathcal{S}) = \mathcal{T}$.

While using a general optimizer is possible, as shown by Fitzgibbon (2001), we chose to use a nonrigid ICP method to minimize the bending energy. To derive the ICP method we first relax the constraint that $T(\mathcal{S}) = \mathcal{T}$ into a distance energy between the deformed source $T(\mathcal{S})$ and the target $\mathcal{T}$, and solve a sequence of optimization problems with a more and more strongly weighted distance energy. This successively pulls the deformed surface $T(\mathcal{S})$ onto the target $\mathcal{T}$, which makes it possible for the optimization to find a better minimum than that which can be found with direct optimization of the unrelaxed problem. Also, using ICP allows the simple incorporation of a distance measure which changes with the currently deformed surface, e.g. the distance between similar regions, instead of the distance between the two surfaces. Additionally it is straightforward to incorporate a robust distance measure by using an iterative re-weighting scheme and by discarding detected outliers in each iteration. A robust distance measure results in a method which fills in the missing regions by relying only on the deformation energy in the missing parts, and adds robustness against spurious measurements.



Caricature Registered Mean Registered Caricature

Figure 3.9: Caricatures help to judge the registration quality by exaggerating flaws and artifacts. The left- and right-most images images are caricatures created by moving the registered and hole filled scans to twice their distance from the mean.



Template Scan Registered

Figure 3.10: The head template used, a typical (cleaned) mesh acquired by our scanning process, and the registration result.

For convex deformation measures, it is possible to find the l_2 optimal deformation between the source and a fixed set of landmarks, because the l_2 distance is also convex. This is the case for classical rigid or affine ICP as introduced by Besl and McKay (1992), and for the deformation measure proposed for nonrigid ICP in Amberg et al. (2007b).

A typical input scan acquired with a coded light scanner, the source template we use and the registration result are shown in figure 3.10. To evaluate the quality of the registration we also show a caricature of the result, by increasing the difference between the average face and the registered examples.

3.1 Prior Work

As registration is a basic tool for many computer vision and medical imaging applications there exist a large literature on registration. We focus here on registration of surfaces embedded in 3D space, and especially on nonrigid ICP. An overview of more related methods can be found in Amberg et al. (2007b).

The ICP method for rigid deformations was introduced by Besl and McKay (1992) and subsequently extended to nonrigid deformations by Feldmar and Ayache (1996); Allen et al. (2003); Amberg et al. (2007b). In Amberg et al. (2007b) it was shown that for the deformation measure in Allen et al. (2003) and a related measure based on the first derivative of the deformations along the surface a closed form solution for the update step can be derived. The deformation was parametrized in Allen et al. (2003) and Amberg et al. (2007b) by attaching a deformation, either an affine deformation or a translation, to each vertex of the source mesh, and minimizing the Frobenius norm of the difference between transformations attached to neighbouring vertices. As translations and affine transformations are linear operations, this measure is convex. In Amberg et al. (2007b) it was also shown that the resulting system of linear equations is fully constrained, when a sufficient number of correspondences are used, even though the position of each vertex is encoded by 12 parameters. Measuring the difference between affine deformations has the advantage that a global affine deformation does not introduce a cost and locally affine deformations are cheap, while a regularization based on translations does not allow the recovery of global affine deformation or locally bending structures. The disadvantage of the proposed measure is that it is dependent on the position of the origin and the units of the coordinate system, as the Frobenius norm of the difference between two affine transformations does weight the translational, rotational and scaling contributions differently. We overcome this problem by exchanging the cost from Amberg et al. (2007b) with the deformation energy proposed in this thesis. This also solves the problem that the previously used energy was based on a first derivative, resulting in a deformation which is not smooth when the correspondences are sparse. On the other hand we have already demonstrated that our new energy results in smooth deformations.

3.2 Additional Constraints

In practice, the minimum deformation between two faces does not always correspond to the actual semantic correspondences. We address this by adding further landmarks to guide the registration. The landmarks are included as additional correspondences in the distance term. Our data comes from a structured light scanner which also acquires three calibrated photographs of the target shape, which are used to texture the shape and to mark curves in these photographs which are extruded into 3D space to generate “surface landmarks” for further correspondences. We mark the outline of the lips and the eyebrows, as these are only visible in the texture, not in the shape, the outline of the ears, as those are cut off in the scanning process, and the outline of the eye as this region is

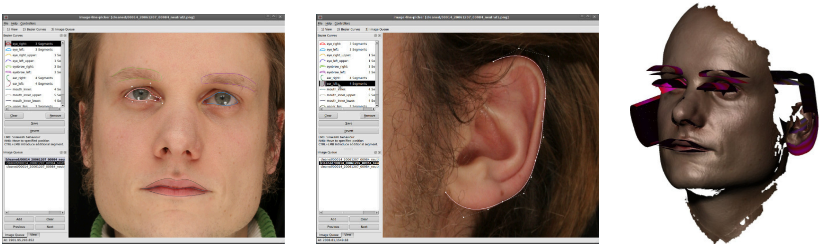

Boundaries in Camera 1 Boundaries in Camera 2 Extruded Boundaries

Figure 3.11: Additional Constraints are given by landmarks and 2D contours. The 2D contours are extruded into 3D space, and the corresponding face vertices are pulled towards the closest points on the extruded contours. This allows 2D landmarks also at positions which do not have a corresponding 3D measurement.

also not measured accurately due to the reflection properties of the eyes and the eyelashes. For the “surface landmarks” we minimize the distance between the landmarked vertices on the template and the extruded curves. This allows us to define landmarks in the photographs even for regions where no data is available in the 3D scan. We do not use the texture directly to register face scans, as the texture differs significanty between subjects and is difficult to match reliably. We opted in this case for some manual work, because the additional manually set constraints significantly improve the resulting model. On the other hand texture is reliable when registering expressions of the same face against the registered neutral scan of the same person. In this case we do incorporate texture into the registration as described in section 3.5.

To make the 3D scans align at the neck, we add an additional plane at a given distance below the chin, and oriented according to the orientation of the head, onto which the vertices at the boundary of the neck are drawn. See figure 3.12 for details.

3.3 Robustness

A larger capture range is achieved by searching not indiscriminately for the closest point on the target, but instead taking into account only those points which have a normal similar to that of the current guess. This simple feature already removes wrong correspondences, which might be closer but are not locally similar. More advanced features such as curvature would also be of interest. Also, the correspondences are weighted by their inverse distance, such that the optimization becomes more robust against outliers. Additionally, to detect missing data we discard vertices whose closest point lies on a border of the target scan.

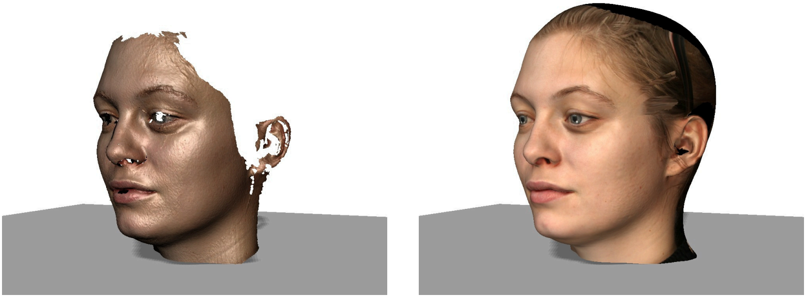

Scan Registered

Figure 3.12: The mesh boundary at the neck is pulled onto a plane which was placed at a constant position relative to the position of some vertices of the face. This requires a second registration run, as the first run defines the position of the neck surface.

3.4 Bootstrapping: Including a Shape Prior

To further improve registration results we incorporate a shape prior into the registration. We have access to the surfaces extracted from 30 volumetric MR-Images of heads, which we registered with the method described above. These MRI scans have a lower resolution, but they include the back of the head, which is missing in the coded light scans. From the MRI surfaces a 3D shape model was built, which includes statistically correct deformations of the back of the head and some information about possible face shapes. We use this shape model as the template during registration of the 3D surface scans by doing a coordinate descent on the template parameters and the deformation parameters. The backs of the heads filled in by the registration when using a shape prior are much more realistic than those resulting from registration with a fixed head model. While the bootstrapping could also be iterated by using the model from all registrations in a second registration round, this proved impractical because the process is divergent as the structured light scans do not constrain the back of the head. To fit the morphable model efficiently we use the method proposed in Amberg et al. (2008a).

3.5 Expression Deformations

To get an accurate model of the deformations of a face showing an expression we marked faces with a random coloured pattern and acquired the neutral and expressed face gestures. Additionally a neutral face without markup was recorded. We then applied the optical flow method of Liu (2009) between the extracted texture of the neutral marked up face and the deformed faces, and used the resulting

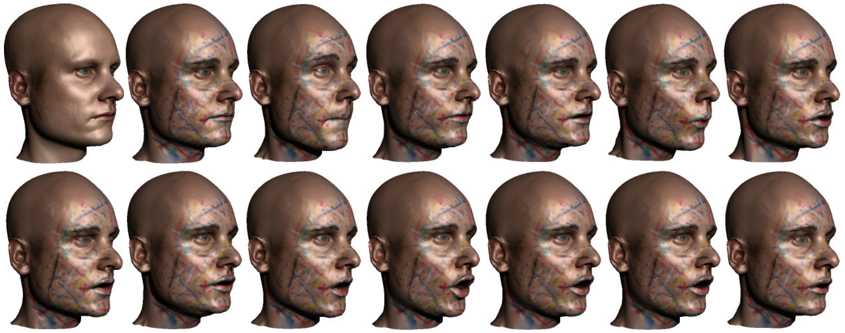


Figure 3.13: We marked up four persons with additional texture such that accurate correspondences between the expressive and neutral faces could be calculated. Shown are some registration results from one person, the first two scans contain the neutral pose without and with markup, then follow a few expression scans showing different visemes.

correspondences in a second registration run to improve the correspondence in ambiguous areas such as the cheeks. This process was iterated twice. The texture of the marked up scans is unusable, but we can replace it with the neutral texture under the assumption that the skin does not change its albedo when compressed or stretched. The markup on one example face is shown in figure 3.13. We are using the registered neutral scan of a person as the template when registering an expression scan. In the areas of the face where the shape does not change under expressions (top of the head, and the ears, see 3.14), we increased the stiffness by multiplying the area weighting factor with 50. This was necessary, because within the stiff regions there are no correspondences, so we have to impose this knowledge a priori. The stiffer areas are shown in figure 3.14.

3.6 Registration Results on Real World Data

In addition to the synthetic datasets we used our method to register over 1200 face surface scans acquired with a structured light system and more than 30 full head surfaces measured with an MRI system. From the data we constructed a 3D Morphable Model of the full head, including the back of the head. These two datasets have different characteristics, the MRI data is complete, but quite noisy, while the surface scans are incomplete, the hair region and the eyes are missing, but less noisy. Some of the registered datasets are shown in figure 3.15.

In figure 3.16 we show the first shape and texture principal components of the resulting model.

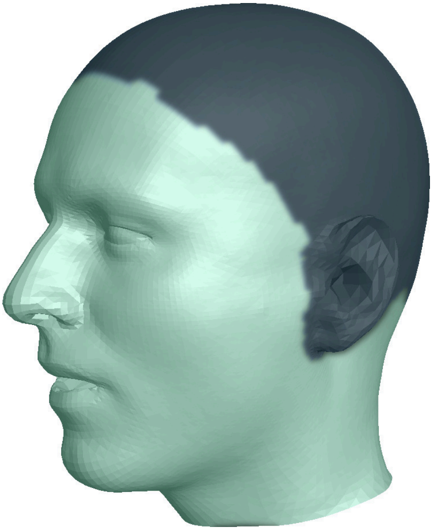


Figure 3.14: When registering an expression scan we are using the corresponding neutral scan as the template, and increase the stiffness in the area marked blue in this figure.

4 Conclusion

We have introduced a novel deformation measure to be used within (1) the non-rigid ICP framework and (2) for mesh editing. It was shown that this energy results in more plausible deformations than previous linear and nonlinear measures, while being computationally cheap.

In addition, we detailed the registration method and presented a bootstrapping method which was used to fuse data with different characteristics. This made it possible to predict convincing back-heads for structured light scanned facial surfaces based on a few MRI-Scans.

The presented algorithm is easy to implement, robust and easy to tune, and can be used to build a high quality 3D morphable model.

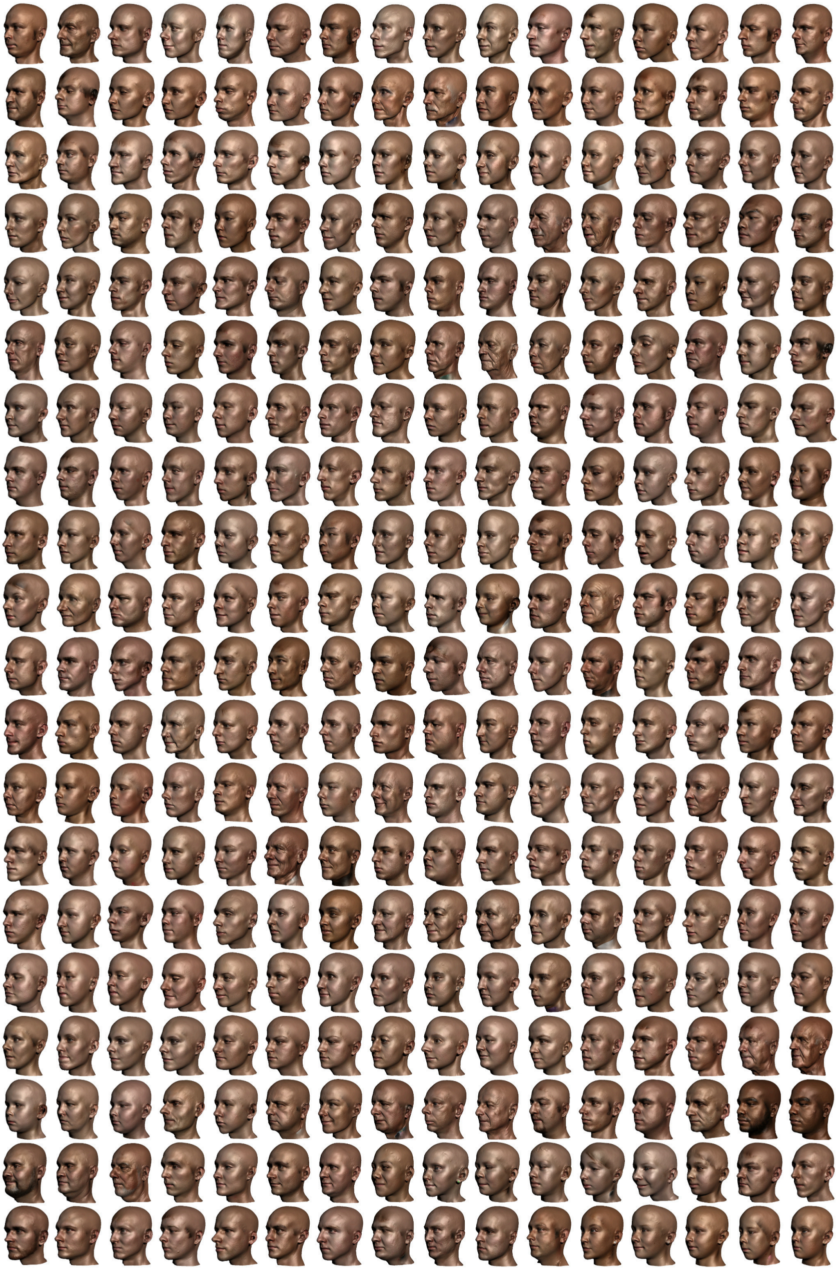
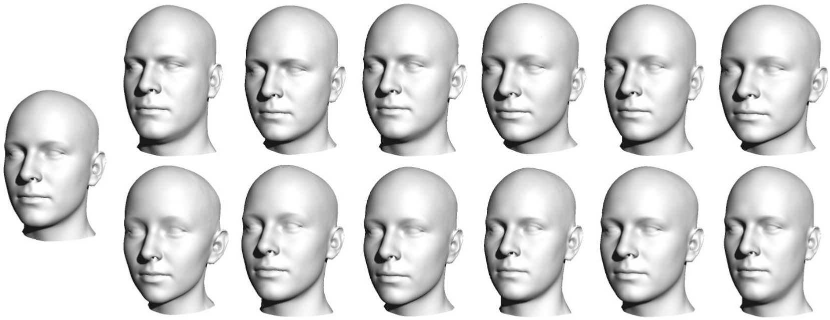
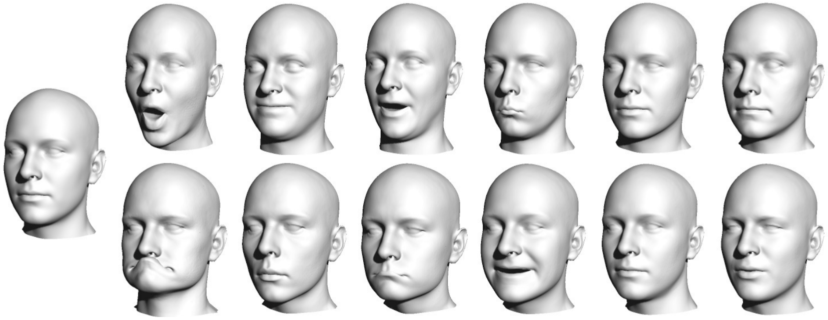


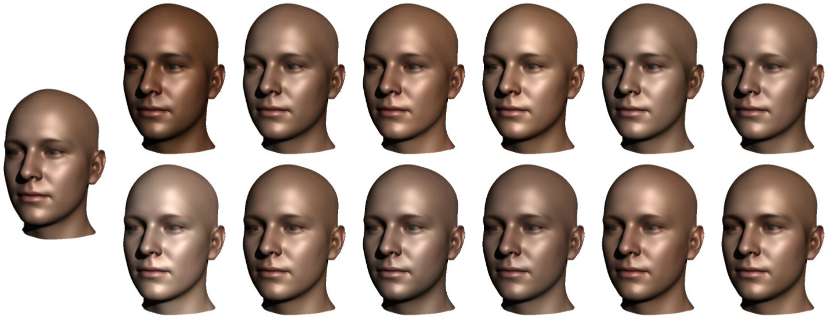
Figure 3.15: Registration results for some of the neutral scans used to build the 3D Morphable Model.



The shape mean and the 6 first principal identity components at $\pm 2.5\sigma$



The shape mean and the 6 first principal expression components at $\pm 2.5\sigma$



The color mean and the 6 first principal color components at $\pm 2.5\sigma$

Figure 3.16: A 3D Morphable Model was constructed from over 1200 scans. Shown are the first 10 shape principal components of the model for the identity and expression model together with the first 10 principal components of the color model.

— Part II —

Fitting a 3D Morphable Model to a Video Sequence

— Morphable Model Fitting — Efficient Feature Point Tracking — Simultaneous Segmentation and Fitting — Model Based Temporal Stereo Reconstruction — AAM Based Detection of Eye and Mouth Outlines —

— Chapter 4 —

3D Morphable Model Fitting

We want to apply the morphable model described in Part I to the task of transferring face expressions in videos. This requires us to fit the model to two input videos, resulting in a simplified description of the video in terms of the model parameters which best explain the video. Using this description it is then relatively simple to transfer expressions. But extracting the description is nontrivial. This chapter describes our approach of fitting the model to a video.

Fitting a 3D Morphable Model is the process of estimating the model parameters which have generated the image. More accurately, we want to determine the probability distribution over the model parameters $\boldsymbol{\theta}$ which results from observing an image $\mathcal{V}$, under the assumption that the image was generated by the model.

By exchanging the camera model it is possible to fit the same 3D Morphable Model to data from different modalities, e.g. single images (Blanz and Vetter, 1999), stereo pairs (Amberg et al., 2007a), range data (Amberg et al., 2008b,a), volume data (Lüthi et al., 2008), or, as in this text, videos. We therefore call any measurement derived from the 3D object an *image*, not just the classical pinhole camera image.

1 Problem Formulation

As the 3D-MM does not model the world and camera with perfect accuracy it has to be extended with an error term. Let us denote the image generated by the model under parameters $\boldsymbol{\theta}$ as $\mathcal{G}(\boldsymbol{\theta})$, and the error image as $\mathcal{E}$. We assume that the observed image $\mathcal{V}$ is generated as

$$\mathcal{V} = \mathcal{G}(\boldsymbol{\theta}) + \mathcal{E} \tag{4.1}$$

and we know the prior probabilities $p(\boldsymbol{\theta})$ and $p(\mathcal{E})$ of the model parameters and the error distribution. We are interested in the probability of the parameters given the observed image $p(\boldsymbol{\theta} \mid \mathcal{V})$ which, by Bayes' rule, is

$$p(\boldsymbol{\theta} \mid \mathcal{V}) = \frac{p(\mathcal{V} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{V})} \quad , \tag{4.2}$$

In practice, we are searching for the MAP estimate of the model parameters, as keeping track of the full probability would be computationally too expensive. As the maximum does not depend on the image probability we can instead maximize the unnormalized quantity

$$p(\boldsymbol{\theta} \mid \mathcal{V}) \propto p(\mathcal{V} \mid \boldsymbol{\theta})p(\boldsymbol{\theta}) \quad . \tag{4.3}$$

If the image would depend deterministically on the model, then $p(\mathcal{V} \mid \boldsymbol{\theta})$ would be a Dirac distribution – or a mixture of Dirac distributions if the mapping would not be one to one – but as the model includes an error term we have a true likelihood. For a specific modality we need to specify the likelihood and the prior from equation 4.3. Once the probabilities are defined, a gradient-based nonlinear optimizer is used to find the maximum of the posterior $p(\boldsymbol{\theta} \mid \mathcal{V})$. One could stop the explanation at this point, but the details of a successful optimization algorithm for this posterior are important, as the model involves many parameters and non-convex likelihoods. We therefore devote the rest of this part to the details of the fitting algorithm.

2 Fitting Strategy

The fitting uses a sequence of increasingly complex and less smooth approximations to the true probability of the model under the image, in the hope that this improves our likelihood of finding the global maximum. We specify the complete likelihood as factorizing into multiple terms, which depend on different features extracted from the video. By leaving out some of these factors we generate smoother approximations of the likelihood. These factors are ordered by their smoothness, and in the first optimization stages the complete likelihood is approximated by leaving out the more complex terms. Obviously, one can construct theoretical cases where this apporach leads the optimization from a perfect initialization to a local minimum, but as we never can hope to initialize at the optimium, this approach is helpful in all practical cases.

The terms which we are using for the fitting are in order of increasing complexity

1. The face prior, describing which shapes are likely faces and the camera prior which restricts the focal length to a sensible value.
2. The movement model, describing how a face is expected to move through a video.

3. The landmark likelihood, measuring how well a set of landmarks can be explained by the model.
4. The silhouette likelihood, which segments the video into foreground and background.
5. The inner edge features, i.e. the contours of the eyes, eyebrows, and lips.
6. The stereo likelihood measuring how well the optical flow induced by the model matches the video.
7. The shape from shading likelihood measuring how well a rendered model matches the image.

We will describe these terms in the following sections, but first need to introduce the imaging model used for the video fitting.

3 The Imaging Model

We model the video as a sequence of images taken by a pinhole camera with intrinsic parameters which stay constant throughout the sequence, and extrinsic parameters which change during the sequence. The extrinsic parameters define a rigid transform, which maps a vertex $\boldsymbol{v}^m$ in model space into a vertex $\boldsymbol{v}^e$ in eye space. The mapping is

$$\boldsymbol{v}^e = \boldsymbol{R}_{\boldsymbol{\rho}_{\boldsymbol{r}}} \boldsymbol{v}^m + \boldsymbol{\rho}_{\boldsymbol{t}} \quad , \tag{4.4}$$

where $\boldsymbol{R}_{\boldsymbol{\rho}_{\boldsymbol{r}}} = \boldsymbol{R}_{r_y} \boldsymbol{R}_{r_z} \boldsymbol{R}_{r_x}$ is the orthonormal matrix associated with the three Euler angles $\boldsymbol{\rho}_{\boldsymbol{r}} = (r_x, r_y, r_z)$, and $\boldsymbol{\rho}_{\boldsymbol{t}}$ are the translation parameters.

The intrinsic parameters are the focal length ρ_s in pixels, and the principal point $\boldsymbol{\rho}_{\boldsymbol{p}} = [\boldsymbol{\rho}_{\boldsymbol{p}_x}, \boldsymbol{\rho}_{\boldsymbol{p}_y}]^T$ also measured in pixels. The perspective projection function which maps a vertex $\boldsymbol{v}^e$ in eye coordinates onto the image is

$$\pi_{\boldsymbol{\rho}}(\boldsymbol{v}^e) = \rho_s \begin{bmatrix} v_x^e \\ v_y^e \end{bmatrix} / v_z^e + \boldsymbol{\rho}_{\boldsymbol{p}} \quad . \tag{4.5}$$

We will denote the function describing the image position of a model vertex i under the camera and shape parameters as

$$\Pi_i(\boldsymbol{\alpha}, \boldsymbol{\rho}) = \pi_{\boldsymbol{\rho}}(\boldsymbol{R}_{\boldsymbol{\rho}}(\boldsymbol{s}_i + \boldsymbol{S}_i \boldsymbol{\alpha}) + \boldsymbol{\rho}_{\boldsymbol{t}}) \tag{4.6}$$

We assume that the texture is constant throughout the video, and all changes in appearance are explained by surface deformations.

4 Modelling the Posterior

To determine the MAP parameters of our model given the image, we need to specify the likelihood of the observed video under the model and the model priors. We start with the priors.

4.1 Priors

We factor the prior $p(\boldsymbol{\theta})$ into a term which states for each frame $\boldsymbol{f}_i$ the likelihood of the camera parameters $\boldsymbol{\rho}^i$, shape parameter $\boldsymbol{\alpha}^i$ and for the whole video a single set of texture parameters $\boldsymbol{\beta}$, plus a motion model which describes the expected movement of the face through the scene:

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\beta}) p_{\text{motion}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) \prod_{i=1}^{F} p(\boldsymbol{\rho}^i) \prod_{i=1}^{F} p(\boldsymbol{\alpha}^i) \quad . \tag{4.7}$$

Single Frame Shape and Texture For each frame we have an independent contribution to the model prior from the its shape and texture parameters. The prior estimated when training the morphable model is a Gaussian, but we observed that the fitting results are better when using

$$p(\boldsymbol{\alpha}^i) \propto \exp\{-\left\|\max(0, |\boldsymbol{\alpha}^i| - 1)\right\|^2\} \tag{4.8}$$

$$p(\boldsymbol{\beta}) \propto \exp\{-\left\|\max(0, |\boldsymbol{\beta}| - 1)\right\|^2\} \quad , \tag{4.9}$$

instead. This distribution, which is depicted for two dimensions in Figure 4.1 assigns equal probability to all faces whose coefficients have an absolute value smaller than 1 and outside of the $[-1, 1]^N$-cube drops of like a Gaussian. With a Gaussian prior it is often difficult to find the correct regularization weight, while the prior from equation 4.9 makes the fitting more robust to different choices of the regularization. This seems to imply that equation 4.9 is a better description of the actual distribution of faces, all linear combinations of faces with small coefficients are equally probable, but it might also be just an artefact of using a maximum a posteriori estimation.

Dynamic Priors The extrinsic parameters of the camera, lighting and face model change smoothly throughout the video. This is captured by the motion model p_{motion}. One could factor this into a prior for every model parameter, which assigns a higher probability to smooth trajectories in model space. But that would introduce a large number of weights to balance the smoothness of the different model parameters. And as the rotation, translation, and shape parameters do not

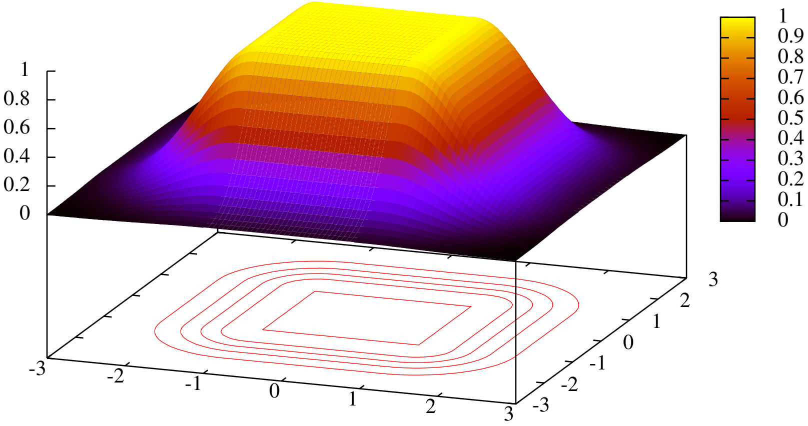


Figure 4.1: Two dimensional slice of the (unnormalized) prior used for texture and shape. The prior assigns a uniform probability density probability to all faces whose paramters do not exceed the [-1,1] range.

Introduction

1.2 Performance Specifications

Specification	A301f
Sensor Type	Sony ICX074AL/AK - 1/2 inch, HAD, interline transfer, progressive scan CCD
Pixels	658 (H) x 494 (V)
Pixel Size	9.9 µm (H) x 9.9 µm (V)
Anti-Blooming	1:100
Max. Frame Rate	80 frames/sec
Video Output Signal	Mono: 8 bits per pixel, IEEE 1394 Compliant Color: 8 bits per pixel, raw data
Gain and Brightness	Programmable via IEEE 1394 bus
Exposure Time Control	Programmable via IEEE 1394 bus
Synchronization	External via External Trigger signal - ExTrig input is opto-isolated, max. 1.4 V, max. 50 mA
Power Requirements	12 VDC (± 10%), ~ 6.0 W, < 1% ripple supplied via 1394 cable
Max. Cable Lengths	1394: 4.5 m I/O: 10 m
Shock	20G with 50 repetitions in each axis
Vibration	10G (58-500Hz) for 1 hour in each axis

Figure 4.2: Determining the focal length in pixels from prior information.

have the same scale, it would be necessary to adjust these weights on a per-video basis.

Instead, a smoothness prior which assigns a high probability to model space trajectories which result in smooth eye space trajectories of the face vertices is used. This couples the extrinsic camera and the shape parameters between all frames. The second order model used here is

$$p_{\text{motion}}(\boldsymbol{\rho}, \boldsymbol{\alpha}) \propto \prod_{q=1}^{2} \prod_{i=1}^{F} \exp\left\{ -\lambda_{\text{motion}^q} \sum_{v} \left\| \nabla_t^q (\boldsymbol{R}_{\boldsymbol{\rho}^i}(\boldsymbol{s}_v + \boldsymbol{S}_v \boldsymbol{\alpha}^i) + \boldsymbol{\rho}_{\boldsymbol{t}}^i) \right\|^2 \right\}, \tag{4.10}$$

where ∇_t^q is the q-th temporal derivative of the vertex position. We evaluate this derivative with finite differences between the frames.

Intrinsic Camera Priors We constrain the camera parameters to sensible values wherever possible. Even though our fitter can optimize over all parameters jointly (bundle adjustement), and determine the intrinsic camera parameters we can reach a higher accuracy when prior information about the calibration is available. This is done by including a prior over the camera parameters, where the nominal parameters were read of the cameras. As there is no obvious better choice we assume that the true camera parameters deviate from the nominal parameters according to a normal distribution, and we also assume that the error in the determination of the focal length and principal point are independent

$$p(\boldsymbol{\rho}_i) = \mathcal{N}(\rho_{s\,i} \mid \rho_{s\,\text{prior}}, \lambda_{\rho_s}^{-1}) \mathcal{N}(\boldsymbol{\rho}_{\boldsymbol{p}\,i} \mid \boldsymbol{\rho}_{\boldsymbol{p}\,\text{prior}}, \lambda_{\boldsymbol{\rho}_{\boldsymbol{p}}}^{-1} \boldsymbol{I}_2) \quad . \tag{4.11}$$

If no camera parameters are available we constrain the principal point to lie close to the center of the image and penalize focal lengths smaller than 5, because these result in strong wide-angle lens effects, which we do not expect to encounter in our scenes.

4.2 Likelihood

The image likelihood $p(\mathcal{V} \mid \boldsymbol{\theta})$ is factored into independent terms,

$$p(\mathcal{V} \mid \boldsymbol{\theta}) = p_{\mathcal{L}}(\mathcal{L} \mid \boldsymbol{\theta}) p_{\mathcal{S}}(\mathcal{S} \mid \boldsymbol{\theta}) p_{\mathcal{E}}(\mathcal{E} \mid \boldsymbol{\theta}) p_{\text{flow}}(\mathcal{V} \mid \boldsymbol{\theta}) p_{\text{sfs}}(\mathcal{V} \mid \boldsymbol{\theta}) \tag{4.12}$$

for landmarks, silhouette, inner edges, stereo and shape from shading, which we will describe now, starting with the smoothest factors and adding more and more complexity.

Landmark Points A landmark is a correspondence between a model vertex and an image position. Landmarks are the feature resulting in the smoothest likelihood, and are also the most informative feature. In a video, if we had the correspondence between enough landmarks and image positions for multiple frames under different poses, then this would be sufficient to exactly identify the generating parameters. The problem is that they are also most difficult to find. We extract the landmarks from the video in a semi-automatic process, where the user marks a landmark in one or more frames, and the position in the remaining frames is found automatically. This is described in detail in chapter 5.

The problem with landmarks is, that there are only few facial landmarks whose image position can be uniquely determined without first fully reconstructing the scene. These points are corners of face features, e.g. the corners of the mouth and eyes, or the ridge of the lips.

If we have a set of landmarks $\mathcal{L}$, where each landmark $\boldsymbol{l}$ is described by a vertex index $i_{\boldsymbol{l}}$ in the morphable model and an image position $\boldsymbol{x}_{\boldsymbol{l}}$ in frame $f_{\boldsymbol{l}}$ of the video, then the landmark likelihood $p_{\mathcal{L}}$ measures the difference between the projected landmarks and the 3D positions as

$$p_{\mathcal{L}}(\mathcal{L} \mid \boldsymbol{\theta}) \propto \prod_{\boldsymbol{l}} \exp\left\{-\lambda_{\mathcal{L}} \left\|\Pi_{i_l}(\boldsymbol{\alpha}_{f_l}, \boldsymbol{\rho}_{f_l}) - \boldsymbol{x}_{\boldsymbol{l}}\right\|^2\right\} \quad . \tag{4.13}$$

$\Pi_{i_l}(\boldsymbol{\alpha}, \boldsymbol{\rho})$ was defined in equation 4.6 and is the projection function mapping the model vertex $i_{\boldsymbol{l}}$ to its image position. This assumes as usual that the deviation between the clicked landmark and what can be explained by the model is normally distributed and independent.

Silhouette The visible silhouette of an object gives an important constraint on the shape of the object, and is relatively easy to detect in controlled scenes with a suitable background. It can also be arbitrarily difficult to detect in the presence of cluttered backgrounds, when the background contains itself faces, when the person has long hair of a color similar to the skin, or in a number of other cases.

Many shape reconstruction methods make use of the visible silhouette (Delamarre, 2001; Romdhani. and Vetter, 2005; Ilić et al., 2006; Vogiatzis et al., 2006; Keller et al., 2007; Amberg et al., 2007a). Plaenkers and Fua (2002) interlocked the estimation of the shape and silhouette, a topic which we expand upon in chapter 6.

Given an extracted silhouette, the corresponding likelihood is the same as in (Amberg et al., 2007a), we integrate the distance of the visible silhouette in the model towards the closest silhouette position found in the image.

$$p_{\mathcal{S}}(\mathcal{S} \mid \boldsymbol{\theta}) \propto \prod_{i \,\in\, \text{Vertices on the silhouette}} \exp\left\{-\lambda_{\mathcal{S}} \text{Distance}_{\mathcal{S}}(\Pi_{\boldsymbol{l}_i^v}(\boldsymbol{\alpha}_{\boldsymbol{l}_i^f}, \boldsymbol{\rho}_{\boldsymbol{l}_i^f}))^2\right\} \tag{4.14}$$

We extend prior work in two ways: First, we motivate the extraction of silhouettes as a way of explaining the full video, not only the face region; and second, we give an iterative algorithm which simultaneously estimates the silhouette and the model. The details are given in chapter 6.

Inner Face Edges Similar to the silhouette term is the inner face edges term. We extract the outline of the lips and eyes using part specific Active Appearance Models (AAM) which are fitted with the CoDe technique introduced in Amberg et al. (2009a). This optimization technique proved to have a larger convergence range than directly fitting the 3D model, and allows us to extract the outlines of the eyes and lips without explicitly modelling the light. The probability is defined equally to the silhouette term, but now we integrate over the fixed set of vertices corresponding to the line features

$$p_{\mathcal{E}}(\mathcal{E} \mid \boldsymbol{\theta}) \propto \prod_{i \,\in\, \text{Feature Vertices}} \exp\left\{-\lambda_{\mathcal{E}} \text{Distance}_{\mathcal{E}}(\Pi_{\boldsymbol{l}_i^v}(\boldsymbol{\alpha}_{\boldsymbol{l}_i^f}, \boldsymbol{\rho}_{\boldsymbol{l}_i^f}))^2\right\} \tag{4.15}$$

Stereo Term Classic stereo reconstruction assumes that multiple views of the same object from different viewpoints are available, and that these views are either taken at the same instant, or that the shape of the object did not change. They essentially find a surface which minimizes the reprojection error, that is the difference of the appearance at image positions in two frames which correspond to the same 3D point under the reconstructed shape. For a survey of 2-view algorithms refer to Scharstein and Szeliski (2002) and for the use of more than two frames Seitz et al. (2006) is the definitive reference. In our case the object is observed by a single camera and deforms between the frames. These kinds of problems are typically called nonrigid structure from motion (Bregler et al., 2000; Brand, 2001; Torresani et al., 2003). In nonrigid structure from motion the deforming shape and the camera parameters are simultaneously inferred from feature tracks, typically learning a linear shape basis simultaneously with the reconstruction. The output is the 3D position of the points corresponding to the tracked landmarks. In our case we are somewhere in between these paradigms. We infer the structure from the motion of a nonrigid object, by explaining the appearance of the video given a generative shape model. What we call the stereo term is the reprojection error of stereo methods, but the position where a point is reprojected is determined not only by the change in perspective but also by the deformation of the object.

The nonrgid reprojection error is calculated by taking for every vertex which is visible in a pair of frames the difference of the appearance at the position that the vertex is projected to in each of the frames

$$p_{\text{flow}}(\mathcal{V} \mid \boldsymbol{\theta}) \propto \prod_{i=1}^{N-1} \prod_{\substack{v \,\in\, \text{Vertices visible} \\ \text{in frame } i \text{ and } i+1}} \exp\left\{-\lambda_{\text{flow}} \left\| \boldsymbol{f}_i(\Pi_v(\boldsymbol{\alpha}_i, \boldsymbol{\rho}_i)) - \boldsymbol{f}_{i+1}(\Pi_v(\boldsymbol{\alpha}_{i+1}, \boldsymbol{\rho}_{i+1})) \right\| \right\} \tag{4.16}$$

To make this more robust and also increase the capture range we measure the appearance difference using the first eight elements of the feature vector described in chapter 5, which captures the appearance within a 19x19 patch.

Shape From Shading The shape from shading term is equivalent to the original 3D MM fitting term introduced in Blanz and Vetter (1999). It measures the difference between the lighted model and the corresponding image positions. We deviate from Blanz and Vetter (1999) in how the light is modelled and estimated. Our lighting model approximates the true lighting situation with a relatively large number of directional lights, equally spaced around the head. Each light has its own color and intensity. We use a nonstandard light model, which is very similar to the Phong (1975) model, actually the Phong model is a special case of our light model, but the fitting of our light model is simpler. The light model we are using consists of the standard diffuse term plus a linear combination of Phong terms for varying exponents. We do not use an ambient component, as there are enough lights to illuminate the whole object. That is for a vertex j of colour $\boldsymbol{c}^j$, with a normal $\boldsymbol{n}^j$, light incidient from the directions $\boldsymbol{d}_i$ with light colour $\boldsymbol{l}_i$ and a viewing direction of $\boldsymbol{v}$ we calculate the light emitted at the vertex in the viewing direction as

$$\sum_i \text{vis}^j(\boldsymbol{d}_i) \left(\boldsymbol{l}_i \circ (\boldsymbol{c}^j \langle \boldsymbol{n}^j, \boldsymbol{d}_i \rangle) + \boldsymbol{l}_i \sum_{n=1}^{10} p_n \langle 2\langle \boldsymbol{d}_i, \boldsymbol{n} \rangle \boldsymbol{n} - \boldsymbol{d}_i, \boldsymbol{v} \rangle^n \right) \quad . \tag{4.17}$$

Here $\text{vis}^j(\boldsymbol{d}_i)$ is an indicator function which is one if the light source is visible from vertex j and zero otherwise. p_n are nonnegative weights determining the slope of the specular reflections. The specular basis slopes are shown in figure 4.3, together with a few linear combinations of these basis slopes to show that building linear combinations of phong bases results in plausible specular highlights. The albedo of every vertex $\boldsymbol{c}$ is described by the linear combination $\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{A}\boldsymbol{\beta}$ of the appearance basis.

We update the light and the shape estimate in a coordinate descent, updating the light and albedo between the stages of our algorithm, and afterwards keeping them fixed while refining the shape.

Given a shape it is easy to optimize the above equation with respect to the (nonnegative) light colors $\boldsymbol{c}_i^l$, the phong weights p_n and the appearance coefficients $\boldsymbol{\beta}$. We do this using the box constrained BFGS method from Zhu et al. (1997) initializing with all light intensities set to $1/N_{\text{lights}}$ and the mean appearance.

5 Optimization Method

The huge number of variables (100 identity coefficients per video + 100 expression coefficients per frame + 1 camera parameter per video + 6 camera parameters per frame), makes it slow to simultaneously optimize all frames of a reasonable long video. We therefore perform a coordinate descend, optimizing over 30 frames of the video at a time, with an overlap of 10 frames between the optimizations. That is in each stage of the optimization we first optimize the parameters for frames 1..30, then 20..50, then 40..70 and so on. For the optimization we are using the L-BFGS-B method of Zhu et al. (1997), which is a memory limited Quasi-Newton method, using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update to the approximate Hessian matrix.

6 Alternatives

Here, the whole sequence is explained simultaneously. A different approach is to track a model through a video sequence as addressed in Muñoz et al. (2009) and Hiwada et al. (2003). While tracking is typically faster because less parameters need to be estimated simultaneously, we are achieving a higher quality by making use of all the available information.

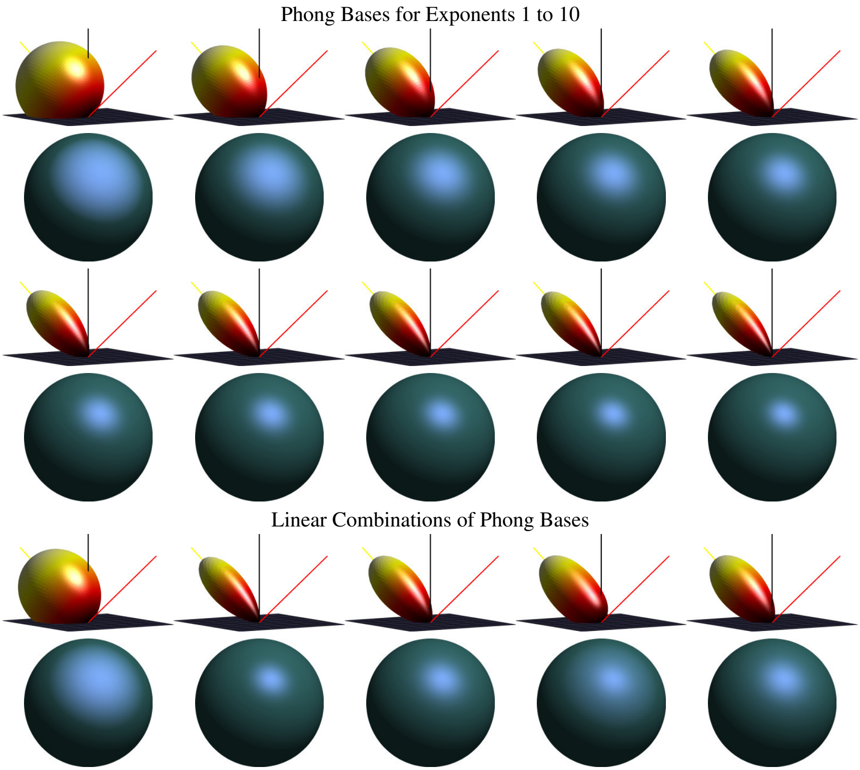

Phong Bases for Exponents 1 to 10
Linear Combinations of Phong Bases

Figure 4.3: The 10 phong bases used in our algorithm and a few linear combinations, showing that linearly combining phong bases is a sensible thing to do. Each row shows at the top the phong slope and below it a green sphere rendered under blue light with ambient + diffuse + specular lighting. The red line in the phong slope is the incident light direction, the yellow line is the mirror direction and the black line is the normal.

— *Chapter 5* —

Interactive Interest Point Tracking

When doing video post-production it is often necessary to track interest points in the video, this is called off-line tracking, because the complete video is available to the algorithm. Off-line tracking should be accurate and, if used interactively, needs to be fast, preferably faster than real-time. We describe a 50 to 100 frames per second off-line tracking algorithm, which globally maximizes the probability of the track given the complete video. The algorithm is more reliable than previous methods because it explains the complete frames, not only the part of the video which is under the track, making as much use of the data as possible. It achieves efficiency by using a greedy search strategy with deferred cost evaluation, which focuses the computational effort on the most promising track candidates while finding the globally optimal track.

1 Introduction

To successfully fit the 3D model to a video we require a few landmarks at every frame to initialize and guide the optimization. While preparing a video for expression editing, we want to minimize the amount of time a user has to spend annotating the video. We decided on a semi-automatic system, because fully automatic systems either work or fail, while semiautomatic systems always work, they only require different amounts of manual interaction depending on the difficulty of the data. Therefore, we decided to concentrate on an interactive first phase where the user accurately marks landmarks in a convenient tool. The landmarks are marked in some frames, and tracked automatically to all remaining frames. The tracking had to be fast enough to allow interactive use, while being as accurate as possible. The type of tracking we are interested in is interactive offline tracking, see for example the work of Agarwala et al. (2004); Buchanan and Fitzgibbon (2006); Sun et al. (2005); Wei et al. (2007). We decided to build on the work of Buchanan and Fitzgibbon (2006), who presented a faster than realtime offline tracker, which uses the full video at once to determine the track. We will denote their algorithm as DP-TRACK, and our descendent of their algorithm as GRAPH-TRACK. Using the full video is a systematic way to counter drift, as described by Matthews et al. (2004), which occurs when tracking only from frame to frame.

DP-TRACK achieves its high speed by preprocessing the video in a compute-intensive step without user interaction, and using the search structure from this preprocessing step for fast lookup of candidate matches. It then find a path through these candidates which simultaneously (1) passes through the landmarks, (2) minimizes the difference in appearance between the marked and detected patches and (3) minimizes the length of the track.

When the user sees that the track is lost in a frame, she can mark the missed patch, and the algorithm recalculates the optimal track over the whole video. This is the behavior which we need for our user interface, but it turned out that when the appearance of the patches changes too strongly throughout the video, the tracking becomes unstable. Correcting a bad frame can lead to losing track in regions of the video which were tracked correctly before adding the additional input.

The reason for this behaviour is that the appearance model becomes too broad, i.e. the uncertainty in the patch appearance increases. We correct this behaviour and increase the tracking stability by explaining not only the track, but instead the full video. This makes it necessary to introduce a background appearance model in addition to the foreground appearance model used in DP-TRACK. The effect is that marking an interest point in a frame does not only tell us *'this is how the interest point looks like'*, but also for every patch which was not clicked upon *'this is how the interest point does not look like'*.

2 Method

The algorithm finds the most probable track through the video given interest points marked in some frames. Denote the video by $\mathcal{V} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_F)$, where $\mathcal{V}$ is a tuple of frames $\boldsymbol{f}_i$, and a track as $\mathcal{T} = (x_1, \ldots, x_F)$, where each x_i is the position of the interest point in frame i. In $L \ll F$ frames we have marked interest points, which we denote by $\mathcal{L}$. We use Bayes' theorem to model the probability of a track given the video and the landmarks as

$$p(\mathcal{T} \mid \mathcal{V}, \mathcal{L}) \propto p(\mathcal{V}, \mathcal{L} \mid \mathcal{T}) p(\mathcal{T}) \quad . \tag{5.1}$$

The video determines the position of the landmarks, but given a track these two are independent. We write this as

$$p(\mathcal{T} \mid \mathcal{V}, \mathcal{L}) \propto p(\mathcal{V} \mid \mathcal{T}) p(\mathcal{L} \mid \mathcal{T}) p(\mathcal{T}) \quad . \tag{5.2}$$

We assume that the user has clicked correctly, such that $p(\mathcal{L} \mid \mathcal{T})$ is a Dirac distribution. This might seem like a strong requirement, but actual user input turned out to be sufficiently accurate. The extensions necessary to make the algorithm more robust to incorrectly or noisily marked landmarks are straightforward, but result in a longer runtime. Assuming correctly clicked landmarks allows us to unclutter the equations by omitting the $\mathcal{L}$ and reasoning only over tracks which pass exactly through the selected interestpoints. We will now describe how we model the prior and the video likelihood.

2.1 Track Prior

The prior over tracks is modelled as a Markov chain looking back a single frame, such that the position of a landmark in frame i depends only on the position of the landmark in frame $i-1$

$$p(\mathcal{T}) = p(x_1) \prod_{i=2}^{F} p(x_i \mid x_{i-1}) \quad . \tag{5.3}$$

Using only the previous frame, we are restricted to a first order motion model, assuming that the new position is most probably close to the old position. We therefore model the new position as an isotropic Gaussian centered on the position in the previous frame. For the first frame we are using a uniform distribution.[1]

$$\begin{aligned} p(x_0) &\propto 1 \\ p(x_i \mid x_{i-1}) &\propto \exp\{-\lambda_d \|x_i - x_{i-1}\|^2\} \quad . \end{aligned} \tag{5.4}$$

2.2 Video Likelihood given the Track

We assume that the likelihood factorizes into a term for each frame and pairs of adjacent frames

$$p(\mathcal{V} \mid \mathcal{T}) = \prod_{i} p_1(\boldsymbol{f}_i, \mid x_i) \prod_{i=1}^{F-1} p_2(\boldsymbol{f}_i, \boldsymbol{f}_{i+1} \mid x_i, x_{i+1}) \quad . \tag{5.5}$$

---

[1] This is a proper distribution, because we are considering only a discrete set of positions.

The frames where interest points are marked are used to determine the per-frame factor p_1 from positive example patches $\mathcal{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_L)$ and negative example patches $\mathcal{N} = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_N)$ extracted from these frames. The resulting appearance model is

$$p(\boldsymbol{f}_i(x) \mid \text{interestpoint}) \propto \exp\{-\lambda_f \min_j \|\boldsymbol{f}_i(x) - \boldsymbol{p}_j\|^2\} \tag{5.6}$$

$$p(\boldsymbol{f}_i(x) \mid \text{background}) \propto \exp\{-\lambda_b \min\{\delta_b, \min_j \|\boldsymbol{f}_i(x) - \boldsymbol{n}_j\}\|^2\} \tag{5.7}$$

where $\boldsymbol{f}_i(x)$ is the patch at position x in frame $\boldsymbol{f}_i$, and δ_b is a parameter giving the maximum possible distance from the background. δ_b is used to model the fact that new samples which are very far from both the known foreground and the known background samples are more probably background than foreground. The norm used here is the Euclidean distance between feature vectors extracted at each patch. The details are given in section 7, but are not necessary to follow the algorithm. Assuming independence beween the patches in a frame allows us to factor the per-frame term given the track as

$$\begin{aligned} & p_1(\boldsymbol{f}_i \mid x_i) \qquad (5.8) \\ &\quad = p(\boldsymbol{f}_i(x_i) \mid \text{interestpoint}) \prod_{x \neq x_i} p(\boldsymbol{f}_i(x) \mid \text{background}) \\ &\quad = \frac{p(\boldsymbol{f}_i(x_i) \mid \text{interestpoint})}{p(\boldsymbol{f}_i(x_i) \mid \text{background})} \prod_x p(\boldsymbol{f}_i(x) \mid \text{background}) \quad . \end{aligned}$$

This assumption is obviously wrong, as an image with independent pixels would be very boring, but is necessary to arrive at an efficient algorithm. Equation 5.8 can be further simplified to

$$p_1(\boldsymbol{f}_i \mid x_i) \propto \frac{p(\boldsymbol{f}_i(x_i) \mid \text{interestpoint})}{p(\boldsymbol{f}_i(x_i) \mid \text{background})} \quad , \tag{5.9}$$

as the product in equation 5.8 does not depend on the choice of x_i.

In the pairwise appearance term we encode the assumption that the appearance of the tracked patch changes only gradually between frames, such that

$$p_2(\boldsymbol{f}_i, \boldsymbol{f}_{i+1} \mid x_i, x_{i+1}) \propto \exp\{-\lambda_s \|\boldsymbol{f}_i(x_i) - \boldsymbol{f}_{i+1}(x_{i+1})\|^2\} \quad . \tag{5.10}$$

As opposed to the per-frame factor p_1 which explains the complete frame, our definition of p_2 explains only the appearance change of the patches under the tracked patch, and ignores the remaining area of the frames. This is necessary to keep the algorithm efficient, but might be an interesting starting point for further research.

3 Efficient Optimization

We minimize the negative log of the posterior of the track probabilities, which has the form

$$\begin{aligned} -\log p(\mathcal{T} \mid \mathcal{V}, \mathcal{L}) &= \text{constant} \\ &+ \sum_{i=1}^{F} (\lambda_f \min_j \|\boldsymbol{f}_i(x_i) - \boldsymbol{p}_j\|^2 - \lambda_b \min\{\delta_b, \min_j \|\boldsymbol{f}_i(x_i) - \boldsymbol{n}_j\|^2\}) \\ &+ \lambda_s \sum_{i=1}^{F-1} \|\boldsymbol{f}_i(x_i) - \boldsymbol{f}_{i+1}(x_{i+1})\|^2 \\ &+ \lambda_d \sum_{i=1}^{F-1} \|x_i - x_{i-1}\|^2 \quad . \end{aligned} \tag{5.11}$$

DP-TRACK uses a dynamic programming approach to minimize this cost. This dynamic programming method fails to find the global optimum when handling occlusions, as is demonstrated in the appendix. For our method we observe that the cost function can be encoded as a graph and minimized efficiently and globally optimal with a shortest path search. We use a version of Dijkstra's (1959) shortest path search which speeds up the search and allows us to efficiently incorporate our background model.

We now describe how to map the cost to a graph. We can interpret the cost as a directed acyclic graph. This graph has one layer for every frame, and in each layer a node $n_{i,x}$ for every patch. Each node has a weight corresponding to the single frame match

$$w(n_{i,x}) = -\log p_1(\boldsymbol{f} \mid x) \quad . \tag{5.12}$$

The nodes of one frame are connected to the nodes of the next frame by edges $(n_{i,x_i}, n_{i+1,x_{i+1}})$ with a weight consisting of the movement prior and the pairwise appearance term

$$w(n_{i,x_i}, n_{i+1,x_{i+1}}) = \lambda_s \|\boldsymbol{f}_i(x_i) - \boldsymbol{f}_{i+1}(x_{i+1})\|^2 + \lambda_d \|x_i - x_{i-1}\|^2 \tag{5.13}$$

The frames where the interest points were marked contain only a single node at the interest point. This ensures that all paths run exactly through the selected points. The graph has two additional special nodes, the source, which is connected to all patches of the first frame, and the sink, which is the target of all patches of the last frame. See figure 5.1 for a small example. Every path from source to sink passes through exactly one node of every layer, as each node is only connected to the directly following layer. The possible paths map therefore one-to-one onto the possible tracks, and the sum of the costs of all edges and

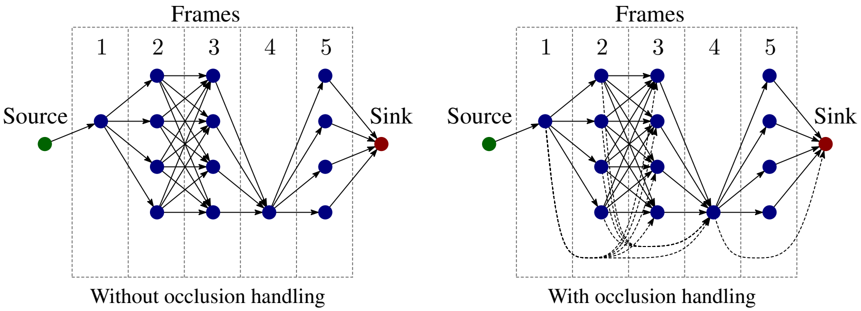


Figure 5.1: The cost function can be interpreted as a directed acyclic graph with weights on the nodes and edges. The optimal track is the path from source to sink which has the minimal weight. On the left is a graph for the cost function without occlusion handling, and on the right the same graph with occlusion handling. Each frame has 4 candidate patches, and frame 1 and 4 have a marked interest point.

nodes of a paths is exactly equal to the cost assigned to the corresponding track. When finding the shortest path from source to sink we have therefore also found the global minimum of the cost function.

We search for the shortest paths using a variant of Dijkstra's (1959) shortest path using a Fibonacci heap (Fredman and Tarjan, 1987). Dijkstra's algorithm partitions the nodes into an 'active' and a 'solved' set. The minimal distance towards the nodes in the solved set is known, and for the nodes in the active set an upper bound on the distance is kept. An invariant of the algorithm is that each node in the active set has a distance which is larger than the maximal distance to any node in the solved set. This invariant is maintained, by greedily choosing the node from the active set which has the minimal distance, moving it to the solved set and updating the distance bound towards the descendants of the expanded node with the distance of the path through the expanded node. To do this efficiently we need a priority queue with a $O(1)$ decrease key operation. The full algorithm is:

1. The source is put into a priority queue, with an associated track length of zero.
2. The remaining nodes are inserted into the priority queue with an associated track length of ∞.
3. Iterate, until the sink is expanded:
    A. Select the node x_i from the queue, which has the minimal track length.

  B. Expand x_i: For every descendent x_j of x_i:
    a. Calculate the length of the shortest path to x_j passing through x_i, this is the distance towards x_i plus the edge cost $w(x_i, x_j)$ plus the per-frame cost $w(x_j)$.
    b. If this length is smaller than the length associated with x_j, decrease the length of x_j to the new length, set a parent pointer from $track_j$ to $track_i$ and update the priority queue.

4. Follow the parent pointers from the sink to the source to recover the shortest path

While this optimization is efficient, it is still slow to calculate the weights for all edges and nodes. Especially evaluating equation 5.8 for each node is expensive, because this involves a search over a large number of negative examples, and the graph has a huge number of edges, as every frame is densely connected to the following frame. We can overcome this by observing that, for realistic settings of the parameter weights, we only have to expand a small fraction of the nodes. The cost of the shortest path towards most nodes is larger than the cost of the minimal path towards the sink. This is especially true when the track prior favours small movements, i.e. λ_d is relatively large. We can therefore remove the cost of evaluating the edge weights by calculating them lazily in step 3.B.a of the algorithm. The number of distance and similarity evaluations is reduced from $N^2_{\text{Candidates/Frame}} N_{\text{Frames}}$ to $N_{\text{Candidates/Frame}} N_{\text{Expanded Nodes}}$.

Note that while we are, for a strong motion prior, effectively calculating costs only for a few nodes around the currently best track, this is different from methods which restrict the search to a small region around the current tracked position, as we are still searching over the full frame whenever the optimal track requires this. The search is merely arranged in the most efficient order.

To be able to lazily evaluate the appearance cost we modified Dijkstra's algorithm by using a lower bound on the track length instead of the actual track length. This lower bound is calculated by replacing the per-frame cost $w(x_j)$ in step 3.B.a with a lower bound of the frame cost if x_j has not been expanded before. The actual cost of x_j is then calculated when it is expanded. Updating the lower bound of the frame cost can lead to it being no longer the overall smallest node. In that case, instead of further expanding it, we just add it back into the priority queue. This guarantees that the globally optimal path is found while performing only the absolutely necessary amount of computation. As every lower bound is updated only once, we can guarantee that the number of times a node is taken from the priority queue is maximally doubled.

The approximate frame cost is calculated by using a restricted set of background examples. We include for each positive example the one negative example which is closest to it.

The true per-frame cost is the distance towards the closest positive example minus the clamped distance towards the closest negative example plus the unknown constant from from equation 5.8. But the constant was left out, as it does not change the position of the minimum. This can result in a negative distance, which is not solveable with Dijkstra's algorithm. To overcome this we add to all nodes an upper bound on the distance to the background. The upper bound is found by taking the distance between each candidate patch and the recduced background model. This does not change the minimal path, because every path has to go through the same number of nodes, and the offset is added to all nodes.

While the algorithm as presented minimizes the number of nodes which actually need to be expanded, it is still too slow to be used with all pixels in all frames. Instead, we select in a first step a number of candidate patches from each frame (typically 150-250 patches) which are the input for all stages of the algorithm. The efficient selection of candidates is explained in section 5.

4 Occlusions

So far, our algorithm cannot handle occlusions. The occlusion handling method introduced in Buchanan and Fitzgibbon (2006) does not result in a globally optimal solution (as demonstrated in the section), but we present a relatively efficient globally optimal occlusion handling method here.

Occlusions are modelled by introducing a new binary random tuple, $\mathcal{O} = (o_1, \ldots, o_F)$ stating for each frame whether an occlusion is happening or not. We now want to find the MAP estimate of

$$p(\mathcal{T}, \mathcal{O} \mid \mathcal{V}, \mathcal{L}) \propto p(\mathcal{V}, \mathcal{L} \mid \mathcal{T}, \mathcal{O}) p(\mathcal{T}) p(\mathcal{O}) \quad , \tag{5.14}$$

where we have assumed that occlusion and track movement are independent. We model the occlusion prior again as a Markov chain

$$p(\mathcal{O}) = p(o_0) \prod_{i=1}^{F} p(o_i \mid o_{i-1}) \quad . \tag{5.15}$$

The resulting cost could be directly encoded in the graph by adding for every pixel in every frame a node corresponding to the state of being occluded at this position. That is not feasible, as the number of nodes would be too large. An efficient method can be found by noting that the track position during a run of occluded frames depends only on the motion model, and with our first order motion model is completely determined by the endpoints of the occluded run. We can therefore combine all possible occluded subpaths between two frames into a single edge, whose weight is the minimum of all these paths. This removes all

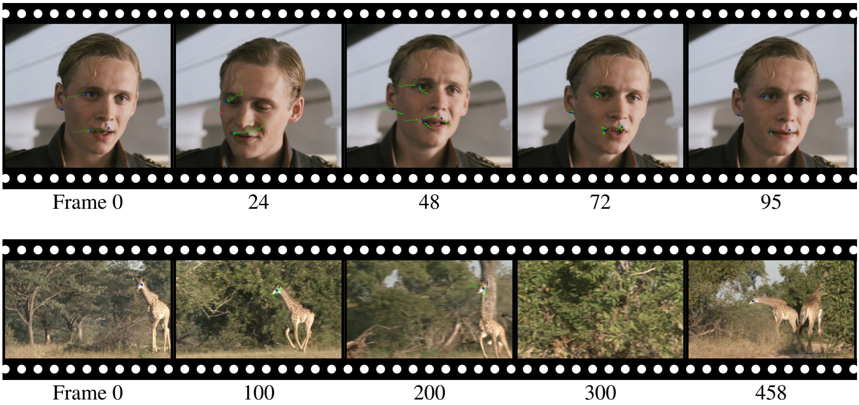


Figure 5.2: Tracking results on a talking head sequence and on a video of a giraffe. Between one and three user clicks were needed to achieve accurate tracking for the head sequence. Note the correct handling of the occluded ear, which was achieved with a single click. The eye of the running giraffe needed eight clicks, of which three marked occlusions. Please refer to the accompanying video for more details.

occluded nodes, and instead introduces on the order of $\mathcal{N}^2_{\text{frames}} N_{\text{Candidates Per Frame}}$ edges. The resulting path search is still relatively efficient, because the weights on all these edges are only calculated lazily for the expanded nodes. For long sequences with more than 100 frames, the cost of updating all nodes in the subsequent frames becomes substantial. In this case we propose to limit the number of edges by adding only edges from a node to the 10 directly following frames, then to every second frame in the range 11-20, every third in the range 21-30 etc. This assumes that the expected length of occlusions is relatively short, or that the object of interest is visible in large parts of the video. For long occlusion we might find a track which stays occluded longer than necessary. This can be remedied by the user, by marking the track position in the first non-occluded frame. Also, we speed up the update of the children by checking if the current cost plus the occlusion cost is already larger than the current minimal cost of a child. In that case we can skip the relatively more expensive calculation of the similarity cost.

Instead of specifying the four probability values of $p(o_i \mid o_{i-1})$ we are using two costs, an occlusion start cost and a cost between occluded frames, such that the cost associated with an occlusion edge from x_i to x_j is

$$\underbrace{\lambda_o + (j-i-1)\lambda_r}_{\text{occlusion}} + \underbrace{\frac{\lambda_s \|\boldsymbol{f}_i(x_i) - \boldsymbol{f}_j(x_j)\|^2}{j-i-1}}_{\text{similarity}} + \underbrace{\frac{\lambda_d \|x_i - x_j\|^2}{j-i-1}}_{\text{distance}} , \qquad (5.16)$$

where we assumed that the appearance and position change linearly during the occlusion.

5 Candidate Selection

To make the algorithm faster than real time it is necessary to restrict the number of candidate positions taken into consideration in each frame. This section explains, how we efficiently extract candidate positions from the video. The GRAPH-TRACK search algorithm is able to handle a few hundred candidates per frame efficiently, as opposed to the four to eight candidates per frame used by DP-TRACK.

Finding the candidates is essentially a template matching problem, and could be addressed by methods such as Anderson and Schweitzer (2009); Schweitzer et al. (2006); Kawanishi et al. (2004); Di Stefano and Mattoccia (2003), but for offline tracking it is allowable to invest some time into the preprocessing, if this leads to immediate feedback during the user interaction. Following Buchanan and Fitzgibbon (2006) we therefore preprocess the image such that each patch is represented by a feature vector. The details of the feature extraction are given in section 7, for now it suffices to say that we extract 16 one-byte features at each pixel position in a one-off preprocessing phase. The rest of the algorithm then works with this 16 byte representations of the image patches, allowing much faster calculations than those obtainable with a generic template matching method.

To efficiently select candidates, Buchanan and Fitzgibbon (2006) stored the patch features into a KD-Tree structure using 24 bytes per patch. While the KD-Trees are quite efficient we found that an exhaustive search can be as fast, but requires only $2/3$ the amount of memory allowing the handling of 1.5 times longer sequences. Another advantage is that with the efficient feature extraction described in section 7 and without the need to construct the KD-Trees the preprocessing time dropped from hours (Buchanan and Fitzgibbon, personal communication) to minutes. The candidates are chosen to be the $N_{\text{candidates}}$ patches which are closest to the positive examples and are locally minimal. To make the search efficient we are not calculating the sum of squared differences while deciding on the candidates, but instead use the sum of absolute differences as a proxy function. The sum of squared differences cost is then calculated only for the examples

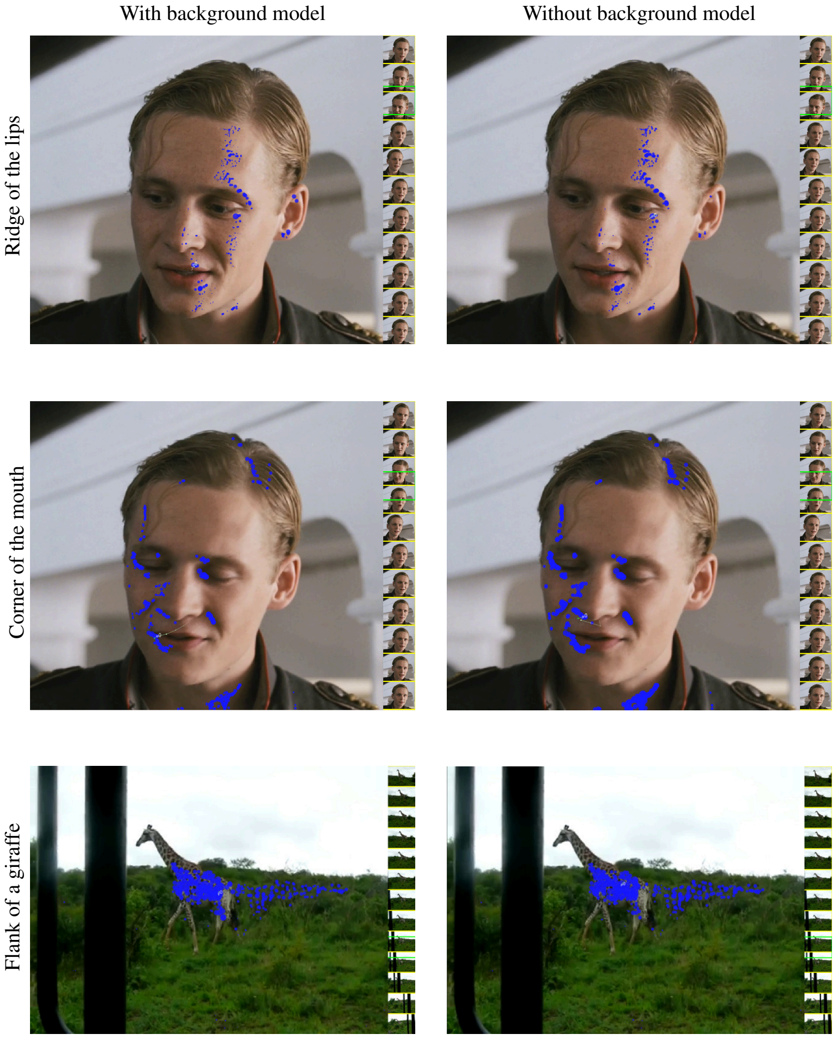


Figure 5.3: The candidate points which were taken into account for a single frame of a video sequence. We show the matches for the ridge of the lip, the right eye corner and the flank of a giraffe. The size of the dots corresponds to $p(\boldsymbol{f}_i \mid track_i)$, where the left column is using the background model and the right column is not using the background model. We picked three examples where the correct track (shown as a microscopically small green line) is found only when including the background model. In all sequences the landmark was marked in the first frame.

selected based on the proxy function. The SSE instruction set of modern CPUs contains a command to calculate the sum of absolute differences between two pairs of unsigned eight byte vectors in a single instruction (Intel, 2010). By organizing the data accordingly we can efficiently calculate the differences. While computing the differences we are performing a non minimum suppression. This requires access to the costs of the current image row and the last row. This fits into the processor cache, resulting in a fast algorithm. Already during the run we are choosing the top $N_{\text{candidates}}$ positions with a heap data structure.

We additionally require that all candidates have a feature space distance of less than 72 from the positive examples, where the threshold of 72 was selected manually once on some test sequences, and proved to work well for all other videos.

6 Accuracy

As in DP-TRACK we follow the path search with a refinement step, where the best SSD match of the 15×15 image patches within 8 pixels of the track to the image patches of the positive examples is found.

7 Feature Extraction

The algorithm consists of an offline phase and an online phase. In the offline phase we calculate a 16 byte feature vector for every patch in the input video. The features are independent from the track, such that the offline calculations have to be done only once per video. We use the features proposed in Buchanan and Fitzgibbon (2006), where a linear filter jet (Schmid and Mohr, 1997), is used, which is adapted to the video under consideration. The features are a projection of the patches p_i into a PCA basis of all patches of the video:

$$\boldsymbol{f}(x_i) = \boldsymbol{U}^T(p_i - \bar{p}) \quad . \tag{5.17}$$

Here $p_i - \bar{p}$ are the patches centered by the mean of all patches in the video. We use the 16 basis vectors of the PCA which have the largest associated eigenvalues and explain most of the variability observed in the video. The PCA basis is trained on the video, by sampling 1/8th of the patches of every frame.

The feature extraction runs at about 20 frames/minute on a commodity PC for a VGA-sized video, when exploiting the following observations. The average over all the patches of all the frames is a patch with a constant color corresponding to the mean color of the video. The reason is, that when extracting all patches every pixel of the video occurs exactly once at every patch pixel position. Therefore instead of calculating the average patch, it is sufficient to calculate the average color, and subtract it from the input video. The patches of the resulting video are then already mean-centered. This turns the projection into the PCA basis into a convolution of the mean-centered frames with the basis vectors. This is implemented efficiently using FFT, by using the well known fact that a convolution can be expressed as $A * B = \mathcal{F}^{-1}(\mathcal{F}(A) \circ \mathcal{F}(B))$, where $\circ$ denotes the elementwise product. As we need to convolve all images A_i with all kernels B_j, we can calculate the forward transform of all kernels and images, and then get the outputs by elementwise multiplication and a backward transform of all combinations (A_i, B_j).

The filter jet extracted from two different videos is shown in figure 5.4, showing that the basis adapts to the video, such that it captures as much information as possible. The features are then scaled to the range of $[0, 255]$ and quantized to eight bits.

8 User Interface

While Buchanan and Fitzgibbon (2006) proposed to let the user modify the values of the parameters, we found that when working with similar scenes it is faster to supply a few additional interest points and use a preset of parameters for the scene type. Now that we have included a background model we found that the result of adding a landmark is more predictable than that of changing the priors, and it is easier to teach new users the meaning of selecting landmarks than the exact meaning of the priors. We found that for talking heads the default parameters $\lambda_f = \lambda_b = 1$, $\lambda_s = 10$, and $\lambda_d = 10$, $\lambda_o = 5000$, $\lambda_r = 1.5\lambda_o$, $\delta_b = 4096$ gave good results over most sequences. Nonetheless different priors are needed for scenes of differing character, e.g. the giraffe sequence shown in figure 5.2 has faster motion and more occlusion, and therefore needed a smaller occlusion and motion weight.

9 Experiments

9.1 Accuracy

In figure 5.2 we show example tracks for a talking head sequence with rapid movements and pose changes, and a video of running giraffe with severe occlusions. See also the accompanying videos to get a feeling for the user interaction required to mark up these videos. As the process is interactive, and we are using a background model, we are able to track anything in any video, given enough user input.

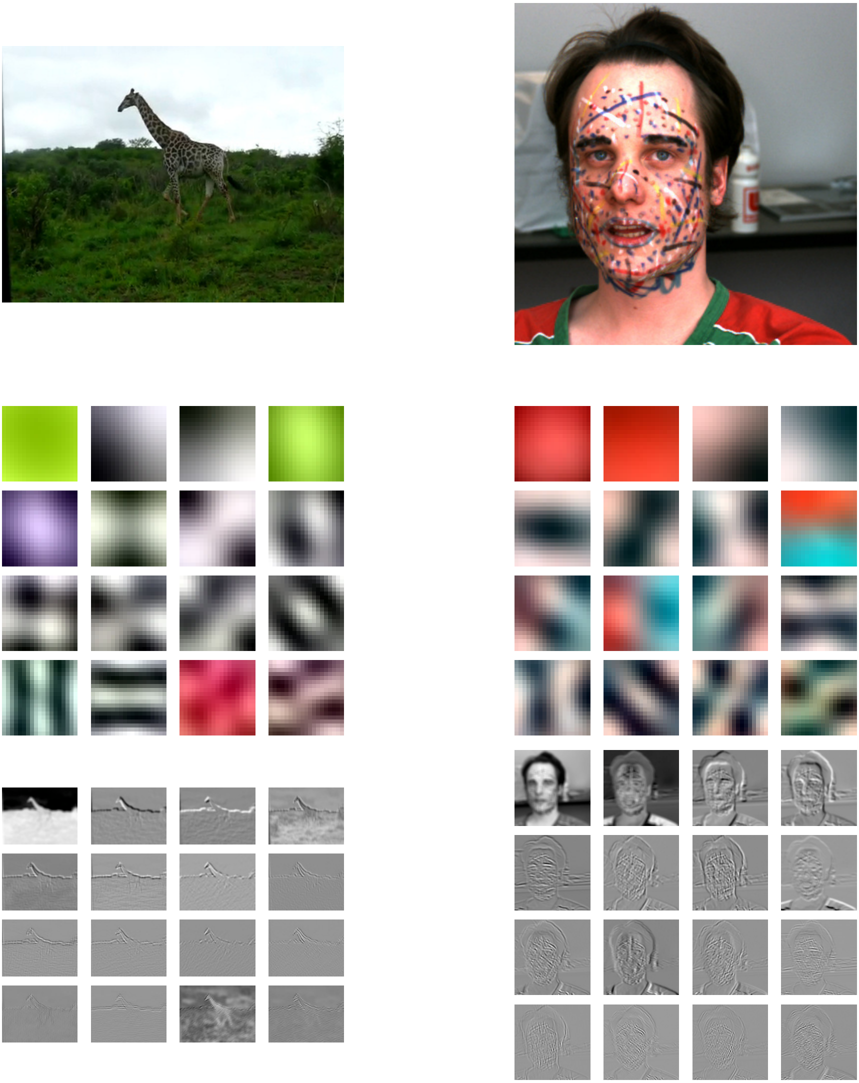


Figure 5.4: The filter jets corresponding to two different videos. A frame of each of the videos is shown next to a depiction of the filter jet, offset such that it can be visualized in RGB space. The bottom row contains the response of the features on the example frame. While the spatial structure of the jets is similar for both videos, they do differ a lot in the color distribution. Filtering the video with a specially tuned basis decorrelates the patches, and thereby removes some of the redundancy in the colour channels, without losing the available information as a transformation to grayscale would.

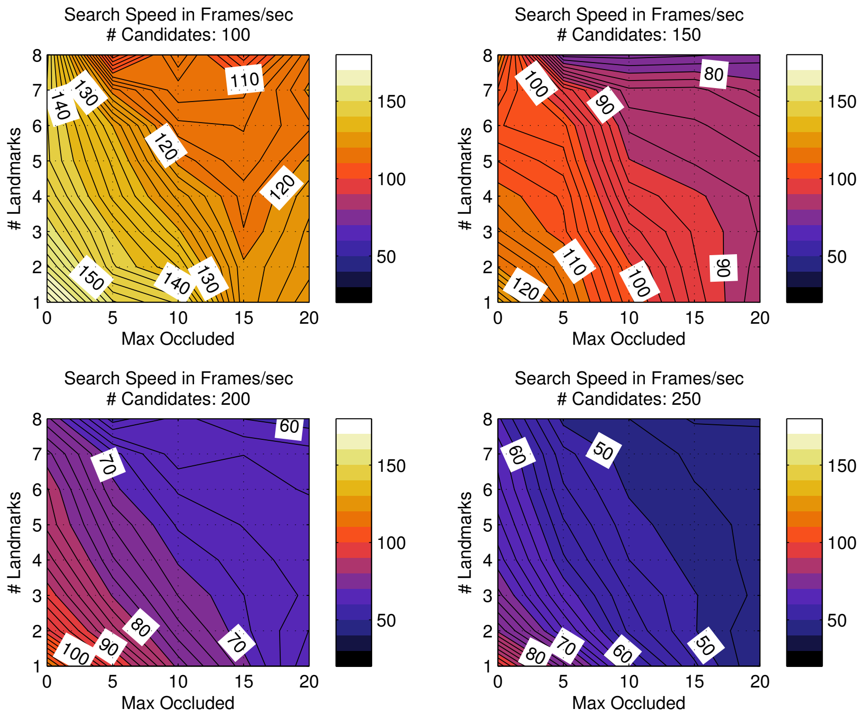


Figure 5.5: Tracking speed as a function of the algorithms parameters. The tracking time of GRAPH-TRACK is increasing approximately linearly with the number of landmarks, the number of candidate patches and the maximum occlusion length, where the largest influence is due to the number of candidate patches.

9.2 Speed

To get a feeling for the speed of the algorithm we evaluated the calculation time for varying numbers of landmarked frames (which increase the size of the foreground model, but also simplify the search graph), and for varying numbers of candidates. We did this by marking a 107 frame sequence with more landmarks than absolutely necessary and then chose all possible subsets of the marked landmarks. The experiments were done for search graphs with different maximum occlusion lengths. The results are shown in figure 5.5. For some combination of landmarks we do not find the correct tracks. In this specific example we do need four landmarks to accurately track the eye corner during all blinking events and head pose changes. The failed tracks are included in the timing and are still interesting, as it is important that the user gets immediate feedback while marking

up the video, such that she knows which frames require further attention.

As opposed to the 4 candidate patches per frame and marked interest point which were used in DP-TRACK, we are extracting between 100 and 300 candidates per frame, which makes the algorithm much more reliable on difficult frames. Our algorithm seems to scale approximately linearly in the number of frames, linearly with a small coefficient in the number of landmarks and linearly in the maximum number of occluded frames which we consider. The sweet spot seems to be at 150 to 200 candidates per frame, and with a value for the maximal occlusion length which closely matches the actual occlusion length in the video.

10 Conclusion

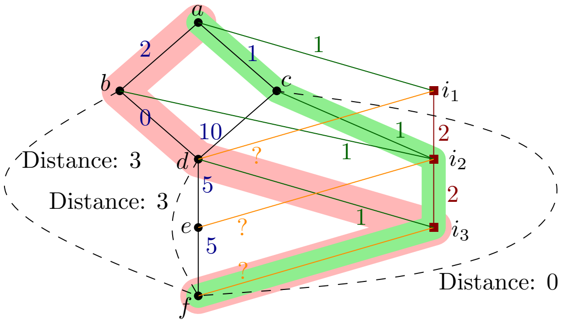
We presented an enhancement of the tracking algorithm of Buchanan and Fitzgibbon (2006), which is more reliable on difficult videos, and more stable when the user adds additional patches. This is achieved by modelling the full frame appearance, instead of only the patch appearance. Even though that seems to imply a lot more computational work, we proposed a better search algorithm which enables us to lazily perform the expensive computations, resulting in approximately the same speed at an increased reliability. The beauty of our search approach is that it adapts automatically to a stronger smoothness or occlusion prior, only performing as much computation as necessary, without absolutely restricting the search. If the appearance model gives enough evidence, arbitrarily large jumps in space and time are possible, while still being efficient.

Appendix: Why the occlusion reasoning of DP-TRACK misses the global optimum

Here we give an example where DP-TRACK fails to find the globally optimal solution for a video when occlusion handling is used. But we start with a word of warning: Buchanan and Fitzgibbon (2006) are very terse on the topic of occlusion handling, so my understanding might misrepresent the authors ideas. DP-TRACK does not fully exploit the graph structure of the problem, only the fact that the corresponding graph is layered, but there is a graph equivalent to the construction in DP-TRACK. This equivalent graph handles occlusions with a single additional node per frame, corresponding to the ‘occluded’ state. The cost of entering this node is fixed, the cost of going from an occluded state to the occluded state of the next frame is another constant, and the cost of leaving the occluded state and entering a non occluded state x_j in the following frame depends on the distance between the node x_i which is found by following the parent pointers along the current occlusion state and x_j. This greedy choice can lead to suboptimal results.

To demonstrate the problem consider the graph shown in figure 5.6 and assume additionally that the between frame similarity and appearance cost are zero everywhere. The cost of going into an invisible state is set to 1 and the cost of staying in an invisible state is 2.

The optimal path through the graph is a, c, i_2, i_3, f, with a total cost of four, while the path found with DP-TRACK is a, b, d, i_3, f, with a total cost of 6. The full result of applying DP-TRACK is given in table 5.1.



The image space distances between the nodes of the graph are:

	b	c	d	e	f
a	2	1	10	10	10
b			0	10	3
c			10	10	0
d				5	3
e					10

Figure 5.6: The computationally less intensive occlusion handling of DP-Track does not find the global minimum. This figure shows the graph of a counterexample. Shown is a graph with visible nodes $a, \ldots, f$ in five frames plus the invisible states $i_1, \ldots, i_3$ of the not landmarked frames, sink and source were ommited. The dashed lines are not edges, they annotate the distance cost which would be incurred between the connected nodes. The weight of the orange lines leading out of the invisible states corresponds to this distance cost for the node which is currently connected via parent pointers along the invisible nodes. The optimal track is marked with green, while the track found by DP-Track is marked in red.

Frame:	1	2	3	4	5
	$a = 0/\text{source}$	$b = 2/a$ $c = 1/a$ $i_1 = 1/a$	$d = 2/b$ $i_2 = 2/c$	$e = 7/d$ $i_3 = 3/d$	$f = 6/i_3$

Table 5.1: The DP-Track dynamic program corresponding to the graph in figure 5.6. Each entry contains the name of the node, the total cost up to this node and the pointer to the parent node.

— Chapter 6 —

Modelling the background for accurate shape extraction

Morphable model fitting algorithms which do not include a silhouette term have a tendency to shrink, that is to create a reconstruction whose silhouette is inside the face to be fitted, leaving a small gap towards the true silhouette. This was the motivation in Romdhani. and Vetter (2005); Amberg et al. (2007a) and Knothe (2009) to include a silhouette term. Here, we describe a novel way of extracting the silhouette feature, and a new view on the reason for using a silhouette term.

1 Why should one use a silhouette term?

We claimed that the factors in equation 4.12 describe the probability of the image given the model parameters. Without the silhouette term this is not correct, the remaining factors are only modelling the part of the image which is covered by the face, which we can write as

$$p(\mathcal{V}|_{\text{face area}(\boldsymbol{\theta})} \mid \boldsymbol{\theta}) \quad . \tag{6.1}$$

This is not a proper likelihood, as the observation changes with the parameters. We therefore should include a term describing the probability of the background. The problem with the background is that it can include anything, even faces. The only thing which we assume about the background is that it is possible to distinguish the face from the background. We should therefore include a term measuring the likelihood of the segmentation of the image into the face of interest and the rest of the image under a set of parameters $\boldsymbol{\theta}$, and this is what we accomplish with our silhouette term.

2 How to detect the silhouette

The fact that we do not have a priori assumptions about the background makes it very difficult to detect the silhouettes in an image. In Romdhani. and Vetter (2005); Amberg et al. (2007a) and Knothe (2009) silhouette candidates were found by detecting edges in the image and minimizing the distance between the model silhouette and the closest edge candidate. The problem with all these approaches, is that the strength of an edge is no indicator of the probability of that edge being a silhouette edge. Often there is a stronger edge of exactly the right shape resulting from the cast shadow, the background is noisy resulting in a dense edge response, or the person has a beard resulting in edges inside the face. Romdhani. and Vetter (2005) used the distance to the closest candidate with an orientation similar to that of the current guess instead of measuring the distance towards the closest edge. This resulted in a more robust fitting. We take the idea of determining the correct silhouette based on our current guess and propose a more efficient way to remove wrong silhouette candidates. Our method takes not only properties of the edge into account, but also properties of the image on both sides of the edge. As in Romdhani. and Vetter (2005) this requires an integration of silhouette detection and model fitting.

Our silhouette detection algorithm consists of the following steps. The output of each step is shown in figure 6.1.

(a) Calculate the area of the image which is under the head and under the face area of the head using the current model estimate $\boldsymbol{\theta}$.

(b) From this mask we determine the area in which we search for the silhouette, and a prior probability on the foreground/background segmentation.

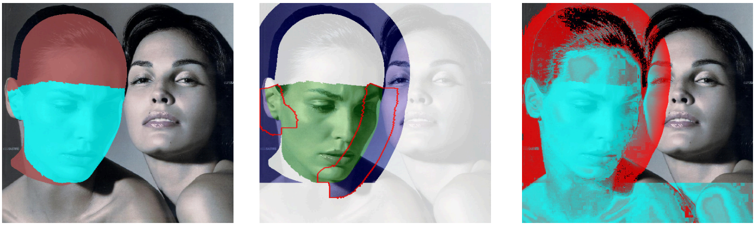
(c) Using the background and foreground prior we learn a color model for the background and foreground, and use this model plus the foreground prior to assign a foreground probability to the image. .

(d) We calculate an over-segmentation using the method of Comaniciu and Meer (2002).

(e) Within each segment of the over-segmentation we average the foreground probability.

(f) Edges between segments with a large difference in average foreground probability are chosen as silhouette candidates.

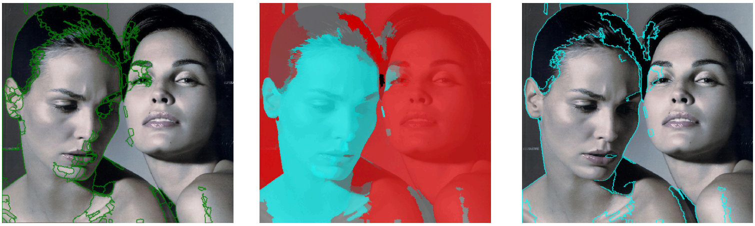
(g) We measure how well the edges are (1) close to a (potentially weak) image edge, (2) align with the current face estimate, (3) are close to the current guess, and (4) are edges between face/background in outwards direction.



(a) Current Segmentation

(b) Active Areas

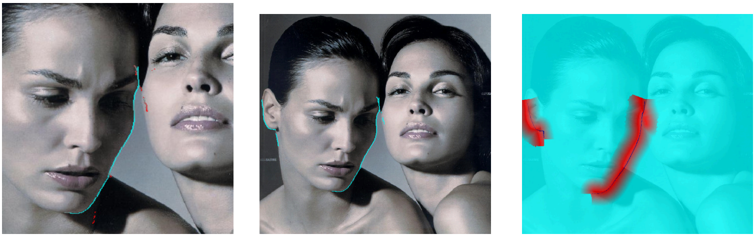
(c) Foreground Probability



(d) Over-segmentation

(e) Average Foreground Probability

(f) Candidate Edges



(g) Candidate Probability

(h) Thresholded Candidates

(i+j) Silhouette Cost and Vertices

Figure 6.1: The steps of our silhouette detection algorithm. (a) shows the segmentation proposed by the model, the head is drawn in red, and the face area marked on the model is drawn in turquoise. In (b) the red outlines mark the search area, where we expect the silhouette, the area tinted blue is used to learn the background color model, while the green area is used for the face color model. (c) shows the foreground probability derived from the color models and the current estimate. (Blue is high probability, while red is low probability.) (d) is the over-segmentation after merging the segments which are 'definitely' inside or outside according to the current estimate. (e) is the foreground probability averaged over the segments. (f) are the edges with a large foreground probability difference, (g) is for each edge the probability of that edge being the true edge, according to the factors detailed in section 2.3, and (h) is the result of thresholding this probability. (i+j) shows the resulting cost surface and the vertices for which the cost is evaluated.

(h) The edge pixels are thresholded, keeping only those edges that have a high probability.

(i) The silhouette cost surface is determined based on the distance transform of the edge candidates.

(j) Find silhouette vertices on the model. We choose those vertices which are on the visible silhouette, project inside the active search region and are close to a detected silhouette.

2.1 The area mask

The masks are set such that we search only in the stripe adjacent to the silhouette of the face, not of the back of the head. The face is marked in the 3D model, and the silhouette area is then determined from the projected mask. We ignore the region of the face which most often contains either the front or hair, as hair should be included in the background model. but we do include a stripe above the head where we expect hair to be. Everything below the chin is also ignored, as this area might as often contain skin as it contains clothing, and clothing would typically be background while skin should be classified as foreground.

2.2 Producing an accurate face classification

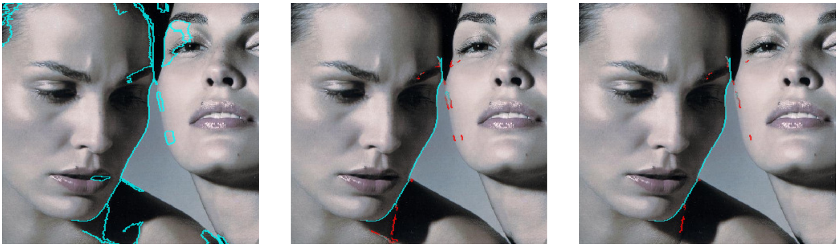
A foreground and background color model $p(\text{color} \mid \text{foreground})$ and $p(\text{color} \mid \text{background})$ is learned from the masked areas by converting the image to the LAB Color space and taking a histogram of size 128^3. The histogram is smoothed with a binomial kernel of radius 15, such that the resulting probability is a smooth kernel density estimate. The foreground probability is calculated as

$$p(\text{foreground} \mid \text{color}) = \frac{p(\text{color} \mid \text{foreground}) p(\text{foreground})}{p(\text{color})} \quad , \tag{6.2}$$

using a foreground prior of $p(\text{foreground}) = 0.5$.

Next the classification threshold is determined such that the resulting classification agrees as well as possible with the current segmentation. This can be done efficiently by considering the histograms of foreground probability for the background and foreground areas. The foreground probability is then turned into a soft classification in the range $[0, 1]$, such that the decision threshold is at 0.5. A spatial prior is added by scaling with a sigmoid function based on the distance to the current boundary, such that outside of the active range the map takes on values of zero or one respectively.

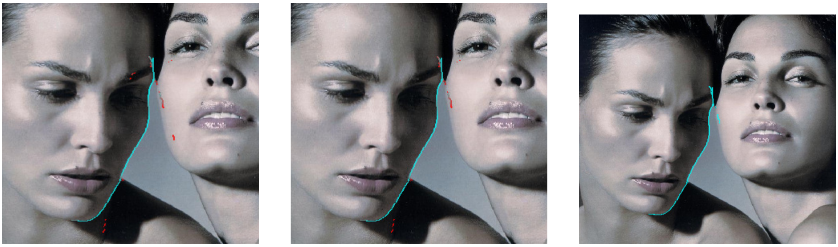
The resulting classification is often good, but is not well defined in the most interesting areas close to the boundary, where the lighting effects are strongest.



(a)
Candidate Edges

(b)
Weighted by alignment

(c)
and by step direction



(d)
and by image edges

(e)
and by distance

(f)
Thresholded Edges

Figure 6.2: Details of the edge filtering steps. The figure shows how more and more wrong edges are removed by the four filtering steps. Refer to the section 2.3 for details.

This is overcome by calculating an over-segmentation of the image using the mean-shift procedure from Comaniciu and Meer (2002). The resulting over-segmentation is simplified by merging all segments which touch the area determined to be surely foreground/surely background by the current segmentation. This area is marked white in figure 6.1(b). The foreground probabilities are then averaged over all pixels inside a segment, yielding a sharp classification.

2.3 Extracting the edges

We extract candidate edges from the foreground classification by taking all the edges that are between areas with more than 0.2 difference in foreground probability.

These candidate edges are then filtered to remove the remaining spurious responses. The filtering takes into account (1) how well the edge is aligned with the current estimate, (2) if the foreground classification is higher on the side closer to the face than away from it, (3) how close the edge is towards an image edge, and (4) how close the edge is towards the current estimate.

Edge Alignment: We check how well the edge is aligned by extracting the patch of candidate edges around each edge pixel and comparing it with the best matching current silhouette patch within a search radius. The match distance between a patch of candidate edges and a patch of the current silhouette is calculated by moving the two patches on top of each other and calculating for every edge pixel of the current silhouette the distance towards the closest candidate edge in the candidate patch. The average distance is then taken as measure of the equality of both matches.

Step direction We also test for each edge if the gradient of the foreground probability is aligned with the edge direction. This is done by smoothing the signed distance transform of the current silhouette and the averaged foreground classification maps, and taking the dot product of their gradient directions. If this dot product is smaller than 0.3 (approximately 70°) then the edge pixel is discarded.

Step size Additionally each edge-pixel is weighted by the magnitude of the difference in foreground probability.

Image edges The segments of the over-segmentation do not always concur with image edges. Especially for smooth gradients the resulting boundaries are just equally spaced. We remove these edges by requiring all result edges to be within 1 pixel distance of an image edge.

Distance to current estimate And lastly we weight the edges by the distance to the current estimate, assuming that edges closer to the current guess are more likely to be correct.

We threshold this per pixel silhouette probability to arrive at the final estimate of the silhouette, which typically contains only very few false classifications.

2.4 Determining the silhouette cost

Inside the active area, the cost map is the distance transform of the found silhouette, and outside of the active area is set to the maximal distance. Using the distance instead of the squared distance makes the silhouette term more robust against outliers.

3 Results

To test the robustness of the silhouette extraction we performed a large test on images uploaded to our web service (`faces.cs.unibas.ch`) and marked up by unknown users. The landmark positions are not perfectly consistent, because the users were not trained to use the system. Nonetheless, we extracted the silhouette from all 167 images using the same set of parameters. We then performed six iterations of fitting to the silhouette and the landmarks using up to 20 identity principal components of the model. The fitting result was then assessed by counting the the number of times that the extracted silhouette was correct after the sixth fit. The counting was performed manually, as it is difficult to devise an objective measure of the accuracy and completeness.

Out of the 167 images the silhouette extracted after six iterations was wrong for 17 examples. To put this into perspective we show in figure 6.3 and 6.4 a few successful and all the unsuccessful cases. The problematic cases either are lighted from behind resulting in a glow around the edges, have a background color which equals the face color, or have hair completely occluding the ears Please note that this evaluation is flawed, because we tuned the parameters on the test set. At least the training set is very difficult and varied, and exactly representative of the data we expect.

4 Conclusion

We propose to interlock image segmentation and fitting to correctly model the image background. Here we presented an algorithm which was tuned to work on data typical for our setting. While the problem seems easy, it turned out that an algorithm which works reliably on many images is relatively complex and contains a number of manually tuned heuristics. The algorithm presented here learns the segmentation hints from the image to be segmented and the current guess. It will also be interesting to compare this to a method which learns the segmentation from a fixed set of training images, hopefully resulting in a simpler method.

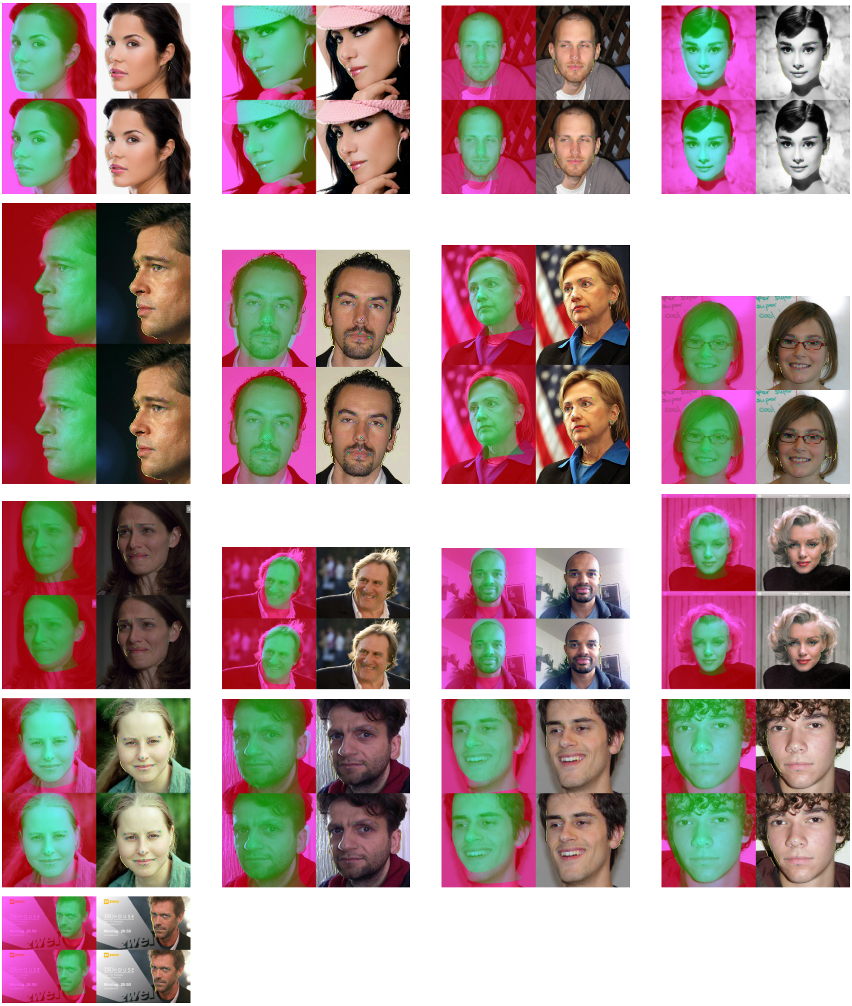


Figure 6.3: Failed silhouette extractions. The problematic cases either are lighted from behind resulting in a glow around the edges, have a background color which equals the face color, or have hair completely occluding the ears. The figure shows for each failed example the model fit after landmark fitting, the first extracted silhouette, the model fit after the sixth fit to the silhouettes, and the final extracted silhouettes.

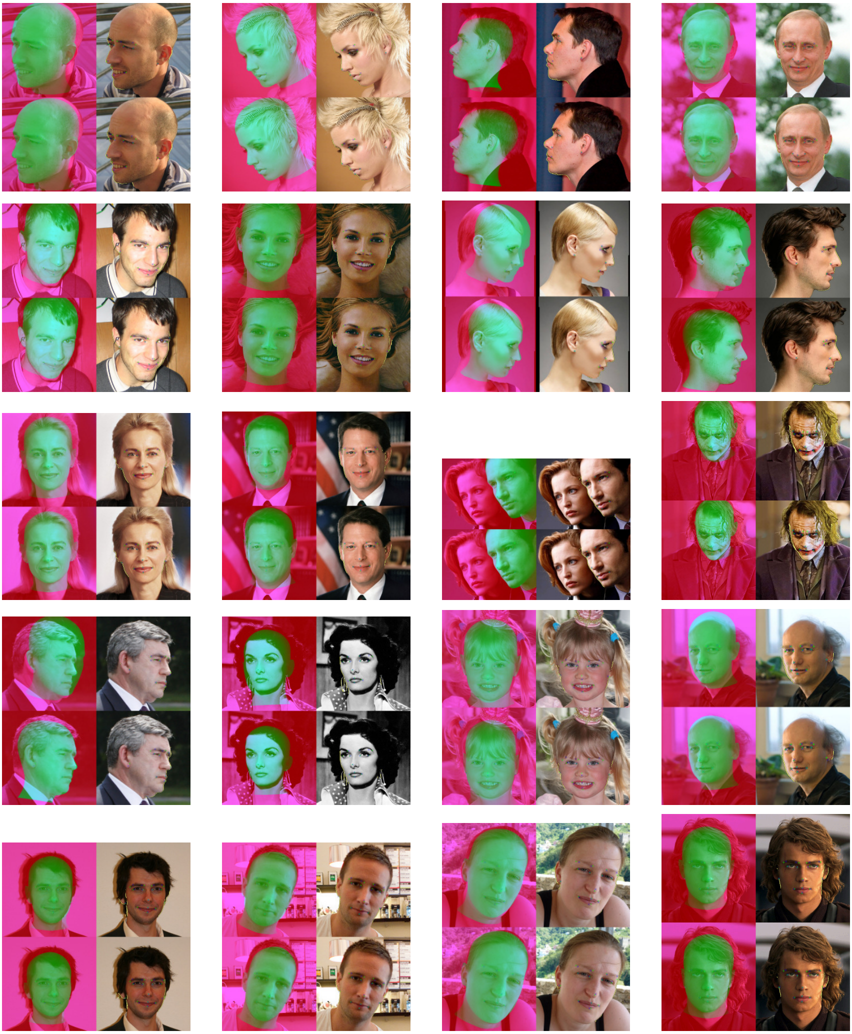


Figure 6.4: Some successful silhouette extractions. In most cases silhouette extraction returns few false positives, and allows the fitting to converge on the correct silhouette. We picked a few examples which were difficult because either the initialization was relatively far from the solution, the background color and foreground color were similar or the background has strong edges. All failures to converge are shown in figure 6.3

— Chapter 7 —

Feature Contour Detection with Active Appearance Models

2D Active Appearance Models (AAM) describe deformation and aperance change of two dimensional objects with a linear model plus a similarity transform. Parts of the face such as the mouth or the eye when seen from the same angle can be well approximated with this model, even under varying illumination. We use part and viewpoint specific AAMs to extract the contours of eyes, nose, and lips during the fitting.

In this chapter we describe an efficient and effective method to fit 2D Active Appearance Models. Previous efficient 2D AAM fitting methods have a relatively small convergence range because of a number of approximations made to speed up the fitting. Our method is constructed to be efficient without needing as many approximations, resulting in a more robust fitting method with a larger convergence range.

Active Appearance Models (AAM) as proposed by Cootes et al. (2001) and Matthews and Baker (2004) are generative 2D models, which describe linearly the variation of shape and appearance of a set of textured objects. Shape is expressed by warps of a reference shape (the "model warp"), possibly accompanied by a shape prior. Texture is expressed as a linear model in the reference frame. Due to their simpler model they are less constrained than 3D Morphable Models, but they can be fitted more efficiently. While an AAM can not model whole 3D faces accurately under rotations, it is a good approximation for parts of the face such as the eyes under a limited range of out-of-plane rotations. We therefore use AAMs of face parts to robustly and efficiently extract the contours of the eyes and lips. This helps the fitting of the 3D model, because the shape from shading likelihood has too many local minima to successfully find the global optimum, and the likelihood based on the eye and lip contours is much simpler to fit. It might not be obvious that it is easier to find only the eye contours than all the contours at once, because when searching for only the contours of one eye we are not exploiting information about the other eye or the overall shape of the face. Nonetheless, we observe that fitting a part AAM is more robust than fitting the whole face with a 3D model. The part and viewpoint specific AAM has a larger convergence range, and can be fitted much more efficiently. Effectively, we use AAMs as specialized edge detectors to preprocess the image.

AAMs are very popular because of the availability of efficient fitting algorithms. The challenge in developing fitting algorithms is to find a good tradeoff between runtime and performance, where performance may be characterised in terms of "capture range" — the range of starting points, relative to the optimal fit, from which the algorithm converges.

Fitting algorithms for AAMs are of two kinds: analysis-by-synthesis and regression-based. In analysis-by-synthesis, a generative model is fitted to data by iteratively minimizing the residual between the synthesized and the observed image (Baker and Matthews, 2004; Matthews and Baker, 2004; Blanz and Vetter, 1999). Regression-based methods use a learned mapping from the residual to parameter updates. The regression may be linear, as in the seminal work of Cootes et al. (2001), or nonlinear as in Saragih and Goecke (2007); Liu (2007); Wu et al. (2008); Wimmer et al. (2008).

In this chapter, we revisit the so-called *Compositional* approach to analysis-by-synthesis. Compositional image alignment, which originates from the thesis of Diehl (1988), seeks to reduce the residual by successively applying (composing) *incremental warps*. The use of incremental warps lends itself well to efficient approximations of the gradient and Hessian of the objective function. In some cases the Hessian is approximated as constant Diehl (1988); Burkhardt and Diehl (1986); Vemuri et al. (1998), so that it can be precomputed, which in turn leads to particularly efficient fitting algorithms. The state of the art is often considered to be the Inverse Compositional Image Alignment (ICIA) method of Baker and Matthews (2001; 2004; 2004).

In Amberg et al. (2009a) we proposed a unified framework for compositional fitting algorithms, and classified them in terms of four factors: the choice of optimization algorithm; the representation of incremental warp; the approximation of the gradient; and the approximation of the Hessian where required by the algorithm. For example, we showed that ICIA can be expressed as a Gauss-Newton algorithm in which the space of incremental warps is the same as the model warp space, the gradient is approximated in each iteration, and the Hessian is approximated by a constant. The performance of such an algorithm has limits, imposed by the validity of the approximations of the gradient and Hessian. We showed that for ICIA, the approximations are valid only when the model matches the data closely. That limits the attainable capture range. This is especially the case when test data is not closely matched by training data as, for instance, in multi-person face fitting. In this chapter we describe the two most sucessfull new algorithms from Amberg et al. (2009a), CoDe (Compositional

Descent) and LinCoDe (Linearised Compositional descent). To make the methods easier to follow we focus on explaining these algorithms instead of setting the focus on how they are related to other less efficient algorithms.

CoDe and LinCoDe are both 1st order algorithms — i.e. gradient descent algorithms. It might appear surprising at first sight that gradient descent algorithms should outperform Gauss-Newton methods. There are several reasons for this. First, Gauss-Newton is powerful, in principle, because of the availability of additional information in the Hessian, but in practice only a severely approximated Hessian can be computed. Secondly, In ICIA only an approximate gradient is used, whereas in CoDe, the exact gradient is computed. Lastly, the choice of the incremental warp space for CoDe and LinCoDe is crucial. We express the incremental warps in an orthonormal basis which makes for well-conditioned optimization and removes the need for 2nd order methods.

The performance of the new algorithms, relative to ICIA as our benchmark, is tested thoroughly over a substantial multi-person face database, with further testing of tracked facial motion on over 5000 frames of movie data. The results on these challenging datasets confirms the predictions suggested by our alignment framework. Indeed, ICIA has a very limited capture range. This is alleviated if ICIA is altered to use orthonormal incremental warps, and more so if regularisation of warps in model space is also applied. The new algorithms, CoDe and LinCoDe, used with regularisation, provide the best runtime-performance tradeoff, with CoDe able to sustain continued tracking for around 100 seconds of a movie with low resolution and demanding head-motions, and 5000 frames for a higher quality dataset.

1 Active Appearance Models

AAMs are generative models consisting of separate shape and appearance models. They are fitted to images

$$I(\boldsymbol{r}') \in \mathbb{R}, \qquad \boldsymbol{r}' \in \mathcal{I} \subset \mathbb{R}^2 \quad , \tag{7.1}$$

treated as continuous functions of the image domain $\mathcal{I}$. In this thesis we use only linear appearance models

$$\Lambda(\boldsymbol{r};\boldsymbol{\beta}) := a(\boldsymbol{r}) + \boldsymbol{A}\boldsymbol{r}\boldsymbol{\beta}, \qquad \boldsymbol{r} \in \mathcal{D} \subset \mathbb{R}^2 \tag{7.2}$$

parametrized by the coefficient vector $\boldsymbol{\beta}$ and defined over the texture domain $\mathcal{D}$. It is warped into the image by a model-warp

$$W(\boldsymbol{r};\boldsymbol{q}) = \boldsymbol{r}' \quad , \tag{7.3}$$

which is parametrized by the shape parameter vector $\boldsymbol{q}$. The model warp used in this chapter is a linear shape model $\boldsymbol{r} + \boldsymbol{M}\boldsymbol{r}\boldsymbol{\alpha}$ concatenated with a similarity transform. The warp parameters are $\boldsymbol{q} = (\boldsymbol{\rho}, \boldsymbol{\tau}, \boldsymbol{\alpha})$, with a global rotation $\boldsymbol{\rho}$, a translation $\boldsymbol{\tau}$ and local deformation coefficients $\boldsymbol{\alpha}$.

$$\begin{aligned} W(\boldsymbol{r};\boldsymbol{q}) &:= \boldsymbol{R}_{\boldsymbol{\rho}}(\boldsymbol{r} + M(\boldsymbol{r})\boldsymbol{\alpha}) + \boldsymbol{\tau} \\ \boldsymbol{M}\boldsymbol{r} &\in \mathbb{R}^{2\times N_{\text{Shape Parameter}}} \\ \boldsymbol{R}_{\boldsymbol{\rho}} &:= \begin{bmatrix} 1+\boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 \\ -\boldsymbol{\rho}_2 & 1+\boldsymbol{\rho}_1 \end{bmatrix} \end{aligned} \tag{7.4}$$

The AAM is fit to an image by minimizing the sum of squared differences between the image warped back into the texture domain $\mathcal{D}$ and the estimated appearance, the cost is

$$\begin{aligned} F(\boldsymbol{q},\boldsymbol{\beta}) &:= \|f(\boldsymbol{q},\boldsymbol{\beta})\|^2_{\mathcal{D}}\,, \\ \text{with } f(\boldsymbol{q},\boldsymbol{\beta}) &:= \Lambda(\boldsymbol{\beta}) - I \circ W(\boldsymbol{q}) \end{aligned} \tag{7.5}$$

Where $\|f(\boldsymbol{q},\boldsymbol{\beta})\|^2_{\mathcal{D}} = \int_{\boldsymbol{r}\in\mathcal{D}} f(\boldsymbol{q},\boldsymbol{\beta})^2_{\boldsymbol{r}}$ denotes the integral over the squared residual in $\mathcal{D}$ and $[I \circ W(\boldsymbol{q})](\boldsymbol{r}) = I(W(\boldsymbol{r};\boldsymbol{q}))$ is the function composition operator. We call f the residual function.

The appearance variation parametrized by $\boldsymbol{\beta}$ is handled by always choosing the appearance parameters which minimize the overall cost. That is, we are working with a cost function

$$F(\boldsymbol{q}) := \min_{\boldsymbol{\beta}} F(\boldsymbol{q},\boldsymbol{\beta}) \tag{7.6}$$

instead of the cost function from equation 7.5.

In the remainder of this chapter we will therefore leave out the dependency on $\boldsymbol{\beta}$, and choose $\boldsymbol{\beta}$ such that it minimzes the cost given the current $\boldsymbol{q}$. This method has been called the *project out* norm by Matthews and Baker (2004).

2 Fitting an Active Appearance Model

One can optimize equation 7.5 directly with a quasi-Newton method. Such a method constructs a local approximation of the cost function as a parabola, determines from this parabola a search direction and performs a line search in this direction. Constructing the parabola means estimating the function's Hessian matrix, which is quite expensive in high dimensional search spaces. We instead take the special structure of the cost function in equation 7.5 into account to find a good search direction in a cheaper manner.

Instead of approximating the original cost function with a parabola, we are approximating it with another function which is derived by prefixing the warp

$W(\boldsymbol{q})$ from the texture domain to the image domain with another *incremental* warp $V(\boldsymbol{p})$ inside the texture domain. This results in an approximate cost

$$\begin{aligned} \tilde{F}(\boldsymbol{q}_0,\boldsymbol{p}) &:= \left\|\tilde{f}(\boldsymbol{q}_0,\boldsymbol{p})\right\|^2_{\mathcal{D}} \\ \text{with } \tilde{f}(\boldsymbol{q}_0,\boldsymbol{p}) &:= \Lambda(\boldsymbol{\beta}) - I \circ W(\boldsymbol{q}_0) \circ V(\boldsymbol{p}) \quad , \end{aligned} \tag{7.7}$$

which is now parametrized in terms of a new set of variables $\boldsymbol{p}$ instead of $\boldsymbol{q}$.

2.1 Finding the update direction

The derivative of this approximate cost with respect to the incremental warp parameters $\boldsymbol{p}$ can be calculated cheaply, if we are only interested in the derivative at the identity warp $\boldsymbol{p} = \mathbf{0}$. The derivative at the identity warp is interesting, because this is where we approximated the original cost with the compositional cost. Using the chain rule we calculate the derivative as

$$\begin{aligned} &\nabla_{\boldsymbol{p}}\, \tilde{F}(\boldsymbol{q}_0,\boldsymbol{p})\Big|_{\boldsymbol{p}=\mathbf{0}} = \\ &\qquad -2\int_{\boldsymbol{r}\in\mathcal{D}} \left[\tilde{f}(\boldsymbol{q}_0,\boldsymbol{p})\Big|_{\boldsymbol{p}=\mathbf{0}}\right]_{\boldsymbol{r}} \nabla_{\boldsymbol{r}}\left[I \circ W(\boldsymbol{q}_0)\right]_{\boldsymbol{r}} \nabla_{\boldsymbol{p}}\left[V(\boldsymbol{p})\right]_{\boldsymbol{r}}\big|_{\boldsymbol{p}=\mathbf{0}} , \end{aligned} \tag{7.8}$$

where $[\cdot]_{\boldsymbol{r}}$ denotes evaluation of the functor $\cdot$ at $\boldsymbol{r}$. Now we notice that at the identity warp $\tilde{f}(\boldsymbol{q}_0,\mathbf{0}) = f(\boldsymbol{q}_0)$ and that the last term is constant because it is evaluated always at $\boldsymbol{p} = \mathbf{0}$ such that the above simplifies to

$$\nabla_{\boldsymbol{p}}\, \tilde{F}(\boldsymbol{q}_0,\boldsymbol{p})\Big|_{\boldsymbol{p}=\mathbf{0}} = -2\int_{\boldsymbol{r}\in\mathcal{D}} \left[f(\boldsymbol{q}_0)\right]_{\boldsymbol{r}} \nabla_{\boldsymbol{r}}\left[I \circ W(\boldsymbol{q}_0)\right]_{\boldsymbol{r}} \boldsymbol{G}_{\boldsymbol{r}} \quad , \tag{7.9}$$

where $\boldsymbol{G}_{\boldsymbol{r}}$ is an appropriately defined constant. We discretize equation 7.9, which allows us to evaluate the spatial derivative of the image warped back into texture space $\nabla_{\boldsymbol{r}}\left[I \circ W(\boldsymbol{q}_0)\right]_{\boldsymbol{r}}$ with finite differences. With this discretization we can calculate the gradient in a number of operations proportional to the number of pixels times the number of incremental warp components. Additionally, to determine the optimal $\boldsymbol{\beta}$ we need to project the currently backwarped image into the texture model, which requires as many operations as the number of pixels times the number of appearance parameters. That is, this part of the algorithm scales linearly with the number of pixels, the dimension of the incremental warp basis, and the dimension of the apperance basis. This is the the minimal cost we can expect to achieve without using approximations.

2.2 Line Search

The derivative gives us a search direction in the space of the incremental warp, but that was only an approximation of the underlying cost. We map from a current

estimate $\boldsymbol{q}_0$ plus an incremental warp $\boldsymbol{p}$ to a new model estimate by projecting the composed warps back into the model warp. This is expressed as

$$\boldsymbol{q}_0 \breve{\circ} \boldsymbol{p} = \arg\min_{\boldsymbol{q}^*} \|W(\boldsymbol{q}^*) - W(\boldsymbol{q}_0) \circ V(\boldsymbol{p})\|^2_{\mathcal{D}} \quad . \tag{7.10}$$

In each step we have to perform a line search over

$$\min_{\alpha} F(\boldsymbol{q}_0 \breve{\circ} \alpha\Delta\boldsymbol{p}) = \min_{\alpha} F(\arg\min_{\boldsymbol{q}^*} \|W(\boldsymbol{q}^*) - W(\boldsymbol{q}_0) \circ V(\alpha\Delta\boldsymbol{p})\|^2_{\mathcal{D}}) \tag{7.11}$$

where $\Delta\boldsymbol{p}$ is the derivative of the approximate cost and α is the length of the search step. We are using for the incremental warp a purely linear warp

$$V_{\boldsymbol{r}}(\boldsymbol{p}) = \boldsymbol{r} + \boldsymbol{V}_{\boldsymbol{r}}\boldsymbol{p} \quad . \tag{7.12}$$

with a basis $\boldsymbol{V}$, and can therefore pull the step length out of the warp, such that the line search becomes

$$\min_{\alpha} F(\boldsymbol{q}_0 \breve{\circ} \alpha\Delta\boldsymbol{p}) = \min_{\alpha} F(\arg\min_{\boldsymbol{q}^*} \|W(\boldsymbol{q}^*) - W(\boldsymbol{q}_0) \circ \alpha V(\Delta\boldsymbol{p})\|^2_{\mathcal{D}}) \quad . \tag{7.13}$$

This is useful, because we can now scale the step size to a sensible value in pixel in texture space. That is we are searching for a step size of

$$\min_{\alpha} F(\boldsymbol{q}_0 \breve{\circ} \alpha\Delta\boldsymbol{p}) = \min_{\alpha} F(\arg\min_{\boldsymbol{q}^*} \left\|W(\boldsymbol{q}^*) - W(\boldsymbol{q}_0) \circ \alpha\beta^{-1} V(\Delta\boldsymbol{p})\right\|^2_{\mathcal{D}}) \quad , \tag{7.14}$$

where β is chosen such that the average displacement in $\beta^{-1}V(\Delta\boldsymbol{p})$ is one. We start with a step size of $\alpha =$ two pixel, and reduce the step size until an improvement has been found. Whenever an improvement has been found in the first try of an iteration the initial step size for the next iteration is increased. Once the step size falls below a minimum distance of 1/20th of a pixel in texture space we stop the search.

3 Efficient Composition

So far we have described how to efficiently find a search direction and how to perform the line search with meaningful step sizes. Until now it remained unclear, what the computational cost of the warp composition is. While the composition actually makes up a large part of the overall cost of the algorithm, it is still possible to calculate the composition efficiently.

Let us write out equation 7.10 for the warps involved here. For simplicity we consider the discretization of the warp composition equation, where the integral of squares is evaluated at every pixel in the texture domain. It is also possible to derive this in the continuous case and only discretize at the end, but that is slightly more involved and leads to the same result. We denote $W(\boldsymbol{q}_0) \circ V(\boldsymbol{p})$ with $\boldsymbol{u}$, and get

$$\boldsymbol{q}_0 \mathbin{\breve{\circ}} \boldsymbol{p} = \arg\min_{\boldsymbol{q}^*} \|\boldsymbol{R}_{\boldsymbol{\rho}}(\boldsymbol{r} + \boldsymbol{M}_{\boldsymbol{r}}\boldsymbol{\alpha}) + \boldsymbol{\tau} - \boldsymbol{u}_{\boldsymbol{r}}\|^2_{\mathcal{D}} \quad . \tag{7.15}$$

The rotation matrix $\boldsymbol{R}_{\boldsymbol{\rho}}$ is orthogonal, which implies that we can multiply the interior of the norm with the (also orthogonal) inverse of $\boldsymbol{R}_{\boldsymbol{\rho}}$ without changing the position of the minimum. This results in

$$\boldsymbol{q}_0 \mathbin{\breve{\circ}} \boldsymbol{p} = \arg\min_{\boldsymbol{q}^*} \left\|\boldsymbol{r} + \boldsymbol{M}_{\boldsymbol{r}}\boldsymbol{\alpha} + \boldsymbol{R}_{\boldsymbol{\rho}}^{-1}\boldsymbol{\tau} - \boldsymbol{R}_{\boldsymbol{\rho}}^{-1}\boldsymbol{u}_{\boldsymbol{r}}\right\|^2_{\mathcal{D}} \quad . \tag{7.16}$$

We now change variables to get

$$\boldsymbol{q}_0 \mathbin{\breve{\circ}} \boldsymbol{p} = \arg\min_{\boldsymbol{q}^*} \left\|\boldsymbol{r} + \boldsymbol{M}_{\boldsymbol{r}}\boldsymbol{\alpha} + \tilde{\boldsymbol{\tau}} - \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \boldsymbol{u}_{\boldsymbol{r}}\right\|^2_{\mathcal{D}} \quad . \tag{7.17}$$

2D similarity transform matrices are linear functions of two parameters, which allows us to rewrite this in the standard form for a quadratic function

$$\boldsymbol{q}_0 \mathbin{\breve{\circ}} \boldsymbol{p} = \arg\min_{\boldsymbol{q}^*} \left\| \left[\underbrace{\begin{matrix} \boldsymbol{u}_1^x & \boldsymbol{u}_1^y \\ \boldsymbol{u}_1^y & -\boldsymbol{u}_1^x \\ \vdots & \\ \boldsymbol{u}_N^x & \boldsymbol{u}_N^y \\ \boldsymbol{u}_N^y & -\boldsymbol{u}_N^x \end{matrix}}_{=\boldsymbol{U}} \;\middle|\; \underbrace{\begin{matrix} \boldsymbol{M}_1 & \boldsymbol{I}_2 \\ \vdots & \vdots \\ \boldsymbol{M}_N & \boldsymbol{I}_2 \end{matrix}}_{=\boldsymbol{D}} \right] \underbrace{\begin{bmatrix} a \\ b \\ \boldsymbol{\alpha} \\ \tilde{\boldsymbol{\tau}} \end{bmatrix}}_{=\boldsymbol{x}} - \underbrace{\begin{bmatrix} -\boldsymbol{r}_1 \\ \vdots \\ -\boldsymbol{r}_N \end{bmatrix}}_{=\boldsymbol{b}} \right\|^2 \quad . \tag{7.18}$$

We denote now the first two columns of the matrix above as $\boldsymbol{U}$ and the rest as $\boldsymbol{D}$ and note that $\boldsymbol{D}$ is constant while $\boldsymbol{U}$ depends on the current warp. Also, we will write $\boldsymbol{b}$ for the vector of $\boldsymbol{r}$'s and $\boldsymbol{x}$ for the parameters.

The solution of equation 7.18 is given by the normal equations as

$$\boldsymbol{x} = \left(\begin{bmatrix}\boldsymbol{U} & \boldsymbol{D}\end{bmatrix}^T \begin{bmatrix}\boldsymbol{U} & \boldsymbol{D}\end{bmatrix}\right)^{-1} \begin{bmatrix}\boldsymbol{U} & \boldsymbol{D}\end{bmatrix}^T \boldsymbol{b} \tag{7.19}$$

$$= \begin{bmatrix} \boldsymbol{U}^T\boldsymbol{U} & \boldsymbol{U}^T\boldsymbol{D} \\ \boldsymbol{D}^T\boldsymbol{U} & \boldsymbol{D}^T\boldsymbol{D} \end{bmatrix}^{-1} \begin{bmatrix}\boldsymbol{U} & \boldsymbol{D}\end{bmatrix}^T \boldsymbol{b} \tag{7.20}$$

The matrix to be inverted here has only size $N_{\text{shape coefficients}} + 4$ and is nearly constant. Only the top two rows and columns are dependent on the current warp.

Inverting the matrix is cheap, as it is small, but calculating $\boldsymbol{D}^T\boldsymbol{D}$ is expensive, because these matrices are of size $N_{\text{pixel}} \times N_{\text{shape coefficients}}$. Luckily, $\boldsymbol{D}$ is constant, and the multiplication can be done offline. From $\boldsymbol{x}$ we can easily reconstruct $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$ with the recovered inverse rotation matrix

$$\boldsymbol{R}_{\boldsymbol{\rho}} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1} \qquad\qquad \boldsymbol{\tau} = \boldsymbol{R}_{\boldsymbol{\rho}}\tilde{\boldsymbol{\tau}} \quad . \tag{7.21}$$

4 The Incremental Warp

The remaining question is how to choose the incremental warp V. Possible choices range from a completely unconstrained warp to warps depending on the current estimate. While this is a very interesting topic for further research, we followed the choice of Baker and Matthews (2004) who used $V = W$. Recall that W is the model warp from the face domain onto the image. We used the linearization of W at the identity parameters, which was then orthogonalized and normalized. Orthogonalizing the warp leads to a much larger capture range, because the parameters of the incremental warp are now well scaled, such that the simple ‘gradient descent’ described in the previous section works well. Without orthogonalization and normalization it becomes necessary to include an approximation of the Hessian matrix as was done in Baker and Matthews (2004). But calculating the true Hessian results in an algorithm which is slow, and using a Hessian which is approximated as being constant as in ICIA results in a brittle algorithm. This can be seen in figure 7.3, where the the exact compositional first order methods CODE described here is compared to the approximate compositional second order method ICIA and to a direct quasi-Newton optimization of equation 7.5. For completeness we also included the remaining variants described in more detail in Amberg et al. (2009a). From the figure we see that CODE lives at the sweet spot of best in class success rate and high efficency, we therefore chose CODE for the video editing system presented in this thesis.

5 Experiments

We described in Amberg et al. (2009a) five compositional image alignment algorithms, of which ICIA is a special case. Here, we presented only CODE in detail, as that is the algorithm used in this thesis. Nonetheless, we would like to replicate the experiments from Amberg et al. (2009a) here, to justify our decision to choose CODE over the other algorithms. The fitting methods are evaluated on a relatively large multi-person AAM, on images of unseen identities. This is the most difficult but also the most typical situation for face analysis.

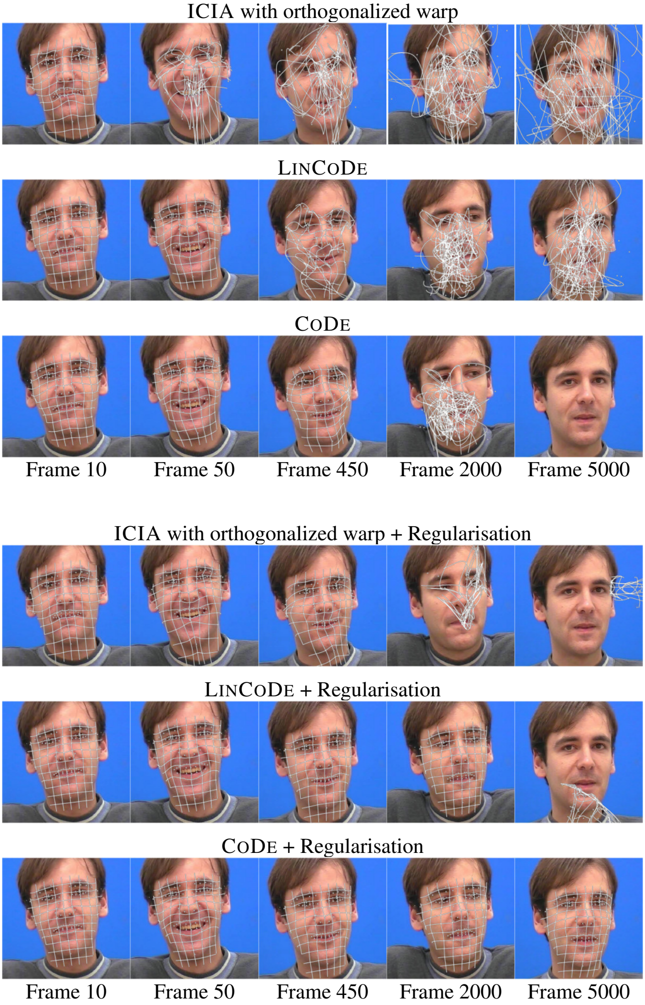


Figure 7.1: Our algorithm makes fast and robust tracking possible. We compare face tracking under natural motion, using ICIA, LINCODE and CODE. The original ICIA fails immediately with this large model and unseen face data. Substituting the orthonormal incremental warp for the original ICIA warp, the algorithm still loses track very early, whereas LINCODE and CODE can track much further. Finally, adding regularisation to all algorithms, ICIA still loses track completely after approximately 500 frames and does not recover the local deformations accurately. In contrast CODE now tracks the full 5000 frame sequence without reinitialization, and LINCODE tracks for 2500 frames.

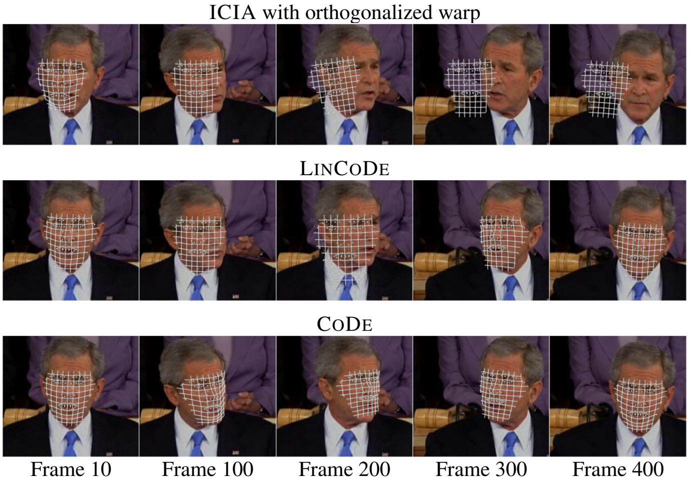


Figure 7.2: **Tracking a low resolution video with large head motions succeeds with CoDE, where ICIA fails.** All methods used the orthonormal incremental warp, and relatively strong regularisation. ICIA starts to drift in the early frames, while CoDE tracks the full sequence. The approximate gradient method LinCoDE also suceeds, but looses track of the details for about 100 frames.

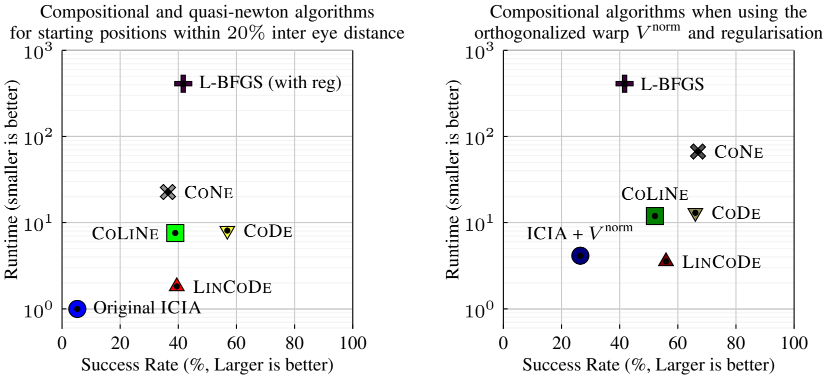


Figure 7.3: The Best speed–performance tradeoffs come from the two new algorithms CoDE and LinCoDE. Left: Without regularisation. Right: with regularisation and orthogonalized warp. Note that ICIA is practically useless on this difficult multi-person dataset with a success rate near zero (left). It can be improved (right) by using the orthonormal incremental warp and regularisation. The CoDE algorithm with regularisation (right) is as accurate as the slow, approximation-free, compositional Gauss-Newton CoNE method but is seven times more efficient.

We will show now, that the proposed CoDe algorithm has the largest capture range achieveable by any of the iterative optimisation methods described in Amberg et al. (2009a), while being only eight times slower than the fastest contender, ICIA. The differences in capture range are most pronounced when fitting a test face outside the training set. Then ICIA fails to converge for most starting positions, while CoDe converges much more reliably. Of the other methods only CoNe (Amberg et al., 2009a) can compete in success rate to CoDe, but is significantly slower.

Model We trained AAMs[1] from publicly available images with manually selected landmarks. The models are learned from 456 images from the datasets XM2VTS by Messer et al. (1999) and IMM by Nordstrøm et al. (2004), which where marked up with 120 landmarks. The data contains 62 identities, multiple expressions, light variation and up to 30 degree out of plane rotation. We used 31 identities with 248 images from the XM2VTS dataset, and 39 identities with 208 images from the IMM dataset. To increase the variability of the model, we added the mirrored version of each image to the training set. The correspondence between the models was established with thin plate splines from the manually selected landmarks. The model was calculated at a resolution of approximately 20000 color pixels. We kept 60 shape and 60 appearance basis vectors. The large variability in the training set facilitates good generalization to novel images.

Multi-Identity Fitting We trained models from subsets of all marked images in a cross-validation framework, using all images *not* from a chosen identity to build an AAM which was tested on the images of that identity. Fitting was started from randomly chosen offsets in the image plane. All fits were initialized with zero shape and rotation parameters, and the approximate size of the face in the image as the scaling factor. All variants of the algorithm were started from the same starting positions. We report the success rate, defined as the ratio of runs that converge within a distance of 5% IED (inter eye distance), averaged over the feature points, and the runtime relative to ICIA. The starting positions had up to 20% IED misalignment. The speed-performance tradeoff is summarized in figure 7.3 On the left we show the main competing algorithms. We use ICIA with the original incremental warp, LinCoDe and CoDe with orthonormal warp and CoNe and CoLiNe applied, for the first time, to AAM fitting with orthonormal incremental warp. For comparison, we also added direct optimisation of the objective function 7.5 with a quasi-Newton method, and the expensive full Gauss-Newton optimisation CoNe of the compositional cost. ICIA fails to converge reliably on this difficult but realistic dataset. The success rate of CoDe is higher

---

[1] The AAM containing the complete training set is available on our website: `www.cs.unibas.ch/personen/amberg_brian/aam/`

than that achievable with direct optimisation using a quasi-Newton method (L-BFGS), while being dramatically faster. LINCODE is as fast as ICIA but converges 8 times as often, though not quite as often as CODE. Using the Hessian approximation and the correct gradient (COLINE) performs no better than approximate gradient descent (LINCODE) but at seven times greater cost. On the right in figure 7.3, we show that adding regularisation considerably improves the capture range for all methods but CODE and LINCODE continue to give the best speed-performance tradeoffs.

Tracking We applied the algorithms to tracking video sequences. A 5000 frame sequence (Figure 7.1) of a talking face Cootes (2008) with a subject which was not in the training set and captured with a different camera and lighting was tracked with the compositional alignment algorithms. All tracks were initialized from a perfect fit to the first frame, and used the orthonormal incremental warp. ICIA immediately loses track, even though the inter-frame displacements are rather small, and this is so even when the original ICIA incremental warp is replaced with the new orthonormal warp scheme. However CODE, admittedly running 8 times slower than ICIA, and LINCODE, which runs in a time similar to ICIA, fail only after approximately 500 frames. When regularisation is added it is possible to track 2500 frames, accurately, with LINCODE and CODE even stays stable for the full 5000 frames sequence, whereas ICIA now fails after 500 frames and, even while tracking, delivers clearly inaccurate warps. Full results are in the online material.

To test the behaviour on a more difficult video we used a speech with large head motions and expressive gesture, acquired under uncontrolled lighting and with relatively low resolution of approximately 18 pixels IED. Again CODE tracks the full sequence (figure 7.2), while ICIA fails, never to recover; LINCODE temporarily loses track during a large out of plane rotation, and is on this more difficult dataset, not as accurate as CODE, showing that the fast approximation of the gradient does not come without a cost. Full tracked sequences are shown in the supplementary material.

6 Conclusion

We presented in Amberg et al. (2009a) a systematic overview of compositional image alignment methods, which gave rise to two new methods, CODE and LINCODE. The evaluation of these methods suggested, that CODE is best suited for our purpose of extracting the contours of eyes, lips and noses. In this chapter we therefore thorougly explained CODE and replicated the experimental results from the overview paper. ICIA, the fastest 2D fitting method known so far, is not usable for the fitting of unseen data. Our experiments demonstrate that CODE, while being only eight times slower than ICIA, has a large capture range making AAMs suitable to be used for face analysis.

— Part III —

Face Editing with 3D Morphable Models

— Adding Virtual Objects into the Scene — Transferring Faces, Identities and Expressions — Light and Shadow — Subtle Warping to Increase the Quality —

— Chapter 8 —

Video Manipulation

Many editing tasks can be done once the video is described in terms of the generative face model. Here, we present two examples. The first task is to add virtual objects to the image that interact with the face, and the second task is animating a face with the expressions from another video.

1 Adding Virtual Objects to the Scene

As an example we added glasses to the original face in a video sequence. The position of the glasses is adapted by rigidly aligning them with four points at the ears and nose which are typical contact points of the glasses. As our reconstruction not only describes the shape, lighting and camera but also the correspondence between the model domain (mean mesh) and the reconstructed surface, we can simply mark these points in the model domain and know their position in all frames of all fitted videos. With this correspondence the addition of glasses to a video becomes fully automatic. As we now know the 3D shape of the face and glasses we can calculate the correct occlusions and – to make the result even more realistic – also model the light interaction between the face and the virtual object, such that it casts shadows on the face, and is itself shadowed by the face. This requires a relighting of a scene. For relighting we first remove the lighting from the scene and then rerender with new light parameters.

The light model was already described in the fitting chapter. We are using a few hundred directional light sources spaced around the face, which obey shadowing and have a multiplicative diffuse component and an additive specular component. The shadow boundaries are smoothed in image space, such that the resulting shadow boundaries are soft. Using soft shadows, we can get away with less light-sources and still get realistic renderings. At each pixel we add up the contribution from all light sources, which results in a diffuse contribution d and specular contribution s for each color channel. The model describes the resulting intensity c when lighting an object of albedo a for each color channel indepen-

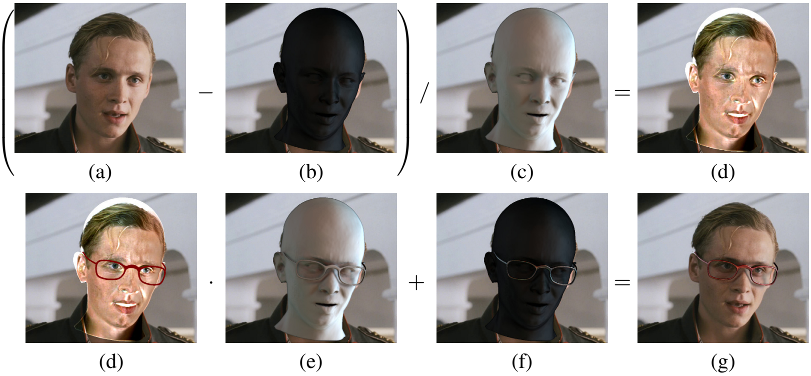


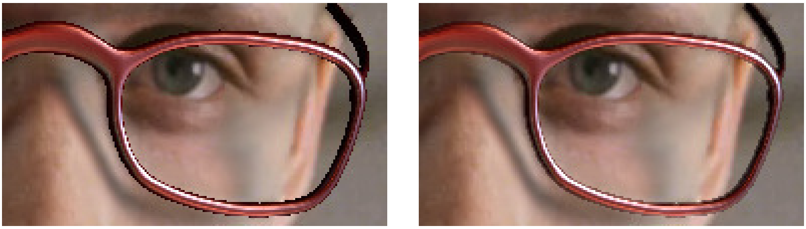
Figure 8.1: Relighting a scene by calculating the light contributions for the fitted and manipulated scene. The original image (a) is light normalized using the diffuse (c) and specular (b) components of the reconstructed scene, resulting in the albedo map (d). Then, the albedo map is manipulated (e) and the changed lighting (f,g) is added again to the light normalized scene, resulting in an image (h) which contains the new object and correctly handles occlusion and shadowing.

dently as

$$c = ad + s \quad . \tag{8.1}$$

We calculate the diffuse and specular contributions for every pixel in the original image using the fitted light and shape. This allows us to recover the albedo, that is the unlighted image as $a = (c - s)/d$. Next, we calculate the lighting for the manipulated scene now containing the new object. The albedo map is changed to the virtual objects color wherever it is in front of the reconstructed scene, and the albedo image is lit according to the diffuse and specular contributions of the manipulated scene. This is shown in figure 8.1 for a single frame example. In this example the recovered albedo is not perfect, as the fit was inaccurate in the fine details such as the nasiolabial folds. Nonetheless, after relighting the shadows and occlusions are convicing, because in the difficult areas the source and target illumnation are the same.

Nevertheless, it is still possible to distinguish the virtual and real objects. One strong cue is that the boundary of the rendered object is too crisp. This happens because in a real camera a pixel is formed by integrating the incoming photons within a small space angle, such that a boundary pixel never contains the reflection of only a single object. In the rendered image this integration is replaced by sampling a ray at the pixel center. We overcome this problem by



(a) (b)

Figure 8.2: The combined image looks unnatural at the boundary between virtual and real objects because in the real scene a pixel is actually the integral over the incoming light within an area, while the computer generated image samples only a ray at the pixel center. The effect is that natural boundaries appear smoother than the computer generated boundaries. This is remedied by averaging the pixel colour of boundary pixels with the colour of surrounding non-rendered pixels. The figure shows a closeup of (a) the boundary before and (b) after selective smoothing.

averaging every boundary pixel just inside the rendered object with the neighbouring non-rendered pixels. A closeup view of the result is shown in figure 8.2. The remaining differences could be removed by choosing appropriate material properties and textures for the rendered object.

Handling the light by rendering up to 800 lights is quite expensive but results in a convincing lighting. Figure 8.3 shows a closeup of a frame where there is still a smooth shadowing visible where the glasses are close to the skin, even though the lighting is ambient, that is coming from nearly the full hemisphere. This is the expected effect as in this area a large fraction of the visible hemisphere is occluded by the glasses. These result could not have been simulated with a single light source. Some frames from a longer video sequence are shown in figure 8.4, the full video is included in the complementary material.

2 Face Exchange

Instead of just adding synthetic objects into the scene, we can also manipulate the face itself. The most straightforward application is to exchange the face of the person in the target video with that of the person in the source video. We do this by exchanging the identity and expression coefficients of the target video with those recovered from the source video. The lighting in the source video is then inverted, the albedo is transferred into the target video and relighted with the new light situation. This approach agrees with that proposed by Pierrard (2008), where it was suggested for the manipulation of single images, and allows

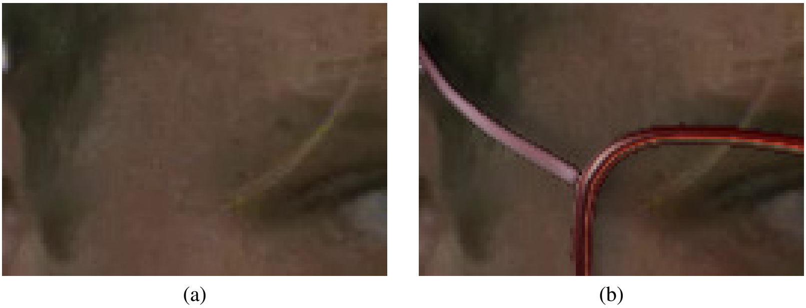

(a) (b)

Figure 8.3: Rendering the frames with 800 light sources results in a correct handling of ambient occlusion. Even though the lighting in the closeup shown in (a) is nearly ambient, that is coming from the full hemisphere, we model a shadowing in the area below the glasses (b), where the visible part of the hemisphere is smaller due to occlusion by the glasses. This could not have been achieved with a single light source.

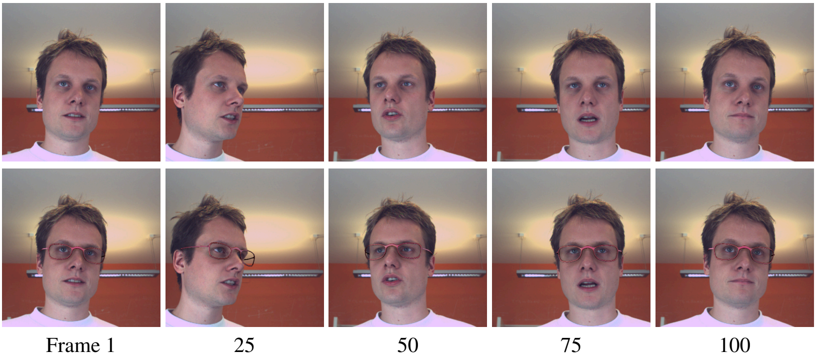


Figure 8.4: A few frames from a video showing the insertion of glasses into a 100 frame sequence. Observe the correct light interaction between face and glasses.

us to exchange faces across different pose, camera and light configurations. The main difference is that we average the albedo over multiple frames to get a more consistent result and to fill in those parts of the face which are invisible in the current source frame but become visible after face editing.

A face exchange is much more complicated than the task of adding a new object into the scene. Adding a new object only added shadows on the face, while we now have to guess and fill in the appearance of parts of the scene which were invisible in the original video and now become visible. This happens in two places. First, parts of the background might no longer be occluded by the face when its shape changes. In Blanz et al. (2003) this was handled by mirroring the originally visible image into the newly visible part. We are adopting a different approach and slightly deform the original image such that the outline of the face in the deformed image is equal to the newly rendered outline. This has the advantage that slight imperfections in the model fit are less visible. The second situation where invisible parts of the scene have to be filled in is when due to pose changes parts of the face become visible which were invisible in the original image. We fill these in with the model texture, which itself was adjusted to the full video, and therefore is correct even for parts of the face which are only visible in other frames of the source video. An example of a face exchange between videos is shown in figure 8.5 and in the accompanying movies.

3 Expression Transfer

A face editing task which can be only achieved with a generative face model, is the transfer of expressions between videos. Recall that in chapter 2 we separated expression and identity components of the model, by additively combining an identity model with an expression model. Having separate model parameters for expressions and identity we can now generate new expressions by exchanging the expression parameters in a fitted sequence with the expression parameters from another sequence. This is independent from the pose and identity of the steering sequence, allowing us to drive the expressions of one actor with the expressions of another actor. This application is useful for speech editing, especially for lip-syncing. An example of this application is shown in figure 8.6 and in the accompanying video material.

4 Identity Transfer

By selectively transferring the identity component we can keep the original expressions but exchange the identity, which allows special effects in movies where the identity changes smoothly between two individuals. An example of this effect

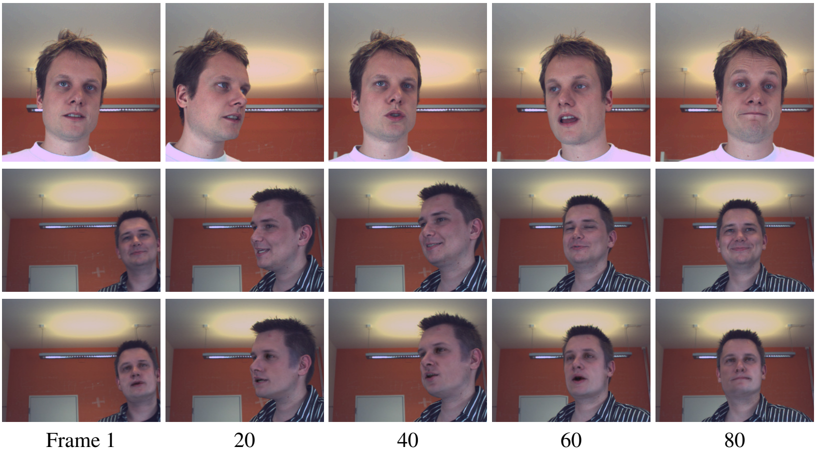


Figure 8.5: Face exchange by copying the identity and expression components as well as the albedo from one face onto another, and relighting the resulting scene. The first row shows the input face, the second row shows the target video, and in the third row the newly rendered video with transferred identity and expression is shown. For more details refer to the complementary material.

is shown in figure 8.8 and in the accompanying movie material.

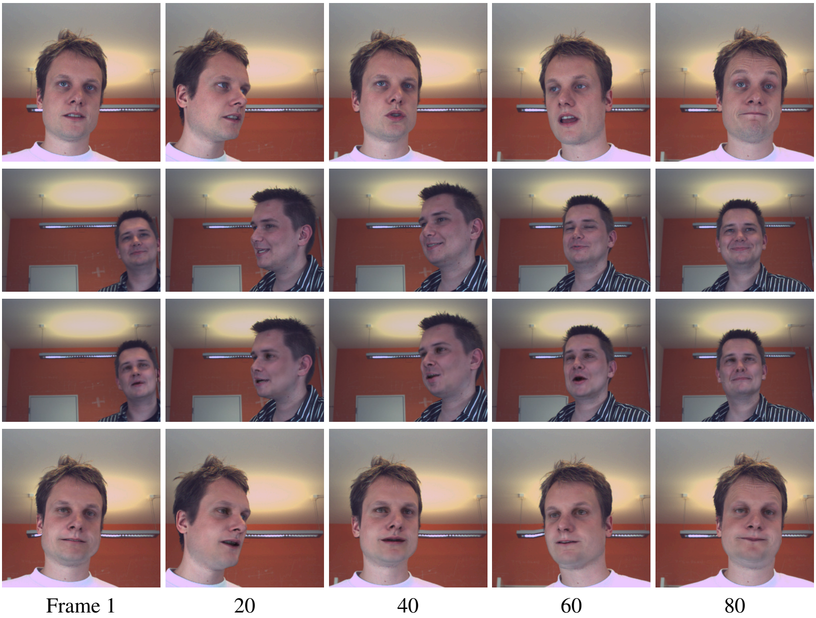
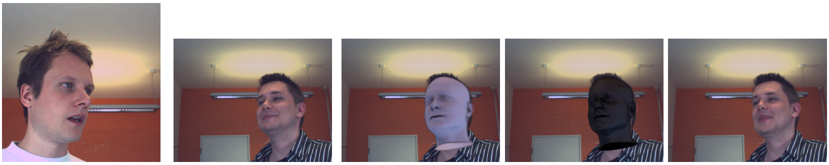
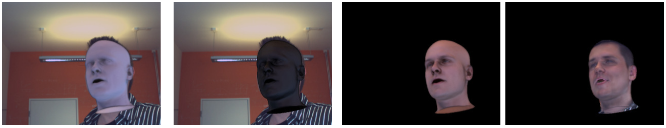


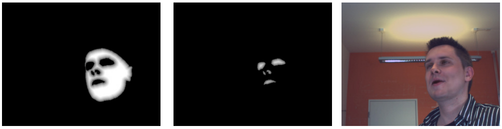
Figure 8.6: Expression transfer by copying the expression components from one video onto another and relighting the resulting scene. The first row shows the input expression, the second row contains the input video and in the third row the newly rendered video with transferred expression is shown. In the fourth row the roles of the input videos have been switched. For more details refer to the complementary material.



(a) Input Expression (b) Input Frame (c) Input Lighting (diffuse) (d) Input Lighting (specular) (e) Warped Input Frame

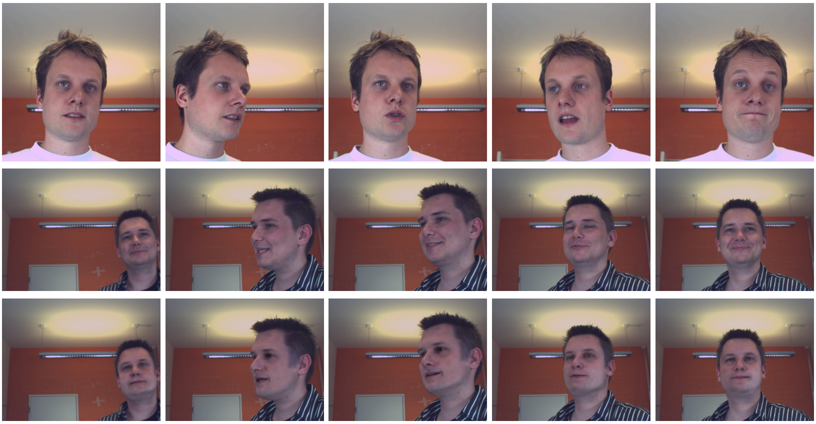


(f) Output Lighting (diffuse) (g) Output Lighting (specular) (h) Lighted Model (i) Lighted Extracted Texture



(j) Texture Mask (k) Model Mask (i) Output Frame

Figure 8.7: The intermediate steps of an expression transfer. The masks are defined in model space and the images are linearly blended.



Frame 1 20 40 60 80

Figure 8.8: Identity transfer by copying the identity components and albedo from one video onto another and relighting the resulting scene. The first row shows the input expression, the second row contains the input video and in the third row the newly rendered video with transferred identity and maintained expression is shown. For more details refer to the complementary material.

— Chapter 9 —

Conclusion and Future Work

We have presented a complete system for video editing, and demonstrated its efficacy on a number of example sequences. We explained each step necessary, and kept the parts as separate as possible, such that the different innovations can be reused in other contexts or improved independently in future work. Let us now recall the novelties introduced in this thesis.

We started with the registration of 3D face scans, which are used to build the generative face model. For registration, we proposed a new deformation cost, which was compared to other deformation energies from the mesh editing community and evaluated on over 1200 face scans.

Next, we described a fitting algorithm, which takes the video as a whole into account, instead of fitting each frame separately.

Fitting a morphable model needs some initial landmarks. To efficiently find these in the video we introduced a semi-automatic interest point tracking method that is used to initialize the fitter.

Even with this initialization, it is still necessary to guide the fitting towards the right minimum within the complex energy landscape. This was achieved by detecting the outlines of eyes, lips, and, nose with the help of part and viewpoint specific Active Appearance Models. To reliably fit these AAMs it was necessary to develop a new AAM fitter, which is much more robust than previous methods while still being efficient.

To further improve the fitting, we proposed to explain the complete data, that is all of the video, and not only the part which is estimated to be the face as was done previously. To this end we introduced an algorithm for the simultaneous segmentation and fitting of images.

Finally, we gave some examples of face editing applications made possible by the model fits. This includes the insertion of virtual objects which interact with the face, and the transfer of expression and/or identity from one video onto another.

Possible extensions to this work lie in two areas. The first area is that of extending the model. There are three levels of detail in faces. The level described nicely by a 200 dimensional linear shape model is that of the overall shape of faces. With 200 parameters we are able to reproduce most faces well enough to make them recognizable and to explain most of the apperance. At the next level lies the accurate shape of the face, which includes the exact position and shape of wrinkles such as the crows feet, nasiolabil folds or glabellar wrinkles. This part of the shape can not be described by linear combinations of tesselated meshes, as the structures differ in topology. The number number of crow feets and minor nasiolabial folds, for example, is different for each person.

It will therefore be necessary to either introduce a different parametrization to build a generative model of wrinkles at this level, as proposed by Paysan (2010), or to apply shape from shading to refine the shape estimated with the 3D-MM.

The third level of detail is that of pores and skin cell boundaries. While the face shape and wrinkles can still be described well with a high resolution tesselated mesh, it becomes infeasible to describe the face at this level with such a parametrization. Here it will be necessary to develop a model describing the structure of the skin, i.e. the kind of pores and the density of the pores but not the exact position of every pore and skin cell.

Such a multi-layer model with many parameters will open up new questions because for example the transfer of details between faces will not be straightforward, and fitting such a model is a difficult task.

While the previous extensions to the model concerned the spatial accuracy, it will also be of interest to develop a model of the temporal dynamics of speech and expressions, which can be used to further constrain the fitting, and to generate completely new speech and expressions without a driving actor.

The second area where extensions are possible is that of the applications. The inclusion of a teeth, eye and hair model will improve the animations on the graphics side. Also, the handling of the background is not solved to our satisfaction in this pipeline. From a user interface perspective it would be interesting to extend the attribute editing method presented in Amberg et al. (2009b) to videos such that the animator can change attributes such as ‘sadness’ or ‘lack of sleep’ in a video.

So we believe that if the face editing pipeline presented in this thesis would be implemented in a commercial environment, where more resources could be spend on the artistic refinement, then it will be directly applicable to commercial video postproduction. The research community on the other hand will benefit from the face model, and the dynamics data which we can extract with our model, and also from the registration method, appearance model fitter, interest point tracking method and video fitter presented in this work.

Bibliography

Agarwala, A., Hertzmann, A., Salesin, D. H., and Seitz, S. M. (2004). Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH '04: ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, volume 23, issue 3, pages 584–591. ACM. [51]

Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, pages 587–594. ACM. [26, 28]

Amberg, B., Blake, A., Fitzgibbon, A., Romdhani, S., and Vetter, T. (2007a). Reconstructing high quality face-surfaces using model based stereo. In *ICCV '07: International Conference on Computer Vision.* [39, 46, 69, 70]

Amberg, B., Blake, A., and Vetter, T. (2009a). On compositional image alignment, with an application to active appearance models. In *CVPR '09: Computer Vision and Pattern Recognition*, volume 0, pages 1714–1721. IEEE Computer Society. [46, 80, 86, 89, 90]

Amberg, B., Knothe, R., and Vetter, T. (2008a). Expression invariant 3D face recognition with a morphable model. In *FG '08: 8th Automatic Face & Gesture Recognition*, pages 1–6. IEEE. [8, 9, 10, 30, 39]

Amberg, B., Knothe, R., and Vetter, T. (2008b). SHREC'08 entry: Shape based face recognition with a morphable model. In *SMI '08: Shape Modeling and Applications*, pages 253–254. [39]

Amberg, B., Paysan, P., and Vetter, T. (2009b). Weight, sex, and facial expressions: On the manipulation of attributes in generative 3D face models. In

Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Wang, J.-X., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnação, M. L., Silva, C. T., and Coming, D., editors, *ISVC '09: 5th International Symposium on Visual Computing, Advances in Visual Computing*, volume 5875 of *Lecture Notes in Computer Science*, pages 875–885. Springer. [104]

Amberg, B., Romdhani, S., and Vetter, T. (2007b). Optimal step nonrigid ICP algorithms for surface registration. In *CVPR '07: Computer Vision and Pattern Recognition*. IEEE Computer Society. [12, 27, 28]

Anderson, R. and Schweitzer, H. (2009). Fixed time template matching. In *SMC 2009, Systems, Man and Cybernetics*, pages 1359 –1364. [60]

Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *CVPR '01: Computer Vision and Pattern Recognition*, volume 1, pages I–1090–I–1097. IEEE Computer Society. [80]

Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *IJCV: International Journal of Computer Vision*, 56(3):221–255. [80, 86]

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256. [27, 28]

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer. [8]

Blanz, V., Basso, C., Vetter, T., and Poggio, T. (2003). Reanimating faces in images and video. In Brunet, P. and Fellner, D. W., editors, *EUROGRAPHICS 2003: European Association for Computer Graphics*, volume 22, issue 3 of *Computer Graphics Forum*, pages 641–650. The Eurographics Association, Blackwell. [1, 2, 3, 8, 10, 99]

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99: Computer Graphics and Interactive Techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co. [1, 2, 7, 8, 26, 39, 47, 80]

Botsch, M., Pauly, M., Gross, M., and Kobbelt, L. (2006a). Primo: coupled prisms for intuitive surface modeling. In *SGP '06: Eurographics symposium on Geometry processing*, pages 11–20. Eurographics Association. [20]

Botsch, M. and Sorkine, O. (2008). On linear variational surface deformation methods. *VCG: IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230. [13, 20, 21, 22]

Botsch, M., Sumner, R., Pauly, M., and Gross, M. (2006b). Deformation transfer for detail-preserving surface editing. In *VMV '06: Vision, Modeling & Visualization*, pages 357–364. [13]

Brand, M. (2001). Morphable 3d models from video. In *CVPR '01: Computer Vision and Pattern Recognition*, volume 2, pages 456+. IEEE Computer Society. [46]

Bregler, C., Hertzmann, A., and Biermann, H. (2000). Recovering Non-Rigid 3D Shape from Image Streams. In *CVPR '00: Computer Vision and Pattern Recognition*, volume 2, pages 2690–696 vol.2. IEEE Computer Society. [46]

Buchanan, A. and Fitzgibbon, A. (2006). Interactive Feature Tracking using K-D Trees and Dynamic Programming. In *CVPR '06: Computer Vision and Pattern Recognition*, pages 626–633. IEEE Computer Society. [2, 51, 52, 58, 60, 62, 63, 66]

Burkhardt, H. and Diehl, N. (1986). Simultaneous Estimation of Rotation and Translation in Image Sequences. In *EUSIPCO-86: European Signal Processing Conference*. [80]

Byun, H. W. (2007). Realistic facial modeling and animation based on high resolution capture. In *ACIVS'07: Proceedings of the 9th international conference on Advanced concepts for intelligent vision systems*, pages 417–426. Springer-Verlag. [3]

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619. [70, 73]

Cootes, T. F. (2008). Talking face video. `http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html`, retrieved on 20th October 2010. [90]

Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active Appearance Models. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685. [79, 80]

Delamarre, Q. (2001). 3D articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328–357. [46]

Di Stefano, L. and Mattoccia, S. (2003). Fast template matching using bounded partial correlation. *Machine Vision and Applications*, 13:213–221. [60]

Diehl, N. (1988). *Methoden zur allgemeinen Bewegungsschätzung in Bildfolgen*. PhD thesis, TU Hamburg-Harburg. Published as Fortschrittsbericht (Reihe 10, Nr. 92) VDI-Zeitschriften, VDI-Verlag. [80]

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271. [55, 56]

Feldmar, J. and Ayache, N. (1996). Rigid, affine and locally affine registration of free-form surfaces. *IJCV: International Journal of Computer Vision*, 18(2):99–119. [28]

Fitzgibbon, A. W. (2001). Robust registration of 2d and 3d point sets. In *British Machine Vision Conference*. [26]

Fredman, M. L. and Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *JACM: Journal of the ACM*, 34(3):596–615. [56]

Hiwada, K., Maki, A., and Nakashima, A. (2003). Mimicking video: real-time morphable 3d model fitting. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 132–139. ACM. [48]

Hotelling, H. (1933). Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441. [8]

Huang, X., Zhang, S., Wang, Y., Metaxas, D., and Samaras, D. (2004). A hierarchical framework for high resolution facial expression tracking. In *CVPRW '04: Proceedings of the 2004 CVPR Workshop*, pages 22+. IEEE Computer Society. [3]

Ilić, S., Salzmann, M., and Fua, P. (2006). Implicit meshes for effective silhouette handling. *IJCV: International Journal of Computer Vision*. [46]

Intel (2010). *Intel® 64 and IA-32 Architectures, Software Developer's Manual: Instruction Set Reference, N-Z*, volume 2B. [62]

Kawanishi, T., Kurozumi, T., Kashino, K., and Takagi, S. (2004). A fast template matching algorithm with adaptive skipping using inner-subtemplates' distances. *Int. Conf. on Pattern Recognition*, 3:654–657. [60]

Keller, M., Knothe, R., and Vetter, T. (2007,). 3D reconstruction of human faces from occluding contours. *Proceedings of the Mirage*. [46]

Knothe, R. (2009). *A global-to-local model for the representation of human faces*. PhD thesis, University of Basel, Faculty of Science. [2, 69, 70]

Krishnamurthy, V. and Levoy, M. (1996). Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 313–324. ACM. [23]

Liu, C. (2009). *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology. [30]

Liu, X. (2007). Generic Face Alignment using Boosted Appearance Model. In *CVPR '07: Computer Vision and Pattern Recognition*. [80]

Lüthi, M., Lerch, A., Albrecht, T., Krol, Z., and Vetter, T. (2008). A hierarchical, multi-resolution approach for model-based skull-segmentation in mri volumes. In *Conference Proceedings 3D-Physiological Human*. [39]

Matthews, I. and Baker, S. (2004). Active Appearance Models Revisited. *IJCV: International Journal of Computer Vision*, 60(2):135–164. [79, 80, 82]

Matthews, L., Ishikawa, T., and Baker, S. (2004). The template update problem. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810 –815. [52]

Messer, K., Matas, J., Kittler, J., Luettin, J., and Maitre, G. (1999). XM2VTSDB: The Extended M2VTS Database. In *2nd Int. Conf. on Audio and Video-based Biometric Person Authentication*. [89]

Muñoz, E., Buenaposada, J. M., and Baumela, L. (2009). A direct approach for efficiently tracking with 3d morphable models. In *ICCV '09: International Conference on Computer Vision*. [48]

Nordstrøm, M. M., Larsen, M., Sierakowski, J., and Stegmann, M. B. (2004). The IMM face database - an annotated dataset of 240 face images. Technical report, IMM, TU Denmark DTU. [89]

Paysan, P. (2010). *Statistical Modeling of Facial Aging based on 3D Scans*. PhD thesis, University of Basel, Faculty of Science. [104]

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317. [47]

Pierrard, J.-S. (2008). *Skin segmentation for robust face image analysis*. PhD thesis, University of Basel, Faculty of Science. [97]

Plaenkers, R. and Fua, P. (2002). Model-based silhouette extraction for accurate people tracking. In *ECCV '02: European Conference on Computer Vision*, pages 325–339. [46]

Romdhani., S. and Vetter, T. (2005). Estimating 3D shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *CVPR '05: Computer Vision and Pattern Recognition*, volume 2, pages 986–993 vol. 2. [2, 46, 69, 70]

Saragih, J. and Goecke, R. (2007). A Nonlinear Discriminative Approach to AAM Fitting. In *ICCV '07: International Conference on Computer Vision*. [80]

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV: International Journal of Computer Vision*, 47(1-3):7–42. [46]

Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535. [62]

Schweitzer, H., Bell, J., and Wu, F. (2006). Very fast template matching. In *ECCV '02: European Conference on Computer Vision*, volume 2353 of *LNCS*, pages 145–148. Springer. [60]

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR '06: Computer Vision and Pattern Recognition*, volume 1, pages 519–528. IEEE Computer Society. [46]

Stoiber, N., Seguier, R., and Breton, G. (2010). Facial animation retargeting and control based on a human appearance space. *Computer Animation and Virtual Worlds*, 21(1):39–54. [3]

Sun, J., Zhang, W., Tang, X., and Shum, H.-Y. (2005). Bi-directional tracking using trajectory segment analysis. In *ICCV '05: International Conference on Computer Vision*, pages 717–724. IEEE Computer Society. [51]

Torresani, L., Hertzmann, A., and Bregler, C. (2003). Learning non-rigid 3d shape from 2d motion. In *NIPS*. [46]

Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM. [23]

Vemuri, B. C., Huang, S., Sahni, S., Leonard, C. M., Mohr, C., Gilmore, R., and Fitzsimmons, J. (1998). An efficient motion estimator with application to medical image registration. *Medical Image Analysis*, pages 79–98. [80]

Vlasic, D., Brand, M., Pfister, H., and Popović, J. (2005). Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3):426–433. [3]

Vogiatzis, G., Hernandez, C., and Cipolla, R. (2006). Reconstruction in the round using photometric normals and silhouettes. In *CVPR '06: Computer Vision and Pattern Recognition*, pages 1847–1854. IEEE Computer Society. [46]

Wang, Y., Gupta, M., Zhang, S., Wang, S., Gu, X., Samaras, D., and Huang, P. (2008). High resolution tracking of non-rigid motion of densely sampled 3D data using harmonic maps. *IJCV: International Journal of Computer Vision*, 76(3):283–300. [3]

Wang, Y., Huang, X., Lee, C.-S., Zhang, S., Li, Z., Samaras, D., Metaxas, D., Elgammal, A., and Huang, P. (2004). High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum*, pages 677–686. [3]

Wei, Y., Sun, J., Tang, X., and Shum, H. Y. (2007). Interactive offline tracking for color objects. In *ICCV '07: International Conference on Computer Vision*. IEEE Computer Society. [51]

Wesseling, P. (1992). *An introduction to multigrid methods*. Pure and applied mathematics. J. Wiley. [18]

Wicke, M., Botsch, M., and Gross, M. (2007). A finite element method on convex polyhedra. *Computer Graphics Forum*, 26(3):355–364. [13]

Wimmer, M., Stulp, F., Pietzsch, S., and Radig, B. (2008). Learning local objective functions for robust face model fitting. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1357–1370. [80]

Wu, H., Liu, X., and Doretto, G. (2008). Face Alignment via Boosted Ranking Model. In *CVPR '08: Computer Vision and Pattern Recognition*. [80]

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions On Mathematical Software*, 23(4):550–560. [48]

Curriculum Vitae
Brian Amberg

Education

2005–2009	PhD Student in the **Computer Vision Group at the University of Basel**.
1999–2005	Diplom (Master) in Computer Science from the **University of Freiburg**, with a minor in cognitive science.
1989–1998	Highschool: **Kurt Tucholsky Schule Flensburg-Adelby**.

Publications

2011	**Book-Chapter: Morphable Models of Faces**, Handbook of Face Recognition 2nd Edition, Reinhard Knothe, Brian Amberg, Sami Romdhani, Volker Blanz and Thomas Vetter. Editors Stan Z. Li, Anil K. Jain, Springer
Nov. 2011	**Optimal Landmark Detection using Shape Models and Branch and Bound**, ICCV'11, Brian Amberg and Thomas Vetter
June 2011	**GraphTrack: Faster than realtime tracking in videos**, CVPR'11, Brian Amberg and Thomas Vetter
Nov. 2009	**Weight, Sex, and Facial Expressions: On the Manipulation of Attributes in Generative 3D Face Models**, ISVC'09, 5th International Symposium on Visual Computing, Brian Amberg, Pascal Paysan and Thomas Vetter

Sept. 2009 **Face Reconstruction from Skull Shapes and Physical Attributes**, DAGM'09 Pascal Paysan, Marcel Lthi, Thomas Albrecht, Anita Lerch, Brian Amberg, Francesco Santini, and Thomas Vetter
(DAGM Best Paper Award)

Sept. 2009 **A 3D Face Model for Pose and Illumination Invariant Face Recognition**, AVSS'09 (IEEE Int. Conf. on Advanced Video and Signal based Surveillance for Security, Safety and Monitoring in Smart Environments), Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani and Thomas Vetter

June 2009 **On Compositional Image Alignment with an Application to Active Appearance Models**, CVPR'09 Brian Amberg, Andrew Blake and Thomas Vetter

Sep. 2008 **Expression Invariant Face Recognition with a Morphable Model**, FG'08, IEEE Int. Conf. on Automatic Face and Gesture Recognition, Brian Amberg, Reinhard Knothe and Thomas Vetter

June 2008 **SHREC'08 Entry: Shape Based Face Recognition with a Morphable Model**, SMI'08, Shape Modeling International Brian Amberg, Reinhard Knothe and Thomas Vetter

Oct. 2007 **Reconstructing High Quality Face-Surfaces using Model Based Stere**, ICCV'07, Brian Amberg, Sami Romdhani, Andrew Fitzgibbon, Andrew Blake, Thomas Vetter

June 2006 **Optimal Step Nonrigid ICP Algorithms for Surface Registration**, CVPR'07, Brian Amberg, Sami Romdhani, Thomas Vetter

Apr. 2004 Diploma Thesis **A 3D Optical Flow Algorithms**, Brian Amberg

Work Experience

June – Aug. 2009 Research internship at **Microsoft Research Cambridge** in the Machine Learning and Computer Vision Lab working on feature point detection with random forests and a morphable model.

Apr. – July 2006 Research internship at **Microsoft Research Cambridge** in the Machine Learning and Computer Vision Lab working on

	model based stereo algorithms.
2001 – 2005	Software Developer at **Steinhart Medical Systems**
2000 – 2009	TA and teaching in computer science courses at **University of Freiburg** and **University of Basel**.
1994 – 2004	Founder of the software development company **BB-Soft** (1994-1998) **/ BS Software** (1998-2004).
1995-2002	Various Internships and jobs, in banking, publishing, electronics and as a call agent.

Projects

since 1992	I did a lot of programming in various languages, for money and for free.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Switzerland License. To view a copy of this license, visit `http://creativecommons.org/licenses/by-nc-nd/2.5/ch/` or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

- **Attribution** – You must give the original author credit.
- **Non-Commercial** – You may not use this work for commercial purposes.
- **No Derivative Works** – You may not alter, transform, or build upon this work.

With the understanding that:

Waiver – Any of the above conditions can be waived if you get permission from the copyright holder.

Public Domain – Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights – In no way are any of the following rights affected by the license: Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations; The author's moral rights; Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice – For any reuse or distribution, you must make clear to others the licence terms of this work.

Disclaimer

This is a human-readable summary of the Legal Code (the full licence). You can get the full licence by visiting `http://creativecommons.org/licenses/by-nc-nd/2.5/ch/`.