Severin Holzer-Graf, BSc

3D Morphable Model Fitting for Faces

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Horst Bischof

Institute for Computer Graphics and Vision

Graz, December 2014

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis dissertation.

Date

Signature

Abstract

Faces are the most distinctive feature of human beings. Accurate and efficient face modeling has always been an active research area in computer vision. Applications range from facial recognition over tracking to photo-realistic modeling.

Real world data is often noisy and incomplete which can be better handled by employing prior knowledge from a model. We employ 3D Morphable Models (3DMMs) to model faces and present a robust and extendable pipeline for fitting 3DMMs for facial scans.

3DMMs are well known statistical models in computer vision and constitute an efficient facial representation. We present the basic concepts of 3DMMs to aid the reader's understanding of the topic.

An existing cost function to fit 3DMMs is extended by point-to-plane constraints. The existing cost function acquires correspondences for fitting by means of a non-rigid registration and does not incorporate possible better correspondence. We tackle this issue and propose two methods that incorporate updated correspondences. Furthermore we adapt an existing non-rigid 3D registration method to provide reliable correspondences for the fitting cost functions. The proposed methods alternately fit a 3DMM and register the input sample to the previously 3DMM, allowing the usage of better correspondences. The algorithms behave like Iterative Closest Point (ICP) and iteratively converge to a local minimum.

We collect a dataset of 10 persons using a handheld 3D scanner to acquire real world data and evaluate the proposed methods with it. Our methods achieve superior results in comparison to the original method and are faster.

Zusammenfassung

Das Gesicht ist das markanteste Erkennungsmerkmal eines Menschen. Die dreidimensionale Computermodellierung eines Gesichts ist seit jeher ein aktives Forschungsgebiet in der maschinellen Bildverarbeitung. Anwendungsgebiete erstrecken sich von Gesichtserkennung über Nachverfolgung bis hin zu fotorealistischer Modellierung.

Aufgenommene Datensätze sind oft verrauscht und unvollständig, was eine Verarbeitung schwierig macht. Eine Methode, um diese Probleme zu bewältigen, ist der Einsatz eines statischen Modells als a priori Wissen. In dieser Arbeit verwenden wir ein 3D Morphable Model um Gesichter zu modellieren und präsentieren eine robuste und einfach erweiterbare Pipeline zur Anpassung von 3D Morphable Models an 3D Scans von Gesichtern.

3D Morphable Models sind wohlbekannte statistische Modelle in der maschinellen Bildverarbeitung und ermöglichen eine kompakte Gesichtsrepräsentation. Wir erläutern das grundlegende Konzept von 3D Morphable Models in dieser Arbeit und zeigen verschiedene Methoden um die Modellparameter für 3D Scans zu finden.

Wir erweitern eine existierende Kostenfunktion zur Anpassung von 3D Morphable Models mit einem Punkt-zu-Fläche Fehlerterm. Die ursprüngliche Kostenfunktion gewinnt Punkt-zu-Punkt Korrespondenzen durch eine einmalige nicht rigide Registrierung. Wir erweitern diese Methode um aktualisierte und möglicherweise bessere Punkt-zu-Punkt Korrespondenzen zu erhalten. Unsere Methode führt abwechselnd eine Modellanpassung und Korrespondenzsuche zum Modell der vorherigen Iteration durch. Der Algorithmus verhält sich vergleichbar zu dem Iterative Closest Point (ICP) Algorithmus.

Um diese Arbeit zu evaluieren, erstellen wir einen Datensatz von 10 Gesichtsscans mit einem Handscanner. Unsere entwickelten Methoden erreichen bessere Ergebnisse als die ursprüngliche Methode.

Acknowledgements

This work would not have been possible without the substantial support of several people in their very specific ways.

First thanks go to my parents Gottfried and Henrike for supporting me all the way that led me to Sweden and enabling me to conduct my studies. Thanks to the Volumental team for providing me the possibility to do my master thesis in an awesome swedish startup: Miro, for offering me to do this thesis and supervising me. Rasmus and Magnus for the guidance and inspiring discussions. All the other team members for many interesting and motivating discussions. I also owe thanks to Björn Möller from KTH Maskinkonstruktion for providing me access to the 3D scanning lab and avoiding unnecessary bureaucracy. Special thanks go to Horst Bischof who opened the possibility to write this thesis abroad and supervised me from Austria.

Last, but not least, many thanks to my friends in Sweden and Austria for keeping me in a good mood by travelling, partying and letting me enjoy their company.

Contents

1. Introduction

Computer Vision often has to deal with objects that belong to the same category (*e.g.* cars, buildings), but differ a lot in shape and texture. Human beings can easily detect and distinguish between different categories of objects and different groups within a category. On the other hand, it is difficult for computers to recognize, detect and classify object categories, and to differentiate objects within them. The amount of different categories is basically unlimited. Human beings learn a new category by only a few samples and intuitively learn a model for a specific category. We know the typical properties and possible variants, and learning the typical shape and/or texture and possible variants for an object is the idea that statistical shape and appearance models are based on.

This work deals with the "category" faces which is one of the strongest features to distinguish between human beings. Automatic processing of facial images has become very important for biometry, recognition, surveillance, facial animation and many other application areas. Many different models like Active Appearance Models (AAMs), Active Shape Models (ASMs) have been developed and heavily used to deal with faces in many different applications. AAMs and ASMs model 2D shape and appearance, in this work we focus on 3D shape, which offers the 3D Morphable Model (3DMM) to be used. The 3DMM is a well-known tool for 3D face reconstruction from either 2D data or 3D data. It models the shape and appearance of a 3D object. It was successfully used for many tasks concerning faces Prominent applications are:

- synthesizing 2D face images from different views [10, 45],
- facial animation [55]
- tracking [45, 55],
- face recognition [2] and
- avatar generation [59].

1.1. Problem Definition

All these very promising applications come with the challenge of fitting a 3DMM. Fitting means finding the model parameters that make the fitted model as similar as possible to the input data. This very imprecise definition of fitting will be refined later in Section 2.1. This task is quite difficult due to complexity of the input data and high variability of faces. For example an algorithm that fits a facial 3DMM to a photograph, has to be able to recover the missing 3D information. Algorithms that fit 3DMMs to 3D data have to

deal with different head poses, light conditions, noise, *etc.*. The high variability of faces makes this task difficult even without the previously mentioned challenges.

Since cheap consumer depth cameras are available, fitting to 3D depth data has become popular. Depth cameras are cameras that give, in addition to color values, the distance from the camera to every pixel in world space. Leveraging some of the difficulties of fitting a 3DMM to 2D data, a 3D based fitting algorithm still has to deal with a variety of problems. Cheap cameras come with the disadvantage of often incomplete and noisy depth data. Small, yet very distinctive details like the nose tip, or even whole parts like ears might be missing.

Nevertheless, the fitting algorithm has to be robust to that and accurately fit the model. Robust fitting of 3DMMs to 3D data is the challenge we want to approach in this work.

1.2. Thesis Structure

This thesis is structured as follows: Chapter 1 motivates and defines the problem this work is dedicated to. A brief historical overview about the research that led to the 3DMM gives the reader insights into its appearance. The related work in Section 1.3 part informs the reader about different fitting methods and developments that were most relevant recently.

Chapter 2 presents the structure of a 3DMM. Model training and certain fitting methods require dense point-to-point correspondence and therefore also contain a non-rigid registration scheme. 3DMMs possess very desirable properties that are usable as a fitting prior. We show how to utilize these properties and also elaborate on the expressive power of the used model.

The core part of this work is presented in Chapter 3. We present three different fitting methods. The first method extends an existing method. The second method iteratively acquires better model fits by finding a better correspondence set. The third method uses a more sophisticated pose estimation method and solves several linear subproblems.

All methods are evaluated in Chapter 4 and discussed in detail. Chapter 5 concludes the thesis with a final discussion of learned lessons and future work.

1.3. Related Work

In Section 1.3.1 we give a historical overview of the 3DMM and briefly refer to competing models. We continue with different fitting strategies for 3DMM that are the actual topic of this thesis. We structure them into two groups: Fitting to pure 2D data and fitting to 3D data. Section 1.3.2 starts with image-based model fitting and Section 1.3.3 continues with methods that incorporate 3D data.

1.3.1. Appearance and Current Trends of Statistical Shape Models

The underlying idea of statistical shape models is that unseen samples of the model's category can be expressed as weighted linear combinations of the model's training samples. A new face could for instance be

$$\text{face}_{\text{unseen}} = 0.1 \cdot \text{face}_{\text{a}} + 0.2 \cdot \text{face}_{\text{b}} + 0.7 \cdot \text{face}_{\text{c}} \tag{1.1}$$

Such a very basic model has many disadvantages. First the dimensionality of the model is very high. Second, the number of samples in the model has to be very high to be able to cover enough variance to gain reasonable expressive power. The need to have many samples in the model can result in high redundancy between the samples.

Kirby et al. [29] tackled the problems of the native approach by applying Principal Component Analysis (PCA) to the vectorized training images. PCA is a classical statistical procedure that gives a more compact data representation by projecting the original high dimensional data into a lower dimensional coordinate space. The so called "Eigenfaces" is another application of PCA for faces. "Eigenfaces" are used for face representation, detection and recognition [54].

The previously mentioned approaches neglect model specificity for gaining high variability. This impacts the robustness during interpretation of images [18]. Cootes et al. [18] incorporate category specific knowledge into his ASM by generating a point distribution model. It learns the variability pattern of annotated feature data points. ASMs are also PCA-based models.

The AAM is the main result evolving from previous models. It is also strongly related to the 3DMM that can be seen as a three dimensional variant of the AAM [10]. ASMs have the shortcoming of not modeling and incorporating both texture and shape. AAMs model the shape using feature points and the texture using texture patches at the underlying feature points. They have been successfully applied to many different computer vision tasks, among them medical imaging [7, 36] and face recognition [19].

In 1999 Blanz et al. [10] proposed the 3DMM. It was originally developed for modeling textured 3D faces [10] and is like the AAM a statistical shape and appearance model. This inherently means that a broad range of realistic faces can be modeled by a few coefficients. The main difference is a much denser model of corresponding points and the extension to 3D.

Since the emergence of 3DMMs, developing robust fitting methods to different kinds of inputs has been an ongoing research topic. Early fitting methods reconstructed the model parameters from photographs. Fitting to photographs is inherently an ill-posed problem, because all 3D information is lost and has to be recovered by various techniques. Fitting to 3D data has become more popular since cheap consumer depth cameras have become available. Earlier, acquiring 3D data was more difficult and was mostly done using laserscanners [40]. Among the most prominent depth cameras are Microsoft Kinect and Primesense. The first generations of depth cameras were quite large external devices.

Recently Intel announced their RealSense depth cameras that are small enough to be integrated in notebooks and tablets. This opens a whole new range of applications.

Wide-spread usage of 3DMMs in the scientific community was also detained by the complexity of model generation. To tackle this issue the Basel Face Model (BFM) was created in 2009 and made available to the scientific community [40]. This model is considered to be the state of the art 3DMM [37] and is used in many recent publications [1, 37, 55].

The previous paragraphs inform the reader about the appearance and the evolution that led to the 3DMM. Following we want to state related work concerning existing fitting methods. We focus on image-based fitting methods in Section 1.3.2 and on point cloud-based methods in for image-based and point cloud-based fitting methods in Section 1.3.3.

1.3.2. Image-based Model Fitting for 2D

"Analysis by synthesis" denotes fitting methods that synthesize the model for the current parametrization and use it to further improve the fitting. Or more precisely, denotes fitting a parameter update, calculated by a model synthetization of the current parameters.

Fitting to pure 2D data is inherently an ill-posed problem. By projecting from 3D to 2D, all depth data is lost and has to be recovered by analysis by synthesis techniques.

The first 3DMM is that of Blanz et al. [10] and introduced the term "morphable head model". They use Stochastic Newton Optimization (SNO) and analysis by synthesis to fit the 3DMM to a 2D photograph of a face.

Optimizing by SNO is quite slow. Improving the fitting efficiency is the motivation behind Inverse Compositional Image Alignment (ICIA) based approaches. Romdhani et al. [44] extends ICIA to fit 3DMMs. They change the cost function in a way that keeps the Jacobian constant. The Jacobian is a matrix containing the derivatives for every variable (see Appendix A). Consequently updating the Jacobian in every iteration is not necessary which greatly increases the fitting performance. However, ICIA is developed for AAMs that only model shape and texture. Image phenomenas like illumination can not be modeled properly.

Kang et al. [28] propose a multi-resolution fitting approach based on ICIA. First, the authors estimate the parameters on a low resolution model. The fitting procedure of a next higher resolution model is initialized by the previously estimated parameters. A newer multi-resolution fitting approach is presented by Hu et al. [25]. They denote it as Resolution-Aware 3DMM (RA-3DMM) and perform a 3 level multi-resolution fitting, based on Multi-Feature Fitting (MFF) [45]. Multi-resolution fitting reduces the probability of getting stuck in a local minimum [24].

Speeding up the performance is the intention of Romdhani et al. [46], proposing Linear Shape and Texture (LiST). The authors' basic idea is to update the gradients of the shape and texture by solving a linear system. The illumination and rigid transformation parameters are optimized in a non-linear way, because this aspect of the fitting problem can not be converted to a linear problem [46].

Romdhani [45] later extends the original SNO based optimization by including image features like edges, specular highlights and texture constraints. This smooths the cost function and removes the need for using SNO to avoid local minima. MFF [45] is currently the best fitting strategy in terms of accuracy [24].

Recently Hu et al. [24] extended MFF with a facial symmetry constraint prior. They use the fact that a human face is roughly symmetric. The additional constraint enabled to recover better illumination parameters and to outperform the original MFF [24].

Aldrian et al. [1] decompose the image formation process into geometric and photometric parts. They fit the BFM to a single photograph of a face by formulating the problem as a multi-linear system. The face recognition performance is slightly inferior to MFF, but computationally less expensive.

Blanz et al. [9] incorporate the range data of face scans in the cost function. In each iteration a random pixel subset is synthesized. The algorithm aims to make these pixels as similar as possible to the color and depth values of the scan.

1.3.3. Point cloud-based Model Fitting

Fitting 3DMMs to 3D data has become a more active research topic since 3D facial scans can be acquired by relatively cheap depth cameras.

Finding Correspondence

A general challenge with this kind of approache is that the source (*e.g.* scan) must be *semantically consistent* with the target (model). *Semantical consistency* means that a point of the sample must semantically correspond to the same point in the template, *e.g.* tip of the nose.

Finding these correspondences is often performed by means of non-rigid registration. A non-rigid registration aligns two meshes, called *target* and *template*, as closely as possible by manipulating the pose and shape of the *template*. This class of methods usually mimic the properties of an elastic material that can be deformed within certain limits (regularization) [33, 48].

The most relevant methods rely on the Iterative Closest Point (ICP) algorithm [2, 3, 6, 8, 58] and on probabilistic modeling [6]. Further references for probabilistic registration approaches are given in [6]. As we use an ICP-based method we limit the related work

to such. ICP is sensitive to outliers and initialization, but improvements to the original ICP algorithm have been made. The authors of [47] evaluate different variants of ICP in terms of convergence speed. Robustness to outliers and bad initialization has been improved by using probabilistic methods [6, 16].

Fitting Methods

Amberg et al. [2] fit an expression-invariant 3DMM to noisy laser scans using an optimal step non-rigid ICP registration method [3]. In every iteration of the non-rigid ICP algorithm, a 3DMM is fitted to the current set of correspondences. A non-linear objective function recovers the model parameters, rotation and translation. Fitting a 3DMM in every iteration enables to recover expressions of the sample [2].

Zollhöfer et al. [59] aim to fit a 3DMM directly to aggregated data of a consumer depth camera. The authors use facial landmark detection on the 2D color data to find the face and crop it. They deform the mean shape of a 3DMM to the aggregated depth data using the non-rigid registration method from [51]. After the deformation, the template does not represent a realistic face, but point-to-point correspondence is available. The authors use dense correspondences to fit a 3DMM by a linear system. Texturization is done in a vertex-by-vertex manner for the 2D color data. Their approach is not real-time capable, but finishes within a minute.

Weise et al. [55] present a method for real-time facial animation. Microsoft Kinect captures the user's facial performance. They use ICP for rigid alignment of the face with a presegmented template where the chin and eyes are excluded. This stabilizes the rigid-registration, because the eyes and chin cause the strongest deformations in a face. They wrap the template to the user's facial expression using a modification of [33]. To track the facial expressions a blendshape rig [31] is used. A dynamic expression priors regularized the blendshape weights to prevent unrealistic face poses. A user-specific expression model is generated as prerequisite. In a later work, PCA-based learning obsoletes this [32].

The authors of [48] propose two ICP-based algorithms for registering 3D laser scans of human heads. They linearly optimize by a first-order approximation of the original non-linear objective function. An ICP-framework replaces a non-rigid registration and solves the pose and shape estimation at once.

A newer approach is the method from Arellano et al. [6] that works completely without point-to-point correspondences. The sensitivity of ICP to outliers is the motivation behind this approach. They redefine the registration problem as a density estimation problem. The target and template data sets are considered as two separate Probability Density Functions (PDFs). Rigid registration and reconstruction of the model is performed by minimizing the Euclidean distance between these two PDFs. Experiments show that this approach is robust and accurate, but computationally expensive [6].

Chapter Summary

The first chapter of this work gave an introduction into the work. It outlined the thesis' structure, defines the problem we want to tackle and motivates why it is worth to do this. Furthermore an overview about related work is given. Chapter 2 continues with presenting background knowledge of the formulation of a 3DMM and necessary tools for solving the fitting optimization problem.

2. Background

Chapter 1 motivates the purpose of this work. This work is using a 3DMM as model for faces, hence we describe the 3DMM in Section 2.1. Dense point-to-point correspondence is necessary for our fitting method and is achieved by non-rigid registration. The used method is proposed in Section 2.2. As fitting the model comes down to an optimization, Section 2.3 formulates this using Bayes statistics.

2.1. 3D Morphable Models

The aim of a 3DMM is to represent the appearance of a face in a mathematical way. A face can be considered as a class of objects. The selected model should be able to represent any valid face while staying in the space of valid faces.

3DMM is a class of generative statistical models. Generative models can be built by examples. The representational power greatly depends on the variations in the training samples. Generative models were successfully used for many other applications like modeling skulls [41], bodies [5, 56] and for registration [48, 50]. Empirical evaluations showed that several hundred examples are enough to create a model with reasonable representative power [39]. In the context of 3DMM example data is the shape and the texture of the faces. Shape and albedo (skin color) are treated independently, which is not absolutely correct [39]. There are some correlations between shape and albedo, although the assumption of statistical independence is in general good enough for fitting realistic faces.

The core statistical tool for generating a 3DMM is PCA. PCA is often used for dimensionality reduction. In this application it is used to generate a parametrizable representation by an uncorrelated set of coefficients. Fitting a PCA model can be interpreted as fitting a multivariate Gaussian distribution. This gives the possibility to use the model as statistical prior. This fact is useful to prevent the generation of unrealistic faces and will be discussed in Section 2.1.3.

A 3DMM is a parametric model based on a vector space representation of the shape and texture. The shape and texture are represented as vectors

$$\boldsymbol{s} = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^\intercal \tag{2.1}$$

$$\boldsymbol{t} = (r_1, g_1, b_1, \ldots, r_n, g_n, b_n)^\intercal, \tag{2.2}$$

where n is the number of vertices.

The 3DMM itself is given by the mean shape/texture ($\boldsymbol{\mu} \in \mathbb{R}^{3n}$), the principal components ($\boldsymbol{U} \in \mathbb{R}^{3n \times m}$) and the corresponding standard deviation ($\boldsymbol{\sigma} \in \mathbb{R}^{m}$), which correspond to the sample data's eigenvalues. $\boldsymbol{M}_s$ is the shape model and $\boldsymbol{M}_t$ is the texture model.

$$\begin{aligned} \boldsymbol{M}_s &= (\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s, \boldsymbol{U}_s) \\ \boldsymbol{M}_t &= (\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t, \boldsymbol{U}_t) \end{aligned} \tag{2.3}$$

The shape and texture model are independent of each other and can be used individually. As we only fit a shape model in this work, subsequently we omit the subscript s and denote the shape model simply as $\boldsymbol{M}$.

Figure 1 visualizes three model principal components. We render the first three principal components of the model with $\pm 5\sigma_i$. Mathematically formulated we render the following model realizations:

$$\boldsymbol{\mu} \pm \boldsymbol{U}_i 5\sigma_i, \; i = 1, 2, 3$$

2.1.1. Describing Faces

Previously we explained the components and construction of a 3DMM. This section comments on the way a parametrized 3DMM expresses a face. As the description is analogous for texture and shape, we describe it jointly for completeness.

A face shape / texture $\boldsymbol{v}_{s,t} \in \mathbb{R}^{3n}$ can be described as linear combination of the mean shape / texture $\boldsymbol{\mu} \in \mathbb{R}^{3n}$ and a weighted sum of m principal components $\boldsymbol{U}_{s,t} \in \mathbb{R}^{3n \times m}$. The principal components are weighted by the real valued coefficients $\boldsymbol{\theta}_{s,t} \in \mathbb{R}^{m}$. For easier reading, we omit the subscripts s, t:

$$\boldsymbol{v} = \sum_{i}^{m} \boldsymbol{u}_i \sigma_i \theta_i + \boldsymbol{\mu} = \boldsymbol{U} \operatorname{diag}(\sigma)\, \boldsymbol{\theta} + \boldsymbol{\mu} \tag{2.4}$$

The closed form solution for the model parameters $\boldsymbol{\theta}$ is as follows [39]:

$$\boldsymbol{U} \operatorname{diag}(\sigma)\boldsymbol{\theta} + \boldsymbol{\mu} = \boldsymbol{v} \tag{2.5}$$

$$\boldsymbol{U} \operatorname{diag}(\sigma)\boldsymbol{\theta} = \boldsymbol{v} - \boldsymbol{\mu} \tag{2.6}$$

$$\underbrace{\boldsymbol{U}^\intercal \boldsymbol{U}}_{\boldsymbol{I}} \operatorname{diag}(\sigma)\boldsymbol{\theta} = \boldsymbol{U}^\intercal(\boldsymbol{v} - \boldsymbol{\mu}) \tag{2.7}$$

$$\boldsymbol{\theta} = \operatorname{diag}\left(\frac{1}{\sigma}\right)(\boldsymbol{U}^\intercal(\boldsymbol{v} - \boldsymbol{\mu})) \tag{2.8}$$

Since $\boldsymbol{U}$ is orthogonal $\boldsymbol{U}^{-1} = \boldsymbol{U}^\intercal$.

As the coefficients $\boldsymbol{\theta}$ are real valued, continuous morphing in face space is possible. This property can be used to modify *e.g.* the age in a certain range or give the face the appearance of a comic [10].

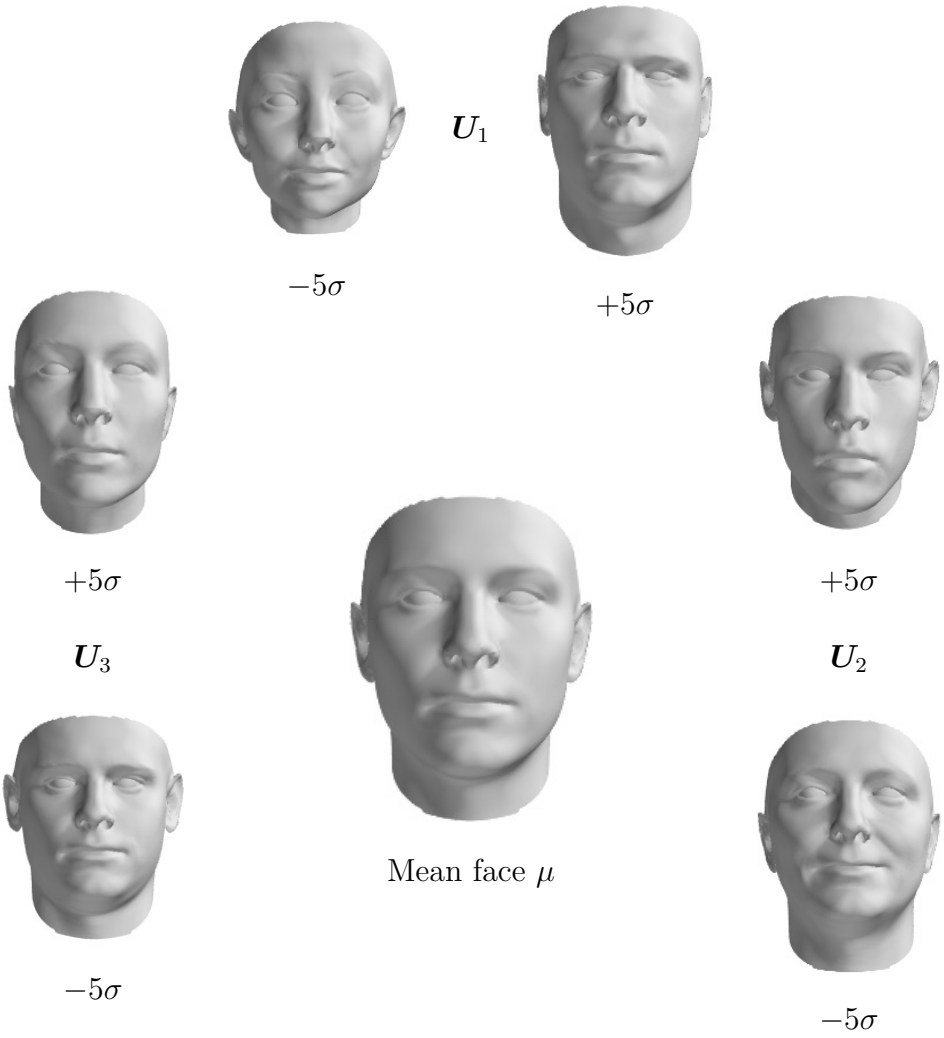


Figure 1.: *Visualization of Principal Components:* Rendered with $\pm 5\sigma$ of a single principal component.

2.1.2. Training a 3DMM

After clarifying the basic structure, we want to describe the model construction.

It is required that the samples have dense correspondences. Correspondence means semantical consistence between point pairs of the sample and the model.

We denote the m example vectors with n vertices as

$$\boldsymbol{v}_i \in \mathbb{R}^{3n},\ i = 1, \ldots, m$$
$$\boldsymbol{v}_i = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^\intercal$$

From all examples the mean is computed by

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{v}_i \tag{2.9}$$

Next the mean-centered samples are arranged in a matrix $\boldsymbol{X} \in \mathbb{R}^{3n \times m}$

$$\boldsymbol{x}_i = \boldsymbol{v}_i - \boldsymbol{\mu},\ i = 1, \ldots, m \tag{2.10}$$
$$\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \tag{2.11}$$

To compute the PCA, $\boldsymbol{X}$ is then decomposed by Singular Value Decomposition (SVD). The SVD decomposition of the $n \times m$ matrix $\boldsymbol{X}$ has the form

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\intercal.$$

$\boldsymbol{U}$ and $\boldsymbol{V}$ are $n \times m$ and $m \times m$ orthogonal matrixes. The columns of $\boldsymbol{U}$ span the column space of $\boldsymbol{X}$ and the columns of $\boldsymbol{V}$ span the row space of $\boldsymbol{X}$. The $m \times m$ diagonal matrix $\boldsymbol{D}$, with diagonal entries $d_1 \geq d_2 \ldots \geq d_m \geq 0$, contains the singular values of $\boldsymbol{X}$ [26, p. 64pp]. $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices, hence $\boldsymbol{U}^\intercal\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\intercal = \boldsymbol{I}$ and $\boldsymbol{V}^\intercal\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^\intercal = \boldsymbol{I}$

Using the SVD formulation the covariance matrix $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$ can be expressed as follows:

$$\boldsymbol{\Sigma} = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\intercal \tag{2.12}$$
$$= \frac{1}{m}(\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\intercal)(\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\intercal)^\intercal \tag{2.13}$$
$$= \frac{1}{m}\boldsymbol{U}\boldsymbol{W}\underbrace{\boldsymbol{V}^\intercal\boldsymbol{V}}_{\boldsymbol{I}}\boldsymbol{W}^\intercal\boldsymbol{U}^\intercal \tag{2.14}$$
$$= \frac{1}{m}\boldsymbol{U}\boldsymbol{W}^2\boldsymbol{U}^\intercal \tag{2.15}$$
$$= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal \tag{2.16}$$

$\boldsymbol{u}_i$ are the eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda} = \mathrm{diag}_i(\lambda_i) = \mathrm{diag}_i(\frac{1}{m}w_i^2)$. The eigenvalues are ordered by their magnitude: $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_m$.

The eigenvalues λ_i are the variance of the covariance matrix. The face space is given by the orthonormal $n \times m$ matrix $\boldsymbol{U}$ with the standard deviation:

$$\sigma_i = \sqrt{\lambda_i} = \sqrt{\frac{1}{m} w_i^2} = \frac{w_i}{\sqrt{m}} \tag{2.17}$$

Refering to Equation 2.3, a 3DMM is given by the principal components $\boldsymbol{U}$, the mean shape $\boldsymbol{\mu}$ and the standard deviation $\text{diag}(\sigma)$.

2.1.3. Face Distribution

Retrospective to the previous section, not every $\boldsymbol{\theta} \in \mathbb{R}^m$ represents a realistic face. The statistical properties of the model lend itself to formulate a prior that constrains the model parameters within the space of valid faces.

The real distribution of face shapes and textures is unknown. However, it is a common assumption that shapes and textures are normally distributed around a mean. This assumption makes it possible to create a distribution prior. We assume that the faces $\boldsymbol{v}$ are distributed by a multivariate normal distribution $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ The probability of $\boldsymbol{v}_i$ being a face can be written as follows. We drop the index i for easier reading:

$$\Pr(v) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{v}-\boldsymbol{\mu})\right) \tag{2.18}$$

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu})^\intercal \boldsymbol{U}^\intercal \boldsymbol{\Lambda}^{-1} \boldsymbol{U}(\boldsymbol{v}-\boldsymbol{\mu})\right) \tag{2.19}$$

$$= \exp\left(-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu})^\intercal \boldsymbol{U}^\intercal \,\text{diag}\left(\frac{1}{\sigma^2}\right) \boldsymbol{U}(\boldsymbol{v}-\boldsymbol{\mu})\right) \tag{2.20}$$

$$= \exp\left(-\frac{1}{2}\left((\boldsymbol{v}-\boldsymbol{\mu})^\intercal \boldsymbol{U}^\intercal \,\text{diag}\left(\frac{1}{\sigma}\right)\right)\left(\text{diag}\left(\frac{1}{\sigma}\right)\boldsymbol{U}(\boldsymbol{v}-\boldsymbol{\mu})\right)\right) \tag{2.21}$$

$$= \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\intercal \boldsymbol{\theta}\right) = \exp\left(-\frac{1}{2}\|\boldsymbol{\theta}\|^2\right) \tag{2.22}$$

In Equation 2.20 $\boldsymbol{\Lambda}$ was substituted by Equation 2.17. Equation 2.21 contains the closed-form solution for $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^\intercal$ from Equation 2.8 and is therefore substituted. Equation 2.22 gives the probability for a face, parametrized by $\boldsymbol{\theta}$, being a realistic face.

The coefficients $\theta_i \sim \mathcal{N}(0,1)$ are assumed to be distributed by the standard normal distribution ($\mu = 0,\ \sigma = 1$). This assumption can be used as a prior for $\boldsymbol{v}$ being a realistic face.

2.1.4. Explanation Power of the Principal Components

PCA is widely used for dimensionality reduction by performing an orthogonal projection into a vector space spanned by the axes with most variance. Usually a low number of

principal components is sufficient to represent a large amount of the total variance, if the data correlation is high. Additionally the computational costs for fitting depend on the number of used principal components. The more principal components are used, the larger the equation system to solve is. We evaluate the explanatory power of the model in this section and provide a basis for deciding the number of necessary principal components in the fitting.

It is possible to measure the explained variance of the individual principal components. The proportion of variance explained by principal component i is given by

$$\text{var explained}(i) = \frac{\lambda_i}{\sum \lambda}. \tag{2.23}$$

Figure 2 shows the cumulated explained variance of the used 3DMM on a logarithmic scale. We clearly see that it is not necessary to use the full amount of 199 principal components. Over 90 % of the variance is explained by only 20 principal components and over 95 % percent by 25 components. The computational costs of the model fitting greatly depend on the number of used principal components.

2.2. Registration

In the previous sections we mention that dense point-to-point correspondences are necessary for fitting. We will later present a fitting method that retrieves them from a deformed template. We gather the correspondences by a modified variant of the non-rigid registration method from [3], which is a template-based registration. The authors named it with the acronym *NICP-A*. 3D registration is a very active research topic, but further registration algorithms are beyond the scope of this thesis. The reader is referred to [27] for a comprehensive survey of shape correspondence methods.

Usually correspondences between template and sample are unknown. Good point-to-point correspondence is a crucial requirement to perform accurate model fitting. Wrong correspondences lead to bad fitting results, because the optimization tries to find parameters for a shape that the model can not synthesize properly. As an example, think about a nose vertex in the model that falsely corresponds to a chin vertex in the sample.

A 3DMM lends itself to be used as a template for registration [3, 33]. We use the model's mean face as deformation template in the registration.

2.2.1. Non-rigid ICP-A

NICP-A is an optimal step non-rigid registration algorithm based on the ICP framework [3]. Sophisticated regularization of the template deformation is a crucial task to achieve

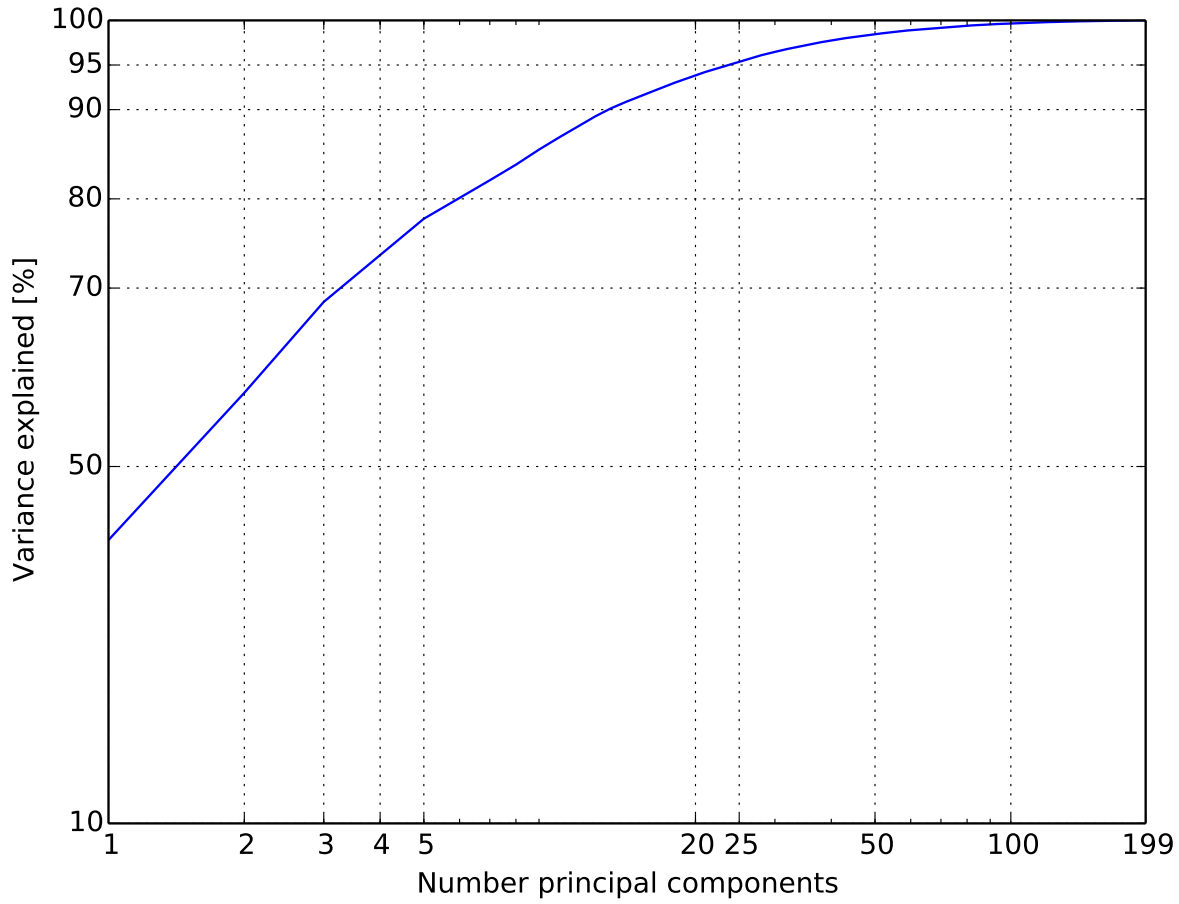


Figure 2.: *Cumulated Explained Variance:* A low number of principal components explains a large percentage of the variance.

good registration results. *NICP-A* includes a locally affine regularization by modeling the mesh as graph.

The algorithm calculates the optimal template deformation in an iterative manner. In every iteration the optimal template deformation for a given stiffness is found. The stiffness controls the allowed non-rigid deformations of the template. The stiffness weight decreases during the iterations and allows stronger non-rigid deformations when the error can not be further reduced with the current regularization.

NICP-A behaves like ICP in the beginning and allows stepwise more non-rigid deformation.

Nonrigid optimal step ICP algorithm

In this section we describe the structure of *NICP-A*. We stick to the notation of [3]. Table 2.1 states the used symbols for the algorithm.

Symbol	Explanation
n	Number of template vertices
α	Stiffness weight (deformation regularization)
$\mathcal{S} = (\mathcal{V}, \mathcal{E})$	Template
$\mathcal{V}$	Template vertices ($\{\boldsymbol{v}_i\}$)
$\mathcal{E}$	Template edges
$\mathcal{T}$	Target surface
$\boldsymbol{v}_i$	Template vertex ($\boldsymbol{v}_i \in \mathcal{V}$)
w_i	Weight of vertex v_i
$\text{dist}^2(\mathcal{T}, \boldsymbol{v}_i)$	distance of $\boldsymbol{v}_i$ to the closest vertex on $\mathcal{T}$
$\boldsymbol{X}$	$4n \times 3$ matrix of unkowns

Table 2.1.: Symbols *NICP-A*

The aim of the algorithm is to find for every vertex i on the template an affine 3×4 transformation matrix $\boldsymbol{R}$. The number of vertices on the template is denoted as n. These transformation matrices are organized in a $4n \times 3$ matrix

$$\boldsymbol{X} = [\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_n]^\intercal$$

The optimization is performed in an iterative ICP-style method. In every iteration point-to-point correspondence between the target surface ($\mathcal{T}$) and the deformed template surface ($\mathcal{V}(\boldsymbol{X}_{j-1})$) is found. $\mathcal{V}(\boldsymbol{X}_{j-1})$ is the original template surface deformed by the optimal deformation found by the previous iteration ($j-1$).

The following pseudo code describes the algorithm:

Algorithm 1 *NICP-A*

```
Initialize X_0
for all α_i ∈ {α_1, α_2, ..., α_n}, α_i > α_{i+1} do
    do
        Find correspondences for V(X_{j-1})
        Weight correspondences
        Find optimal transformation X_j for the correspondences
    while ||X_j − X_{j-1}|| < ε
end for
```

The optimal transformation $\boldsymbol{X}_j$ is acquired by a quadratic function that can be optimized directly.

The cost function consists of two parts: A distance term $E_d(\boldsymbol{X})$ and a stiffness term (regularization) $E_s(\boldsymbol{X})$. The original publication of *NICP-A* includes a landmark term, we omit it, as we do not have landmarks to steer the optimization and it converges also without them [3].

The distance term reflects the natural demand of registration algorithms that the distance between the target and the template should be small. It is given by:

$$E_d(\boldsymbol{X}) = \sum_{\boldsymbol{v}_i \in \mathcal{V}} w_i \operatorname{dist}^2(\mathcal{T}, \boldsymbol{X}_i \boldsymbol{v}_i) \tag{2.24}$$

$\boldsymbol{X}_i \boldsymbol{v}_i$ is the transformed template vertex $\boldsymbol{v}_i$ with the currently determined optimal transformation $\boldsymbol{X}_i$. $\operatorname{dist}^2(\mathcal{T}, \boldsymbol{v})$ is the squared point-to-point distance between the closest vertex on surface $\mathcal{T}$ and vertex $\boldsymbol{v}$.

The stiffness term regularizes the non-rigid deformation of the template. This is the actual reason why *NICP-A* behaves like rigid ICP in the beginning. A high stiffness weight α_i reflects a high penalization of non-rigid template deformation. The weighted difference of the transformations of neighboring vertices is penalized under the Frobenius norm ($\|\cdot\|_F$). The set $\mathcal{E}$ is given by tuples of all neighboring vertices in the template mesh. Assuming vertex $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are connected by an edge, $\mathcal{E}$ contains the tuple $(\boldsymbol{v}_i, \boldsymbol{v}_j)$. $\boldsymbol{G} = \operatorname{diag}(1, 1, 1, \gamma)$ is a weighting matrix using the weighting factor γ.

$$E_d(\boldsymbol{X}) = \sum_{(i,j) \in \mathcal{E}} \|(\boldsymbol{X}_i - \boldsymbol{X}_j)\boldsymbol{G}\|_F^2 \tag{2.25}$$

The full cost function for the registration is given by:

$$E(\boldsymbol{X}) = E_d(\boldsymbol{X}) + \alpha E_s(\boldsymbol{X}) \tag{2.26}$$

In Equation 2.26 α is the stiffness weight. In Equation 2.27 the cost function from Equation 2.26 is rearranged in a way that can easily be optimized by the normal equations. The first row is the stiffness term E_s, the second the distance term E_d. The matrix $\boldsymbol{M}$ is a node-arc incidence matrix and represents the neighborhood in the mesh. $\boldsymbol{D}$ is a block diagonal sparse matrix,

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{v}_1^\mathsf{T} & & & \\ & \boldsymbol{v}_2^\mathsf{T} & & \\ & & \ddots & \\ & & & \boldsymbol{v}_n^\mathsf{T} \end{bmatrix}.$$

$\boldsymbol{U}$ is the correspondence points of the template vertices and $\boldsymbol{W}$ weights the correspondences.

$$E(\boldsymbol{X}) = \left\| \begin{bmatrix} \alpha \boldsymbol{M} \otimes \boldsymbol{G} \\ \boldsymbol{W}\boldsymbol{D} \\ \beta \boldsymbol{D}_L \end{bmatrix} \boldsymbol{X} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{W}\boldsymbol{U} \\ \boldsymbol{U}_\boldsymbol{L} \end{bmatrix} \right\|_F^2 = \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F^2 \tag{2.27}$$

The solution for this optimization problem

$$\arg\min_{\boldsymbol{X}} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F^2 \tag{2.28}$$

is the widely known solution of a linear least-squares problem. Deriving the minimum of $E(\boldsymbol{X})$ by calculating $\frac{\partial \|\boldsymbol{AX}-\boldsymbol{B}\|_F^2}{\partial \boldsymbol{X}} = \boldsymbol{0}$ [23, p. 12] leads to:

$$\boldsymbol{X} = (\boldsymbol{A}^\intercal \boldsymbol{A})^{-1} \boldsymbol{A}^\intercal \boldsymbol{B}$$

A proof that $\boldsymbol{A}$ has rank $4n$ and hence $\boldsymbol{A}^\intercal \boldsymbol{A}$ is invertible is given in [3].

2.3. Bayesian Model Fitting

For the problem formulation and derivation of the mathematical formulation we follow Amberg [4] and Weise et al. [55]. Their formulation is very clear and provides a framework for different fitting methods.

Fitting a 3DMM might be interpreted as estimating the parameters $\boldsymbol{\theta}$ that were used to generate an facial image $\mathcal{V}$. We denote all measurements of the 3D object as *image*. More precisely we want to determine the distribution over the parameters $\boldsymbol{\theta}$ that were used to generate $\mathcal{V}$, assuming that $\mathcal{V}$ was generated by the model.

This kind of problem can be formulated in a clean way using maximum a posteriori probability (MAP).

2.3.1. Formulating as Maximum a Posteriori Probability Problem

To derive the formulation we assume that image $\mathcal{V}$ was generated by the model G. As this it not true for fitting the model to a real face, we add an error term ϵ. We assume that the observed image $\mathcal{V}$ is generated as

$$\mathcal{V} = G(\boldsymbol{\theta}) + \epsilon$$

by G parametrized by $\boldsymbol{\theta}$ with error ϵ. We want to maximize the probability $\Pr(\boldsymbol{\theta}|\mathcal{V})$ of the model parameters $\boldsymbol{\theta}$ given the observed image $\mathcal{V}$. We therefore solve the following MAP problem:

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \Pr(\boldsymbol{\theta}|\mathcal{V}) && (2.29)\\
&= \arg\max_{\boldsymbol{\theta}} \frac{\Pr(\mathcal{V}|\boldsymbol{\theta}) \Pr(\boldsymbol{\theta})}{\Pr(\mathcal{V})} && (2.30)\\
&= \arg\max_{\boldsymbol{\theta}} \Pr(\mathcal{V}|\boldsymbol{\theta}) \Pr(\boldsymbol{\theta}) && (2.31)\\
&= \arg\min_{\boldsymbol{\theta}} -\ln(\Pr(\mathcal{V}|\boldsymbol{\theta}) \Pr(\boldsymbol{\theta})) && (2.32)\\
&= \arg\min_{\boldsymbol{\theta}} -\ln \Pr(\mathcal{V}|\boldsymbol{\theta}) - \ln \Pr(\boldsymbol{\theta}) && (2.33)
\end{aligned}$$

The probability $\Pr(\mathcal{V})$ of $\mathcal{V}$ being a face is constant for all $\boldsymbol{\theta}$ and does not have any influence on optimizing $\boldsymbol{\theta}$. Therefore we simplify by discarding the term after Equation 2.30.

In the further work we define the fitting energy as

$$E_{\text{fit}} = -\ln \Pr(\mathcal{V}|\boldsymbol{\theta}) \tag{2.34}$$

and the prior energy as

$$E_{\text{prior}} = -\ln \Pr(\boldsymbol{\theta}) \tag{2.35}$$

E_{fit} is the energy related to how well the model describes the given sample $\mathcal{V}$. The prior energy E_{prior} is low if the parameters $\boldsymbol{\theta}$ are realistic parameters for the model. Section 2.4 describes the design of the prior in more detail.

We use Equation 2.33 as base framework for minimizing different fitting energies and priors. Section 2.4 presents more insights into the shape of the prior.

2.4. Modelling the Prior

Not all parameters $\boldsymbol{\theta}$ represent a realistic face, therefore a regularization is necessary. Following we present two priors.

The prior probability $\Pr(\boldsymbol{\theta})$ regularizes $\boldsymbol{\theta}$. Referring to Equation 2.22 we know that the prior of the 3DMM is

$$\Pr(\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\|\boldsymbol{\theta}\|^2\right) \propto \exp\left(-\|\boldsymbol{\theta}\|^2\right). \tag{2.36}$$

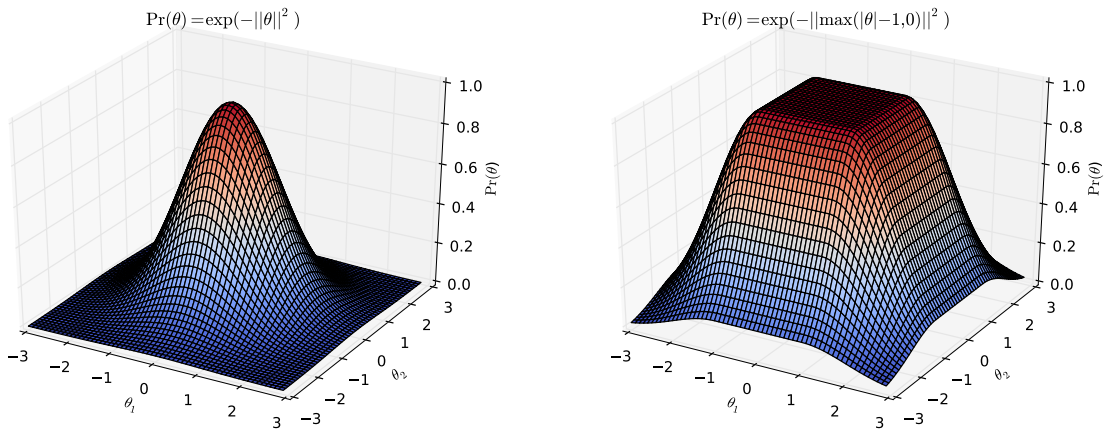
This is proportional to a Gaussian Distribution (see Section 2.1.3).

However, Amberg [4] observed that better fitting results are achieved by modeling the prior as

$$\Pr(\boldsymbol{\theta}) = \exp\left(-\sum_{i}^{m} \max(|\theta_i| - 1, 0)^2\right). \tag{2.37}$$

This prior assigns equal probability to all faces whose coefficients have an absolute value smaller than 1. Outside of the $[-1, 1]^d$-cube the distribution drops off like a standardnormal Gaussian [4].

For better clarification we give a plot for a two dimensional $\boldsymbol{\theta}$ in Figure 3. Figure 3(a) shows the prior in Equation 2.36, Figure 3(b) respective for Equation 2.37.



(a) Gaussian Prior **(b)** Modified Gaussian

Figure 3.: Prior Distributions

2.5. Point Distances

A cost function usually models some kind of error that should be minimized. In our case we want to minimize the distance between two point clouds and define the distance function as φ. The point clouds are given by the model and respectively the sample. We present two distance measures for points, namely the point-to-point distance in Section 2.5.1 and the point-to-plane distance in Section 2.5.2.

2.5.1. Point to Point Distance

The most common distance measure for points is the point-to-point distance, also known as the Euclidean distance. For a point $\boldsymbol{x}$ and another point $\boldsymbol{p} \in \mathcal{P}$, where $\mathcal{P}$ is the target surface, the point-to-point distance φ is defined as follows:

$$d(\boldsymbol{x}, \boldsymbol{p}) \approx \|\boldsymbol{p} - \boldsymbol{x}\|_2 \tag{2.38}$$

Calculating the distance to a first-order approximation of the target surface leads to the point-to-plane distance.

2.5.2. Point to Plane Distance

The point-to-plane error measure can be seen a first-order approximation of the surface. Rusinkiewicz et al. [47] observed experimentally that the point-to-plane ICP variant has better convergence speed and is more robust than the point-to-plane variant. Subsequently Pottmann et al. [43] demonstrated formally that the classic ICP has linear, respective

quadratic convergence speed. This was achieved using the following Taylor expansion of the point-to-point distance:

For a "foot point" $\bar{\boldsymbol{p}}$, *i.e.* the closest point onto the surface $\mathcal{P}$ to the point $\boldsymbol{x}$, the Taylor expansion of d is given as

$$\tilde{d}(\boldsymbol{x}, \bar{\boldsymbol{p}}) \approx d(\boldsymbol{x}, \bar{\boldsymbol{p}}) + \Delta d(\boldsymbol{x}_0)^\intercal \cdot (\boldsymbol{x} - \bar{\boldsymbol{p}}) \tag{2.39}$$

$$= \|\boldsymbol{x}_0 - \bar{\boldsymbol{p}}\|_2 + \left(\frac{\boldsymbol{x}_0 - \bar{\boldsymbol{p}}}{\|\boldsymbol{x}_0 - \bar{\boldsymbol{p}}\|_2} \right)^\intercal \cdot (\boldsymbol{x} - \bar{\boldsymbol{p}}) \tag{2.40}$$

As $\boldsymbol{x}$ approaches $\bar{\boldsymbol{p}} \in \mathcal{P}$, the term $\|\boldsymbol{x} - \bar{\boldsymbol{p}}\|_2$ approximates 0 and vanishes. The term $\left(\frac{\boldsymbol{x}_0 - \bar{\boldsymbol{p}}}{\|\boldsymbol{x}_0 - \bar{\boldsymbol{p}}\|_2} \right)^\intercal$ is the unit surface normal $\boldsymbol{n}$ in point $\boldsymbol{p} \in P$. Hence Equation 2.40 can be written as

$$\tilde{d}(\boldsymbol{x}, \bar{\boldsymbol{p}}) \approx \boldsymbol{n}^\intercal \cdot (\boldsymbol{x} - \bar{\boldsymbol{p}}), \tag{2.41}$$

which describes the well known point-to-plane distance [21].

One can see the point-to-plane distance as allowing a source point $\boldsymbol{s}_i$ to "slide" on the normal plane of a target point $\boldsymbol{t}_i$, while the distance is constant. This property is especially useful when the template and target mesh have different densities. Figure 4 illustrates the difference between the point-to-point and point-to-plane distance.

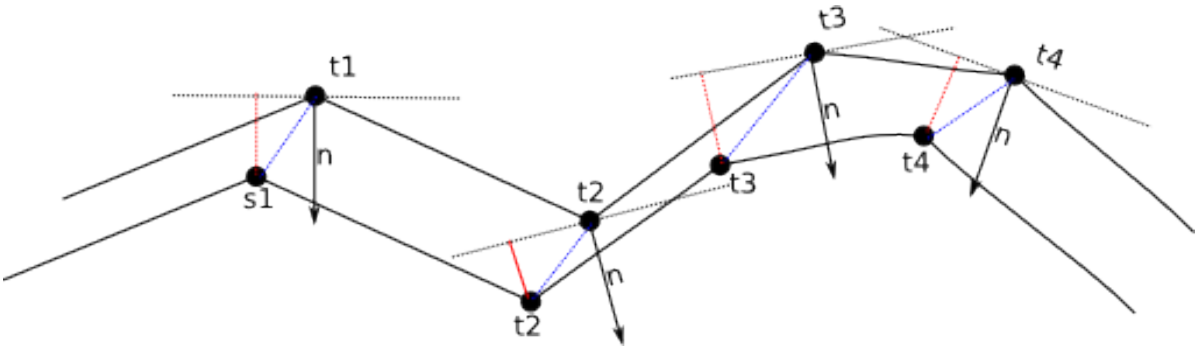


Figure 4.: *Point to Point/Plane distance:* $\boldsymbol{t}_i$ are the target vertices and $\boldsymbol{s}_i$ the source vertices. The black dashed lines are the tangential planes of the normals. The red lines are the point-to-plane distance, the blue lines the point-to-point surface.

Calculating point normals for a mesh is straightforward. Assuming a triangular mesh we compute the normals for a triangle $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$ as follows:

$$\boldsymbol{n}_{1,2,3} = (\boldsymbol{v}_1 - \boldsymbol{v}_2) \times (\boldsymbol{v}_1 - \boldsymbol{v}_3) \tag{2.42}$$

The unit length normals $\hat{\boldsymbol{n}}_i$ are consequently computed as

$$\hat{\boldsymbol{n}}_i = \frac{\boldsymbol{n}_i}{\|\boldsymbol{n}_i\|}. \tag{2.43}$$

Chapter Summary

Recapitulating the previous chapter, we introduce the formulation of the 3DMM in Section 2.1. Moreover we present how faces are described in Section 2.1.1. An obvious prerequisite is training a model. The way to do this and why this is a challenging task is presented in Section 2.1.2. The model also has an inherent statistical prior. The derivation of it is described in Section 2.1.3. Section 2.1.4 shows why the used 3DMM is a compact model with strong expressive power. How to solve the fitting problem in a probabilistic way is described in Section 2.3.

3. Method

Chapter 2 gives the foundation for the main part of this work. This chapter describes the proposed pipeline and different fitting methods. A high-level overview of the fitting method is given in Section 3.1. Three different fitting methods that can be embedded are presented in the consecutive sections.

The first method establishes correspondence by non-rigidly deforming the model's mean shape and then recovers the model coefficients using a non-linear least-squares optimization. We describe it in Section 3.3. We developed a correspondence weighting and rejection scheme for the registration method to increase the robustness. We explain it in Section 3.4.

The second method, described in Section 3.7, is based on the same non-linear least-squares optimization, but does not need a non-rigid template deformation to acquire dense correspondence. It finds the correspondences and model parameters in an iterative manner.

An advancement of method one and two constitutes the third method: Section 3.8 explains a fully linear fitting method. This method is faster and easier to optimize, because no non-linear optimization is needed.

3.1. Fitting Strategy

After stating the necessary foundation, we describe the fitting pipeline that constitutes the core part of this work.

Figure 5 shows the structure of the pipeline, which we describe following in more detail. The input is a mesh and the outputs are coefficients of the fitted model.

Rough initial alignment is a prerequisite for all other steps. A sufficient initial alignment is necessary for all subsequent steps to converge.

Finding correspondence is necessary to fit the model parameters in a least-squares manner.

Fitting the model comes down to a linear or non-linear optimization problem that aims to find the model parameters for a given set of point correspondences.

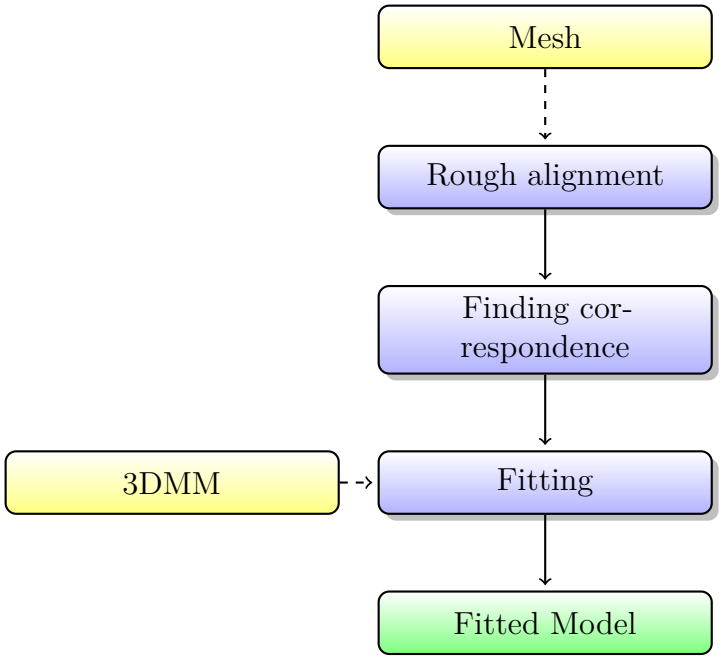


Figure 5.: Pipeline Structure

Rough alignment is in general a requirement for ICP based methods. ICP is quite sensitive to initialization [47]. The current version of this pipeline requires a manual rough alignment to succeed. Different solutions to this problem are discussed in Section 5.3.1.

Before describing the different fitting functions, we state the used notation in Section 3.2 for common understanding.

3.2. Notation of Objective Functions

We denote $f(\cdot)$ as the objective function. The model parameters are denoted as $\boldsymbol{\theta} \in \mathbb{R}^m$. The model data is denoted as introduced in Equation 2.3. The rotation $\boldsymbol{R}$ is parametrized by 3 Euler angles and is evaluated in the objective function with a 3×3 rotation matrix. The translation is a 3-element vector, corresponding to t_x, t_y, t_z.

3.3. Non-linear Fitting from a Deformed Template

The first method can be seen as extension of the work from [2] and [59]. We modify the cost function of Amberg et al. [2] to minimize the point-to-plane error which is known to be more robust [47]. For recovering the model parameters, dense point-to-point correspondence is required. This is achieved by non-rigidly deforming the model's mean

shape (see Section 2.1.1) to the sample. As the sample can be incomplete and noisy, the deformed template might look unnatural. The deformed mean shape is then used to recover the model parameters in a non-linear least-squares optimization. As the deformed mean shape has the same point topology and number of points, we have dense correspondence.

Considering the last paragraph, this fitting method consists of two steps. First, non-rigidly deforming the model's mean shape to the sample. Second, recovering the model parameters using a non-linear least-squares optimization. We describe these steps in Section 3.4 and respectively Section 3.5. We extend the original cost function by point-to-plane constraints in Section 3.6.

3.4. Finding Correspondences by Deforming the Mean Shape

The model fitting needs dense correspondences. More specifically, we need the semantically correct correspondence of the model's vertices and the target's vertices. To achieve this, the mean face of the 3DMM gets registered towards the sample in a non-rigid fashion.

The non-rigid ICP framework of *NICP-A* [3] is used in this work. ICP methods are sensitive to outliers and bad correspondences [12, 47]. To decrease the influence of bad correspondences and outliers, often a correspondence weighting and rejection scheme is introduced [47]. The next sections present a correspondence scheme that both rejects outliers and weights correspondences based on their distance. To avoid clustering of several sample points on the template, we also present a method that rejects correspondences that are likely to cause point clustering.

3.4.1. Rejecting of Correspondences

Weighting and rejection scheme improves the convergence rate and helps to avoid getting stuck in local minima.

Looking at Equation 2.27 and 2.24 we are able to weight the correspondences. We denote setting the weight of a correspondence to zero as *rejecting* it.

Border Vertices

We reject border vertices in order to avoid unravelled edges of the deformed template. By rejecting border vertices the border of the template mesh stays smooth.

Calculation of the border vertices is performed by employing mesh connectivity constraints: Boundary edges of a triangular mesh are only referenced by single triangle. By finding the unique edges we get the border vertices.

Incompatible Surface Normals

Rejecting points with incompatible surface normals is a widespread approach in the literature [3, 47, 55]. A correspondence pair gets rejected if the angle between the surface normals exceeds a certain threshold.

Calculating the angle between two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$ is performed by the well known formula:

$$\cos \alpha = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|} \tag{3.1}$$

If the normals have unit length, the denominator may be omitted.

Cross-side Matching

The cross-side matching (CSM) method was developed to prevent self intersections of the template and point clustering: The face model also models the inner parts of the nose. A facial scan does usually not model inner parts of the nose properly, so inner part nose points can correspond to points such that a self intersection occurs. Successively this leads to unnatural template deformations. Deformations that cluster template points together happen when several template points have a single target point as correspondence. The proposed cross-side matching method also efficiently rejects a great part of these correspondences.

Both unnatural deformation and point clusters have negative influence on successive steps. Especially the model fitting, which minimizes a quadratic function, is sensitive to outliers. In order to avoid that, we present cross-side matching. This method rejects correspondences by checking the closest points from two sides: From the template side and from the sample side. This section explains the method. We begin by stating the notation.

$\mathcal{C}_T(\boldsymbol{v}_i)$ denotes the nearest neighbor of point $\boldsymbol{v}_i$ on the target $\mathcal{T}$. $\mathcal{C}_V(\boldsymbol{v}_i, n)$ returns a set of the n-nearest neighbors on $\mathcal{T}$. $\mathcal{C}_V(\boldsymbol{v}_i)$ gives the nearest neighbor of $\boldsymbol{v}_i$ on in the set of template points $\mathcal{V}$. $d(\cdot)$ is a distance operator that gives the distance of a given vertex to the nearest neighbor on the other surface, hence $d(\boldsymbol{v}_i),\ \boldsymbol{v}_i \in \mathcal{V}$ gives the distance to the nearest point on the target for template vertex $\boldsymbol{v}_i$.

CSM-hard is the most restrictive method. Correspondences are only accepted when the corresponding point on the template and on the target are the same, seen from both sides. Equation 3.2 gives a mathematical definition of CSM-hard. The vertex weight is set to one in the following case: Template vertex $\boldsymbol{v}_i$ has target vertex $\boldsymbol{t}_i$ as closest vertex. The closest vertex on the template of vertex $\boldsymbol{t}_i$ is $\boldsymbol{v}_i$. Equation 3.2 states the above in a more formal way.

$$w_i = \begin{cases} 1 & \text{if } \boldsymbol{v}_i = \mathcal{C}_V(\mathcal{C}_T(\boldsymbol{v}_i)) \\ 0 & \text{else} \end{cases} \tag{3.2}$$

A clear disadvantage of this method is that a lot of vertices get rejected if the target is sparser than the sample. It prevents association of multiple template vertices to the same template vertex.

CSM-soft is similar to CSM-hard. It rejects a correspondence pair, if $\boldsymbol{v}_i$ is not in the set of the n-nearest neighbours given by $\mathcal{C}_V(\boldsymbol{p}_i, n)$. $\boldsymbol{p}_i$ denotes the nearest neighbour of $\boldsymbol{v}_i$ on the target ($\boldsymbol{p}_i = \mathcal{C}_T(\boldsymbol{v}_i)$). Equation 3.3 describes the above more formal.

$$w_i = \begin{cases} 1 & \text{if } \boldsymbol{v}_i \in \{\mathcal{C}_V(\mathcal{C}_T(\boldsymbol{v}_i), n)\} \\ 0 & \text{else} \end{cases} \tag{3.3}$$

CSM-distance accepts correspondence pairs, if the distance between the nearest neighbor of $\boldsymbol{v}_i$ on the target ($\boldsymbol{p}_i = \mathcal{C}_T(\boldsymbol{v}_i)$) and the distance of the nearest-neighbor of $\boldsymbol{p}_i$ on the template is smaller or equal than $\text{dist}^2(\mathcal{V}, \boldsymbol{v}_i)$ plus a small tolerance. We use the squared Euclidean point-to-point distance. Equation 3.4 gives a more formal description of this textual description.

This is especially important, if the sample vertices are sparser than the template vertices. In this case multiple template vertices can be assigned to the same vertex on the target.

$$w_i = \begin{cases} 1 & \text{if } \text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) \leq \text{dist}^2(\mathcal{V}, C_T(\boldsymbol{v}_i, 1)) + \Delta d \\ 0 & \text{else} \end{cases} \tag{3.4}$$

3.4.2. Robust Correspondence Weighting

The registration has to provide robustness to outliers and missing regions. We use a robust correspondence weighting similar to [2].

The weight w_i of the correspondence of template vertex $\boldsymbol{v}_i$ is defined as follows in every iteration of the registration algorithm. $\text{dist}^2(\mathcal{T}, \boldsymbol{v}_i)$ is the squared distance of template vertex $\boldsymbol{v}_i$ to the closest vertex on the target surface $\mathcal{T}$.

$$w_i = \begin{cases} \text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) & \text{if } \text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) \leq \text{max linear weight distance} \\ 1/\,\text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) & \text{else if } \text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) \leq \text{max pair distance} \\ 0 & \text{else if } \text{dist}^2(\mathcal{T}, \boldsymbol{v}_i) > \text{max linear weight distance} \end{cases}$$

This method is suitable, because it rejects points that are too far away and does not overweigh very close points. Overweighting very close points which are most probably already well registered, results in clustering of points. As we estimate a transformation for every point, also the mesh topology is changed. We want to preserve the mesh topology as well as possible. As non-rigid ICP (NICP) minimizes the point to point distance point, point clustering is a problem when registering meshes with different densities. Good weight selection does not solve this problem completely, but decreases the influence. Another desirable property of this weighting scheme is that it weighs points that are likely correct correspondences, but not yet registered higher.

Consider the following examples: A point which has a distance of 3 mm to its correspondence is weighted with $1/0.0003 \approx 3333$. If the majority of all points is for example within 2 mm and an outliers point is 6 mm away, it has a weight of ≈ 1666, but most other points have ≈ 5000. The weighting scheme underweights this point and makes it contribute less to the transformation estimation.

Summarizing this section, the output of the non-rigid registration is a deformed mean shape of the 3DMM. Figure 6 shows a typical result of the non-rigid registration. The eye and mouth regions show unnaturally deformed regions and the ear is completely destroyed. This is due to holes in the sample and/or bad correspondences. In order to acquire a natural and complete facial mesh, we fit the 3DMM to the deformed mean shape.

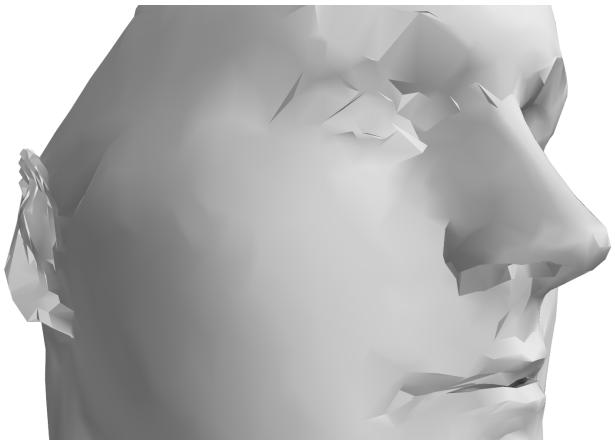


Figure 6.: *Deformed mean shape:* Registered model mean shape. Unnatural deformations due to holes in the scan are clearly visible in the eye and mouth area. The ear is completely crimped.

For this template deformation approach, we present a point-to-point and a point-to-plane constrained cost function. The first is presented in Section 3.5 and the latter in Section 3.6.

3.5. Point-to-Point Non-linear Model Fitting

After having acquired dense point correspondence by means of the non-rigid registration, we can fit the 3DMM to the deformed mean shape.

Fitting a 3DMM in a point-to-point manner is the most straight forward approach for fitting a 3DMM. Our point-to-point cost function is the same as Amberg et al. [2] use.

Following our problem formulation and Equation 2.33, we define the fitting energy E_{fit} and the prior energy E_{prior} as follows. Index i denotes the error for the i-th point.

$$E_{\text{fit}} = \sum_i \|\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i\|^2 \tag{3.5}$$

$$E_{\text{prior}} = \lambda\|\boldsymbol{\theta}\|^2 \tag{3.6}$$

The prior is weighted by the regularization weight λ. For a derivation of the prior see Section 2.4. As the registration can introduce additional rotation and translation, the cost function also optimizes for a rotation and a translation. The full cost function is given by

$$E = E_{\text{fit}} + E_{\text{prior}} \tag{3.7}$$

The aim of the optimization is to find the arguments $\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{\theta}$ that minimize E, more precisely

$$\underset{\boldsymbol{R},\boldsymbol{t},\boldsymbol{\theta}}{\arg\min} \sum_i \|\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i\|^2 + \lambda\|\boldsymbol{\theta}\|^2 \tag{3.8}$$

Equation 3.8 is a non-linear function, hence can not be optimized directly by setting the derivatives zero. We optimize using the Levenberg-Marquardt (LM) algorithm, which is an iterative algorithm to solve non-linear least-squares problems. For faster computation, we optimize using analytical derivatives instead of numerical derivatives.

To ease the computation of derivatives, the rotation is inversed and the objective function is reformulated as follows:

$$f(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{\theta}) = \sum_i \|\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\boldsymbol{\theta} + \boldsymbol{t}' - \boldsymbol{R}'\boldsymbol{v}_i\|^2 + \lambda\|\boldsymbol{\theta}\|^2 \tag{3.9}$$

$$\boldsymbol{R}' = \boldsymbol{R}^\intercal \qquad \boldsymbol{t}' = \boldsymbol{R}^\intercal\boldsymbol{t}$$

While performing an iterative optimization, the Jacobian matrix has to be evaluated every iteration. For the cost function in Equation 3.9, the Jacobian can be split up into

a constant and a dynamic part [2]. The dynamic part of the Jacobian $\boldsymbol{J}_d$ has to be evaluated in every iteration. The constant part $\boldsymbol{J}_c$ can be precomputed.

$$\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{U}\,\mathrm{diag}(\sigma) & \mathbf{1} \otimes \boldsymbol{I}_3 \\ \boldsymbol{I} & \mathbf{0} \end{bmatrix} \tag{3.10}$$

$$\boldsymbol{J}_d = \begin{bmatrix} \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}^\intercal & \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}^\intercal & \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}^\intercal \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.11}$$

The full Jacobian is given by

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_c & \boldsymbol{J}_d \end{bmatrix}. \tag{3.12}$$

The derivation of the derivatives is stated in Section A.1 in the appendix.

We evaluate and discuss the results of this cost function in Chapter 4 and continue with extending it by using point-to-plane constraints.

3.6. Point-to-Plane Non-linear Model Fitting

We extend the point-to-point based cost function presented in Section 3.5 with a point-to-plane constraint. Point-to-plane distance is derived and explained in Section 2.5.2. In a nutshell, cost functions using point-to-plane constraints often converge faster and are more robust [43, 47].

We employ point-to-plane constraints by extending the fitting energy term E_{fit} from Equation 3.9 and writing it as

$$E_{\text{fit}} = \sum_i \left((\boldsymbol{\mu}_i + \boldsymbol{U}_i\,\mathrm{diag}(\sigma)\,\boldsymbol{\theta} + \boldsymbol{t}' - \boldsymbol{R}'\boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2. \tag{3.13}$$
$$\boldsymbol{R}' = \boldsymbol{R}^\intercal \qquad \boldsymbol{t}' = \boldsymbol{R}^\intercal \boldsymbol{t}$$

$\boldsymbol{n}_i \in \mathbb{R}^3$ is the unit length normal of sample point $\boldsymbol{v}_i \in \mathbb{R}^3$. Without changing the prior energy E_{prior} the full cost function is given by

$$f(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{\theta}) = \sum_i \left((\boldsymbol{\mu}_i + \boldsymbol{U}_i\,\mathrm{diag}(\sigma)\,\boldsymbol{\theta} + \boldsymbol{t}' - \boldsymbol{R}'\boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 + \lambda \|\boldsymbol{\theta}\|^2 \tag{3.14}$$
$$\boldsymbol{R}' = \boldsymbol{R}^\intercal \qquad \boldsymbol{t}' = \boldsymbol{R}^\intercal \boldsymbol{t}.$$

As the result of the dot product is a scalar, we replace the sum of norms with a sum over the squared signed point-to-plane distances.

The optimization is performed in the same way as for the point-to-point objective function in Section 3.5. The analytical derivatives can be found in appendix Section A.2.

Summing up, we extended the cost function of Amberg et al. [2] with a more robust error metric. Both solve the correspondence problem by non-rigid registration method. This method has two main disadvantages: First it is computationally expensive. Second

possibly better correspondences, acquired using a fitted model, can not be used. These disadvantages encourage us to fit the model in an iterative way, which enables us to use possibly better correspondences in every iteration. We continue by tackling this limitation in Section 3.7.

3.7. Non-linear Iterative Fitting

In the previous sections we introduced a method that deforms the model's mean shape non-rigidly and then uses it to recover the model parameters. The registration is limited by its deformation capabilities by a regularization. This means that the model's mean shape can only be deformed to a certain extent. As the deformed template represents the correspondences used for model fitting, it is possible that there are better correspondences available. We approach this issue by putting the cost function into an iterative framework that refines the model iteratively with better correspondences.

The difference to the previous method is that we directly work with noisy correspondences. We do not fit to the already registered mean shape. Registering the template avoids fitting to outliers and noisy points. The ℓ_2 norm is known to be sensitive to outliers [12, 47]. An often used method to reduce the influence of outliers and bad correspondences is to use a weighting and rejection scheme. Therefore we extend Equation 3.14 by a weight $w_i \in \mathbb{R}$ for every correspondence. The weight is selected by a distance-based threshold.

The cost function for iterative non-linear fitting is given by

$$f(\boldsymbol{R}, \boldsymbol{t}, \boldsymbol{\theta}) = \sum_i w_i \left((\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta} + \boldsymbol{t}' - \boldsymbol{R}' \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 + \lambda \|\boldsymbol{\theta}\|^2 \qquad (3.15)$$
$$\boldsymbol{R}' = \boldsymbol{R}^\intercal \qquad \boldsymbol{t}' = \boldsymbol{R}^\intercal \boldsymbol{t}.$$

The fitted model iteratively approaches the sample points. We acquire the best model fit by alternating between model fitting and acquiring correspondences. In every iteration the acquired model fits the sample more closely. Listing 2 shows the algorithm in pseudocode. K denotes the closest points for model points M on sample $\mathcal{T}$. The trailing model M is initialized with the model's mean shape. The algorithm iterates then between finding new correspondences and fitting the model until the absolute difference between the fitting error of the current and the previous iteration is below a threshold.

Algorithm 2 Iterative non-linear model fitting

```
M = μ
while |E − E_old| < threshold do
    Find closest points K for M on T.
    M = Fit model to correspondences K.
    E_old = E
    E = Evaluate model error for M on T.
end while
```

This section explains a method that solves the correspondence problem in an iterative fashion. The motivation behind this is to acquire possibly better correspondences by performing iterative steps. As an ℓ_2 norm is minimized in our cost function, we introduce correspondence weighting for robustness. This reduces the influence of outliers and speeds up the convergence.

Minimizing continuous non-convex error functions involves the risk of getting stuck in local minima. As a 3DMM is a linear model that can be optimized by solving a linear equation system, we present a method that fits the 3DMM fully linear including rigid pose estimation. We derive and present this method in Section 3.8.

3.8. Linear Iterative Fitting

All previously presented cost functions are non-linear. Minimizing a continous non-convex function has always the risk of getting stuck in a local minima. Minimizing a convex cost function is in most cases easier, faster and not subject to get stuck in local minima. A 3DMM is in fact a linear model that can be optimized by solving a linear equation system. Previously used cost functions also solve for the model's pose. We demonstrate in this section that we can solve *alternately* for correspondences, pose and the model coefficients by solving several linear problems.

We decompose the previously used fitting energy

$$E_{\text{fit}} = \sum_i \left((\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 \tag{3.16}$$

into a term solving for the rigid pose and a second term solving for the model coefficients. We solve for the rotation matrix $\boldsymbol{R}$, translation $\boldsymbol{T}$ and model coefficients $\boldsymbol{\theta}$.

$$\arg\min_{\boldsymbol{R},\boldsymbol{t},\boldsymbol{\theta}} \sum_i \left((\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 \tag{3.17}$$

We reformulate the cost function into two parts, where $\boldsymbol{m}_i = \boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}$

$$\arg\min_{\boldsymbol{R},\boldsymbol{t},\boldsymbol{\theta}} \sum_i \left((\boldsymbol{R}\,\boldsymbol{m}_i + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 + \left((\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 \tag{3.18}$$

Subsequently decompose this into two independent steps:

$$\textbf{Step 1:} \qquad \arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_i \left((\boldsymbol{R}\,\boldsymbol{m}_i + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 \tag{3.19}$$

$$\textbf{Step 2:} \qquad \arg\min_{\boldsymbol{\theta}} \sum_i \left((\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2 \tag{3.20}$$

We *alternately* solve step 1 and step 2. In the first step the model is fixed and the only free parameters are $\boldsymbol{R}, \boldsymbol{t}$. This problem is the classical discrete pairwise registration problem

and can be solved using the ICP algorithm [12, 58]. We give a short overview about ICP in Section 3.8.1 and explain how to recover the optimal transformation. In the second step, $\boldsymbol{R}$ and $\boldsymbol{t}$ are fixed and we only solve for $\boldsymbol{\theta}$. Solving step 2 is explained in Section 3.8.2. We perform this two-step optimization *alternately* until the algorithm converges and there are no significant changes in the residual errors. This method is related to the Expectation-Maximization (EM) algorithm and the Concave-Convex Procedure (CCCP) [57]. Both solve a problem by splitting it up and solving easier subproblems.

3.8.1. Iterative Closest Point

ICP is an algorithm that iteratively finds the optimal transformation that establishes correspondence between two datasets, assuming a coarse alignment. Equation 3.21 and 3.22 use the point to plane distance which is known to have better convergence speed [43, 47]. The ICP algorithm essentially solves two problems: First it finds correspondences and second it finds the optimal transformation between two datasets. In each iteration the registration error decreases and converges to a local minimum [58]. We directly employ point-to-plane constrained ICP for faster convergence and better results [47].

$$\textbf{Step 1.1:} \qquad \arg\min_{Y} \sum_{i=0}^{n} ((\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{y}) \cdot \boldsymbol{n}_i)^2 \tag{3.21}$$

$$\textbf{Step 1.2:} \qquad \arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{i=0}^{n} ((\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{y}) \cdot \boldsymbol{n}_i)^2 \tag{3.22}$$

The correspondences in Equation 3.21 are computed by finding the closest points $\boldsymbol{y}_i \in \mathcal{Y}$ to $\boldsymbol{x}_i$. Finding correspondences can be sped up by efficient data structures, *e.g.* KD-Tree [8, 12, 58]. Equation 3.22 computes the optimal transformation between $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_i \in \mathcal{Y}$. $\boldsymbol{n}_i$ is the normal of point $\boldsymbol{y}_i$.

Step 2.2 is efficiently solved by linearizing the rotation matrix. The rigid transformation is then parametrized by the Euler angles α, β, γ and translation $\boldsymbol{t}$, stacked in a vector $\boldsymbol{c}$.

The solution to the rigid pose estimation in Equation 3.22 is given by setting the derivatives zero. We refer the reader to the work of Chen et al. [15] and Low [34] for a comprehensive derivation of the derivatives. We briefly state how the rigid-transformation is recovered. For better readability, we introduce $\boldsymbol{p}_i = \boldsymbol{x}_i \times \boldsymbol{n}_i$, where $\times$ denotes the

cross product. After rearrangement, we write our problem using matrices

$$\boldsymbol{A} = \sum_i \begin{pmatrix} p_{i,x}p_{i,x} & p_{i,x}p_{i,y} & p_{i,x}p_{i,z} & p_{i,x}n_{i,x} & p_{i,x}n_{i,y} & p_{i,x}n_{i,z} \\ p_{i,y}p_{i,x} & p_{i,y}p_{i,y} & p_{i,y}p_{i,z} & p_{i,y}n_{i,x} & p_{i,y}n_{i,y} & p_{i,y}n_{i,z} \\ p_{i,z}p_{i,x} & p_{i,z}p_{i,y} & p_{i,z}p_{i,z} & p_{i,z}n_{i,x} & p_{i,z}n_{i,y} & p_{i,z}n_{i,z} \\ n_{i,x}p_{i,x} & n_{i,x}p_{i,y} & n_{i,x}p_{i,z} & n_{i,x}n_{i,x} & n_{i,x}n_{i,y} & n_{i,x}n_{i,z} \\ n_{i,y}p_{i,x} & n_{i,y}p_{i,y} & n_{i,y}p_{i,z} & n_{i,y}n_{i,x} & n_{i,y}n_{i,y} & n_{i,y}n_{i,z} \\ n_{i,z}p_{i,x} & n_{i,z}p_{i,y} & n_{i,z}p_{i,z} & n_{i,z}n_{i,x} & n_{i,z}n_{i,y} & n_{i,z}n_{i,z} \end{pmatrix}, \tag{3.23}$$

$$\boldsymbol{c} = \begin{pmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{pmatrix}^\intercal, \tag{3.24}$$

$$\boldsymbol{b} = \sum_i \begin{pmatrix} p_{i,x}(x_i - y_i) \cdot \boldsymbol{n}_i \\ p_{i,y}(x_i - y_i) \cdot \boldsymbol{n}_i \\ p_{i,z}(x_i - y_i) \cdot \boldsymbol{n}_i \\ n_{i,x}(x_i - y_i) \cdot \boldsymbol{n}_i \\ n_{i,y}(x_i - y_i) \cdot \boldsymbol{n}_i \\ n_{i,z}(x_i - y_i) \cdot \boldsymbol{n}_i \end{pmatrix}. \tag{3.25}$$

Matrix $\boldsymbol{A} \in \mathbb{R}^{6\times6}$ and coefficient vector $\boldsymbol{c} \in \mathbb{R}^6$ and depending vector $\boldsymbol{b} \in \mathbb{R}^6$ constitute the following linear system:

$$\underset{\alpha,\beta,\gamma,t_x,t_y,t_z}{\arg\min} = \boldsymbol{A}\boldsymbol{c} - \boldsymbol{b} = 0 \tag{3.26}$$

As $\boldsymbol{A}$ is a positive-definite symmetric matrix, Equation 3.26 can be efficiently solved by a Cholesky decomposition. Using Cholesky factorization we write

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^*, \tag{3.27}$$

where $\boldsymbol{L}$ is a lower triangular matrix with real and positive diagonal entries and $\boldsymbol{L}^*$ its conjugate transpose. Solving

$$\boldsymbol{L}\boldsymbol{q} = \boldsymbol{b} \tag{3.28}$$

by forward substitution and

$$\boldsymbol{L}^*\boldsymbol{c} = \boldsymbol{q} \tag{3.29}$$

by backward substitution, efficiently calculates the solution for $\boldsymbol{c}$. Fixing the computed rigid-transformation in Equation 3.20, we continue computing the model coefficients $\boldsymbol{\theta}$.

3.8.2. Linear Model Fitting

Step 1 in Equation 3.19 computes the rigid transformation aligning the model with the sample. Step 2 in Equation 3.20 computes the model coefficients. ($\boldsymbol{\mu}$ is a $3n$ vector and $\boldsymbol{U}$ a $3n \times m$ matrix. To simplify the parameter computation, we change the direction of the rigid transformation and rewrite step 2

$$\arg\min_{\boldsymbol{\theta}} \sum_i \left((\boldsymbol{R}(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta}) + \boldsymbol{t} - \boldsymbol{v}_i) \cdot \boldsymbol{n}_i \right)^2$$

as

$$\arg\min_{\boldsymbol{\theta}} \sum_i \left((\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta} + \boldsymbol{R}^\intercal \boldsymbol{t} - \boldsymbol{R}^\intercal \boldsymbol{v}_i) \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i \right)^2 \tag{3.30}$$

Notice that $\boldsymbol{R}$ is an orthogonal matrix, therefore $\boldsymbol{R}^\intercal \boldsymbol{R} = \boldsymbol{I}$. Further rearrangement leads to

$$\arg\min_{\boldsymbol{\theta}} \sum_i \left(\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta} \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i + \boldsymbol{R}^\intercal \boldsymbol{t} \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i - \boldsymbol{R}^\intercal \boldsymbol{v}_i \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i \right)^2 \tag{3.31}$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_i \left(\boldsymbol{U}_i \operatorname{diag}(\sigma)\, \boldsymbol{\theta} \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i + \underbrace{\boldsymbol{\mu}_i + \boldsymbol{R}^\intercal \boldsymbol{t} \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i - \boldsymbol{R}^\intercal \boldsymbol{v}_i \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i}_{b_i} \right)^2 . \tag{3.32}$$

$$\tag{3.33}$$

We can apply $R^\intercal \boldsymbol{n}_i$ directly on $\boldsymbol{U}$, before we evaluate the model, therefore, we write

$$= \arg\min_{\boldsymbol{\theta}} \sum_i \left(\underbrace{\boldsymbol{U}_i \operatorname{diag}(\sigma) \cdot \boldsymbol{R}^\intercal \boldsymbol{n}_i}_{\boldsymbol{A}_i}\, \boldsymbol{\theta} - b_i \right)^2 . \tag{3.34}$$

Collecting the terms in matrix notation, results in a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{b} \in \mathbb{R}^n$. In matrix notation we state the final linear equation system as

$$= \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}\|_2^2. \tag{3.35}$$

The solution to this linear system is computed by

$$\frac{\partial \|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}\|_2^2}{\partial \boldsymbol{\theta}} = 0 \tag{3.36}$$

$$= (\boldsymbol{A}^\intercal \boldsymbol{A})^{-1} \boldsymbol{A}\boldsymbol{b} = \boldsymbol{\theta} \tag{3.37}$$

Equation 3.37 is a least-squares problem, which is sensitive to outliers and bad correspondences. See Section 3.7 for a more detailed explanation of the inherent problematic. To stabilize the optimization and increase the robustness, we employ the same simple correspondence rejection scheme as in Section 3.7.

The previous section illustrates one of the key contribution of this work. We presented an ICP-style formulation for fitting the model which computes the optimal model parameters by *alternately* recovering the optimal transformation between sample and model and fitting the model parameters. We derived and stated the solution as two subproblems that can be solved as linear least-squares problems. Another notable fact is that the model is fitted in a point-to-plane fashion.

Chapter Summary

This chapter describes the core part of this work. We start by outlining a high level overview in Section 3.1. That section presents the basic fitting method, starting from the input and resulting model parameters.

The fitting can be embedded into a non-rigid registration to acquire more robust correspondences. Therefore we extended the NICP registration scheme (see Section 2.2.1) with methods for correspondence weighting and rejection and present them in Section 3.4. Recovering the model parameters is accomplished by minimizing objectives function: Section 3.5 presents an existing point-to-point cost function from the literature. We modify this cost function to minimize a point-to-plane error in Section 3.6.

In Section 3.7 we develop an iterative framework embedding the point-to-plane based cost function from Section 3.6. We linearize this method for faster computation in Section 4.6. Linearizing the rigid pose estimation and linear model fitting obsoletes the non-linear least-squares optimization.

4. Experiments and Results

Previous sections described theoretical background and developed methods. We present this work's results in the following sections. To provide a fair evaluation with real world data, we collected a small dataset of facial scans.

Section 4.1 describes and motivates the selected evaluation methods. All methods in this work minimize a point-to-point and/or point-to-plane error. For better understanding of the fitting quality, we evaluate a third error measure: We propose evaluating using the mean angular error in Section 4.1. The pipeline parameters used for evaluation are stated in Section 4.2.

The dataset itself and the capturing modalities are presented in Section 4.3.

For all evaluations we exclude the 10 % most distant correspondences and all border points. The intention is to achieve a fairer evaluation for samples with missing regions, *e.g.* ears or parts of the nose. We use millimeters as length unit in all measurements.

4.1. Evaluation Method

We perform a qualitative evaluation of the proposed methods. Furthermore we want to interpret the results and analyze them in detail.

For this purpose a good error measure has to be found. A good error measure for 3DMM facial fits is an ongoing research issue. Many publications perform an empirical evaluation by visual inspection of the fitted models. We also provide quantitative results for better comparison.

The most obvious choice is the squared sum over the Euclidean distance of all n corresponding vertices

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{n} \sum_{k=0}^{n-1} \|\boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k}\|^2, \tag{4.1}$$

where $\boldsymbol{x}_{i,k}$ and $\boldsymbol{x}_{j,k}$ are corresponding vertices of the two surfaces. Equation 4.1 is the point-to-point distance. A similar error metric is the point-to-plane error (See Section 2.5.2), given by the squared sum between a point on the source surface and the tangent plane at its corresponding point.

Standard error measures like point-to-point or point-to-plane do not give a good statement about the accuracy of a fit. Using *e.g.* mean squared error (MSE) with these kind of error measures does not give any information about a good facial reconstruction with a realistic curvature.

To tackle this issue we follow [13] and [38] and use the MSE of the difference of the surface normals of two corresponding vertices. The rationale is that two similar surfaces should have rather similar surface normals at corresponding points. This error metric gives evidence about the similarity of the overall curvature of the reconstructed face. Knothe evaluated a similar work with a mean curvature distance metric and stated that the results were similar to the normal distance [30].

Hence we define the following error measure, following [38]. i, j are two meshes, *vn* is a function that returns the normalized surface normal of vertex k in mesh i. The parameter for *arccos* is given by Equation 3.1, assuming the normal vectors are normalized, hence the denominator might be omitted.

$$\mathrm{MSE}(i,j) = \frac{1}{n}\sum_{k=0}^{n} \arccos(\mathrm{vn}(i,k) \cdot \mathrm{vn}(j,k))^2 \tag{4.2}$$

Furthermore, it is a neutral measure, because it is not minimized in the cost function.

Summarizing, we present for every method the mean point-to-point, point-to-plane and angular error and discuss the results. We begin the actual result presentation by stating the used pipeline parameters.

4.2. Pipeline Parameters

The proposed pipeline has three components where parameters can be adjusted. Namely: *Rough alignment*, *Registration* and *Fitting*. The *Rough alignment* has to be performed manually in the current implementation.

First, we state the parameters for the *Registration* in Section 4.2.1. Secondly, the parameters for the model fitting are presented in Section 4.2.2.

4.2.1. Registration

The *Registration* (see Section 2.2.1) has the most adjustable parameters. We tuned them empirically.

More stiffness steps than necessary do not influence the registration quality, they just increase the computational time [3]. We empirically evaluated that $|\boldsymbol{\alpha}| = 5$ is enough for a good registration result.

As minimum stiffness we use 1×10^1 and as maximum 1×10^3. It is lowered when the $\|\boldsymbol{X}_j - \boldsymbol{X}_{j-1}\| < \epsilon$, where $\epsilon = 30$. The weighting parameter in the stiffness term is set $\gamma = 1$.

Correspondence Rejection

We reject all correspondences with a difference of the surface normals greater than 45°. The threshold is constant for all iterations, therefore a high threshold is necessary to prevent rejection of too many correspondences in the beginning.

Correspondence Weighting

We manually determined the parameters in a way that the registration result is both accurate and has minimal vertex topology changes. *max linear weight distance* = 20 mm and *max pair distance* = 30 mm. See Section 3.4.2 for details.

Cross-side Matching

Among the three evaluated approaches CSM-distance performs best. The other two approaches are too restrictive and reject too many correspondence. We empirically evaluated that 5 mm is a good distance tolerance value.

4.2.2. Model Fitting

The *Fitting* is mainly the minimization of the presented cost functions in Chapter 3. We use a downsampled face model with 2733 vertices.

Referring to Figure 2 we set $\boldsymbol{\theta} \in \mathbb{R}^{25}$. 25 principal components explain 95 % of the total variance of all training samples. We consider this as a good trade-off between fitting speed and explanatory power.

The only directly tunable parameter in the cost function is the regularization strength λ. We emprically evaluated that $\lambda = 10$ gives the fitting results.

4.3. Dataset

In order to provide evaluations using real world data, we scanned 10 persons, 8 male and 2 female. A *Creaform Go!SCAN* 3D Scanner was used. It is a portable handheld scanner using structured light. The age of the samples is between 20 and 30 years.

4.3.1. Scanning Procedure

Figure 7 shows the three steps. The person sits down and positions the head within a set of markers on the wall (Figure 7(a)). The markers are necessary for the scanner to perform reliable tracking. Next the person leans the head towards the wall, closes the eyes and remains as steady as possible.

The scanning itself takes about 3-5 minutes. The quality of the resulting scan greatly depends on how steady the person remains during the scan. The scanning itself is quite annoying for the person due to the continous blinking of the structured light pattern projector. Figure 7(b) shows that the scans are noisy and contain artifacts. Areas that are covered by hair result in holes in the final mesh.

Last the scans are manually cleaned and aligned (Figure 7(c)). We do no further postprocessing like cleaning holes or smoothing.

(a) Person placed between markers **(b)** Uncleaned scan with noise and artifacts **(c)** Scan aligned and cleaned from artifacts

Figure 7.: *Dataset Acquisition Procedure Example:* Three steps of the data acquisition. The most right step gives the input for the pipeline.

4.4. Non-linear Template Deformation Fitting

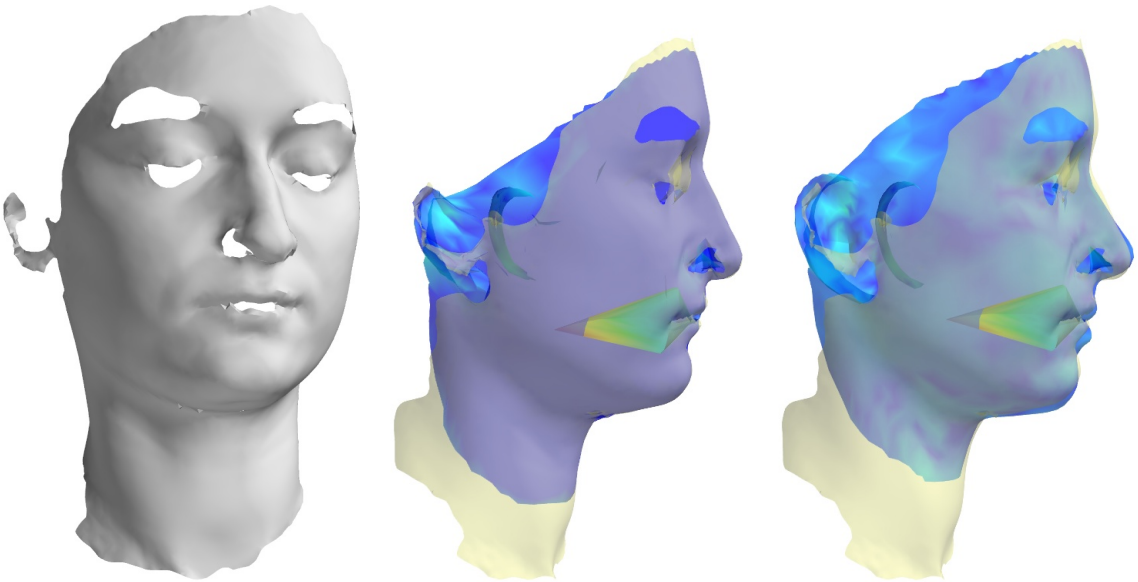
This Section presents the results of the non-linear template deformation fitting method. First, the model's mean template is non-rigidly deformed to the input mesh. Second, the deformed mean template is used as input for a non-linear fitting method that recovers

the model parameters. The reader is referred to Section 3.7 for more details about this method.

We discuss the fitting performance using a visual inspection in Section 4.4.1 and present quantitative results in Section 4.4.2.

4.4.1. Visual Evaluation

Figure 8 shows the fitting results for a sample from the dataset. Figure 8(a) is the input to the fitting method. Rough alignment was preformed manually before as a preprocessing step. The first step of this fitting method is a non-rigid deformation of the model's mean shape onto the sample.



(a) Input sample

(b) Model's mean template registered onto the sample

(c) Fitted model overlayed on sample

Figure 8.: *Non-linear template deformation fitting example:* Shows the input sample, the deformed mean template with deformed ears and the overlayed fitted model.

We see the result in Figure 8(b). It is clearly visible that the registration was successful and the mean template is closely registered with the sample. The non-rigid registration also deformed the ears, but due to bad correspondences, the ears were unnaturally deformed.

The last step is recovering the model parameters by minimizing a non-linear objective function that minimizes either a point-to-point or a point-to-plane error. Figure 8(c) shows the original sample from Figure 8(a) overlayed with the fitted model. Comparing

to Figure 8(b), the ears are modeled in a natural way and are no longer deformed. It is evident that the fitted model does not overlay as closely as the registered template. We think that the naturally deformed ears have strong negative influence on the least-squares optimization.

4.4.2. Quantitative Evaluation

Table 4.1 states the quantitative results of the template deformation fitting method. All evaluated point-to-plane errors are, as expected, lower than point-to-point errors. As Figure 8 suggests, the non-rigidly registered mean shapes, model the face much better than the model, but still have deformed ears.

Sample 7 has a lot higher errors than the other samples. This is mostly caused by the incompleteness of the scan, see Figure 9. As stated at the beginning of Chapter 4, we exclude 10 % of the most distant correspondences to allow a fairer evaluation. Although this does not completely remove the effect of holes in the scan. Hence the numerical results have to be considered with caution while comparing against each other. Very incomplete samples have many unsuitable correspondences, even after exclusion of the worst 10 %. Hence the calculated means can not be seen as final result of the fitting performance on their own.

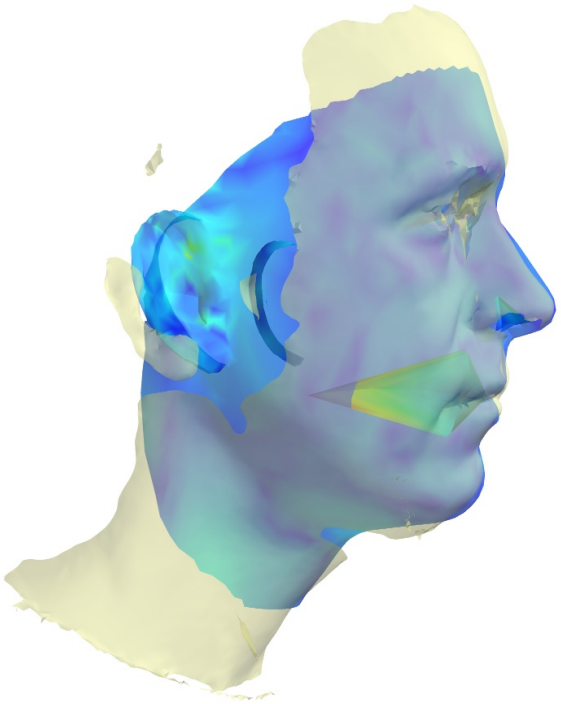


Figure 9.: Fitted model overlayed on sample 7

4.5. Non-linear Iterative Fitting

This method can be seen as natural evolution of the previous method. Previously we non-rigidly deformed the model's mean shape and then performed a single fitting step.

The driving idea of this method is that there are better correspondences in each iteration available. This encourages us to fit the model in an iterative way and acquire a new set of correspondences in each iteration. See 3.7 for a detailed description of this method.

Test #	R: MAR [°]	R: Pt2Pt [mm]	R: Pt2Pl [mm]	F: MAR [°]	F: Pt2Pt [mm]	F: Pt2Pl [mm]
0	21.67	0.93	0.54	29.04	6.83	3.69
1	18.44	0.20	0.10	27.91	6.77	4.59
2	20.28	0.26	0.13	30.26	7.73	4.82
3	17.98	0.42	0.26	26.01	4.35	2.71
4	23.24	0.45	0.25	30.79	9.24	5.21
5	17.02	0.40	0.22	25.91	5.62	3.07
6	19.86	0.33	0.16	28.05	10.97	6.49
7	27.67	11.18	5.83	34.92	33.49	15.45
8	19.61	0.45	0.24	27.10	9.40	4.64
9	19.15	0.41	0.21	28.50	7.60	4.70
mean	20.49	1.50	0.79	28.85	10.20	5.54
stddev	2.93	3.23	1.68	2.53	7.97	3.46
median	19.74	0.42	0.23	28.28	7.66	4.67

Table 4.1.: *Test results template deformation fitting:* **R** stands for registration, **F** for fitting, Pt2Pl is the Point-to-Plane error metric, Pt2Pt the Point-to-Point and MAR is the mean angular error between the corresponding surface normals.

Table 4.2 shows the results for our dataset. The bad scores of Sample 7 have the same reasons as for the previous method: The input mesh is so noisy that even rejecting 10 % of the worst correspondences does not allow a fair comparison.

It is evident that all error scores in Table 4.2, are much lower than in Table 4.1 which states the results of the template deformation fitting method. These results demonstrate that fitting the model iteratively, leverages iteratively found correspondences. Looking at the point-to-plane error, non-linear iterative fitting gives about 15 % better results. The improvement for point-to-point and mean angular errors is comparable.

We omit pictures of the fitting results, because by means of visual inspection, no strong difference between the two presented methods is visible.

The two previous methods used a cost function to minimize a point-to-plane constrained error. As the used cost function can be separated into several independent steps, we reformulated and linearized the cost function in Section 3.8. We evaluate this method in the following section.

4.6. Linear Iterative Fitting

Previously presented fitting methods required a non-linear optimization. This means that the cost function has to be linearized in every iteration and solved.

Test #	F: MAR [°]	F: Pt2Pt [mm]	F: Pt2Pl [mm]
0	29.52	7.31	3.47
1	26.17	6.90	3.72
2	29.23	6.52	2.85
3	26.22	4.39	2.53
4	30.00	10.18	4.64
5	25.05	6.53	2.85
6	28.67	23.39	7.56
7	35.01	30.52	11.90
8	25.72	10.56	3.95
9	26.57	6.28	3.29
mean	28.22	11.26	4.68
stddev	2.82	8.19	2.77
median	27.62	7.10	3.59

Table 4.2.: *Test results non-linear iterative fitting:* **F** for fitting, Pt2Pl is the Point-to-Plane error metric, Pt2Pt the Point-to-Point and MAR is the mean angular error between the corresponding surface normals.

Section 3.8 presented a fitting method that achieves the fitting without any non-linear optimization. This is achieved by reformulating the cost function and linearizing the non-linear parts. We optimize it by *alternatively* solving every step.

Figure 10 shows the model fitting and rigid pose estimation point-to-plane MSE over iterations. The plot shows that our fitting method converges iteratively to a local minimum. In the beginning the rigid pose estimations from ICP change the error a lot, as the model is not correctly registered with the model. Later iterations show hardly any difference between the model fitting and pose estimation error. The optimal rigid transformation is already recovered and decreasing errors can only be achieved by a better model fit. The error curve is not monotone decreasing. Refitting to a new correspondence set in every iteration and assuming 10 % as noise by default breaks the monotony.

The results for this method are stated in Table 4.3. The mean errors are comparable with the non-linear iterative fitting method. The median errors for point-to-point and point-to-plane distance are slightly lower. The mean is higher, because the linear fitting method converges to a bad local minimum for Sample 6.

Figure 11 clarifies that. It shows the difference of the point-to-plane errors of the linear iterative fitting method and the non-linear iterative fitting method. For the majority of tests the MSEs are lower, significantly lower for Sample 7.

For better understanding why the fit for Sample 6 failed, we perform a visual inspection of the input mesh and the fitting result. Figure 12(a) shows the input sample for Test 6. It is evident that the scan is quite incomplete and has many holes. Especially the ears

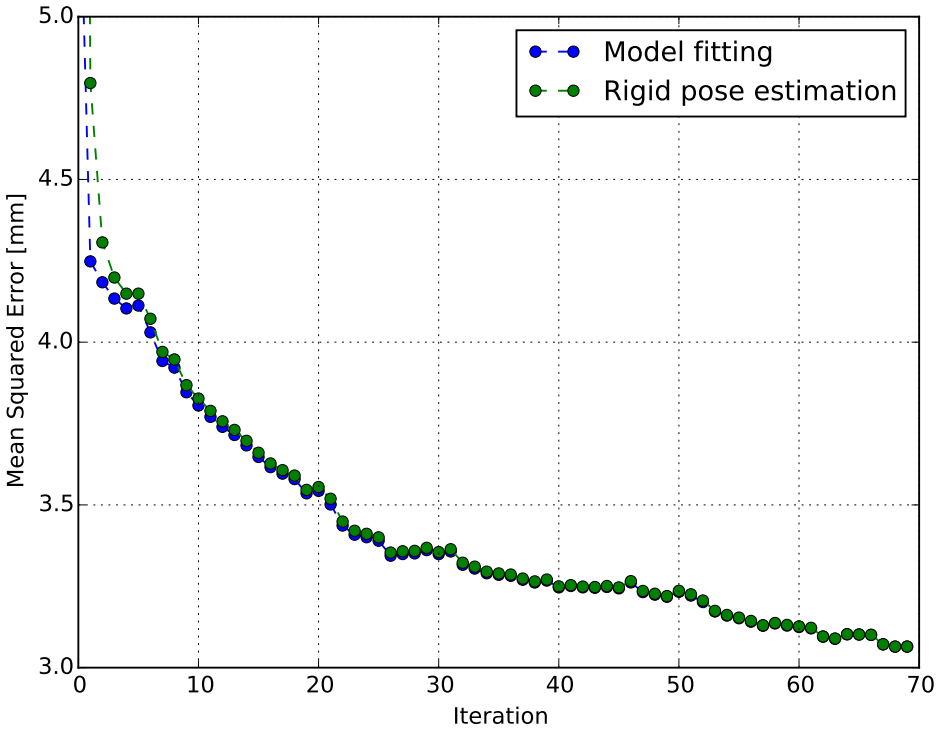


Figure 10.: *Point-to-plane Errors over iterations:* The algorithm converges iteratively to a local minimum. Increasing errors occur from fitting the model to new correspondences.

are nearly missing. Figure 12(b) visualizes the fitting result. The fitting process finished with a degenerated model. The figure's colormap visualizes the point-to-plane error.

The colorization indicates that the biggest errors are in the ear and mouth regions. Figure 13 shows a boxplot of the first ten model parameters. According to Section 2.4, we want to keep the squared sum of the model parameters small, as this increases the probability of getting a realistic result. As the linear iterative fitting method is currently not regularized, the solution can easily be an unrealistic face. This greatly depends on the quality of correspondences and has a higher sensitivity to outliers. Figure 13 shows that many parameters are higher than three times the standard deviation of the particular principal component. See 2.1 for more details about the structure of a 3DMM. Possible solutions to this problem are discussed in Section 5.3.

Test #	F: MAR [°]	F: Pt2Pt [mm]	F: Pt2Pl [mm]
0	31.22	8.32	3.71
1	28.13	5.82	3.41
2	30.93	6.34	2.74
3	26.36	4.38	2.62
4	31.55	10.66	4.45
5	25.68	5.78	2.54
6	37.38	28.14	10.28
7	39.23	18.48	10.41
8	27.33	10.48	4.00
9	27.91	5.68	3.03
mean	30.57	10.41	4.72
stddev	4.34	7.08	2.87
median	29.53	7.33	3.56

Table 4.3.: *Test results linear iterative fitting:* Pt2Pl is the Point-to-Plane error metric, Pt2Pt the Point-to-Point and MAR is the mean angular error between the corresponding surface normals.

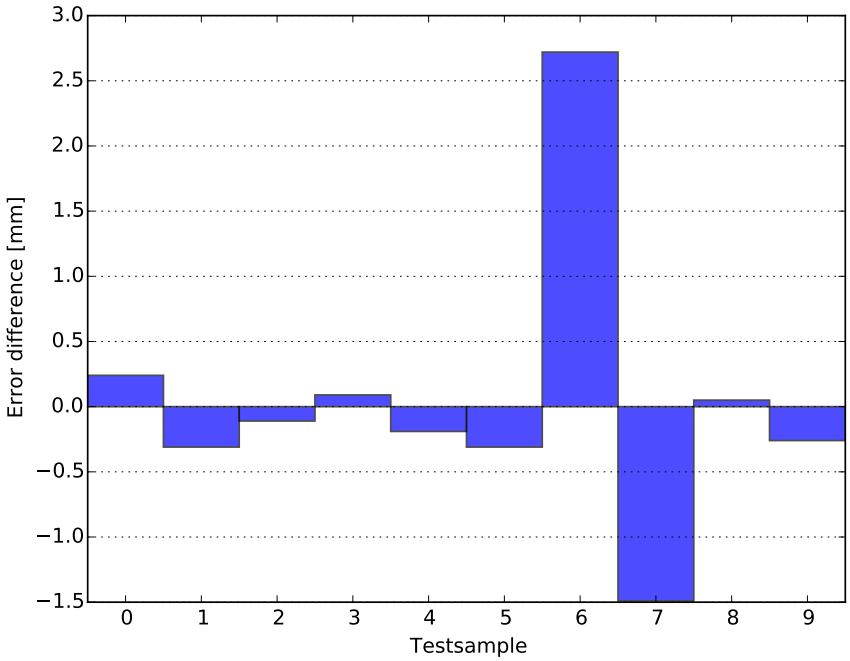
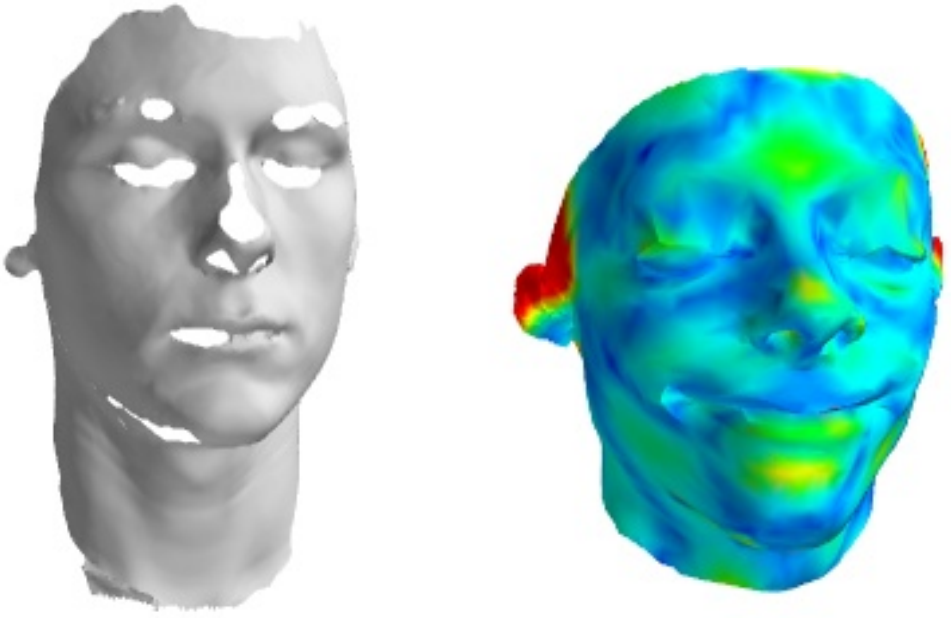


Figure 11.: *Error difference linear iterative fitting and non-linear iterative:* The mean squared errors of the linear iterative fitting method are for the majority of samples lower, but much stronger on Sample 6.



(a) Input sample

(b) Fitted model, colormap is the point-to-plane error

Figure 12.: *Iterative linear fitting Sample 6*: The fitted model is degenerated and shows high errors for the ears.

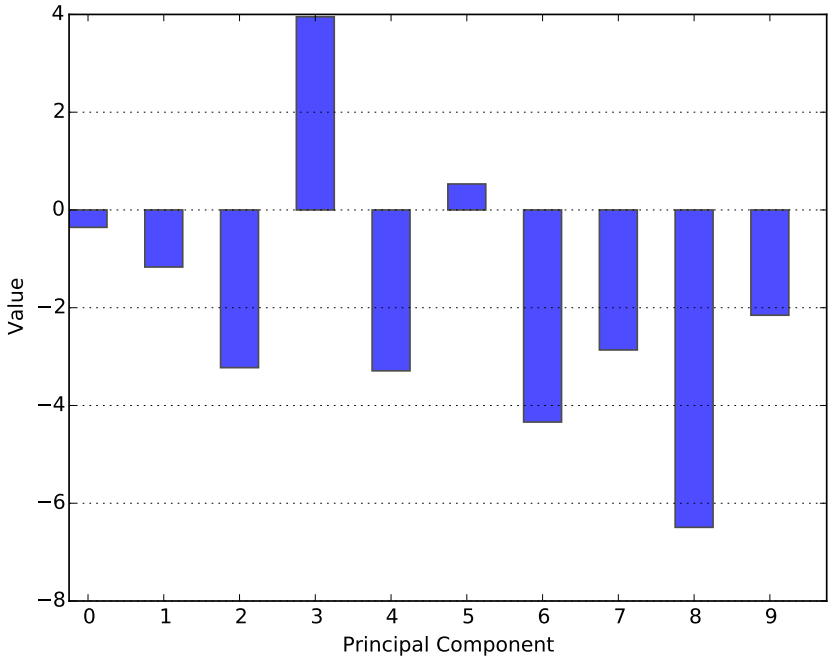


Figure 13.: *Boxplot of the first ten model parameters of Sample 6:* The high values suggest unrealistic parameters.

5. Discussion

This chapter concludes this thesis by a discussion of the results and findings. In the previous chapter we presented the results in visual and quantitative ways. Following we interpret and discuss the results. Section 5.1 discusses the results of the registration and their implications. Section 5.2 discusses the fitting results, especially the difficulties of selecting a good error measure. Furthermore we describe future work in Section 5.3 and conclude the thesis with Section 5.4.

5.1. Discussion of Registration Results

The registration is a necessary prerequisite for the fitting method presented in Section 3.3. It provides the fitting process with the correspondences. To achieve this the 3DMM's mean face is non-rigidly deformed to the sample.

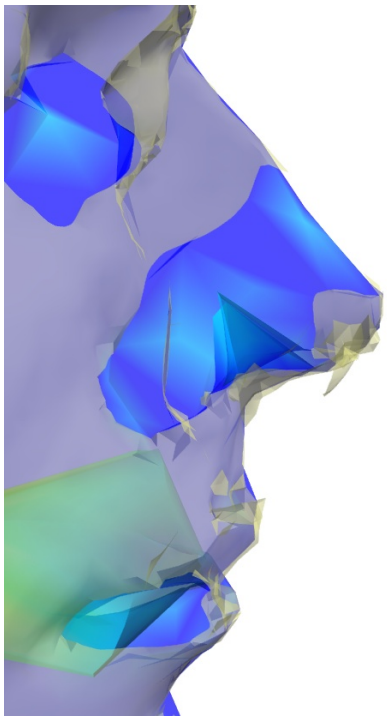


Figure 14.: *Registered scan with artifacts:* Mean face non-rigidly registered to an incomplete scan with several artifacts.

5.1.1. Robustness of the Registration

Several figures within this work showed that the scanned faces are quite incomplete and contain artifacts. Figure 14 shows a zoomed in version of sample 2. It is clearly visible that the scan is very incomplete in the nose area and contains artifacts. Nevertheless the registered template captures the shape of the face very accurately and fills in missing regions.

Despite the robustness, the registration is done by minimizing a point-to-point cost function. We discuss the issue of this in Section 5.1.2.

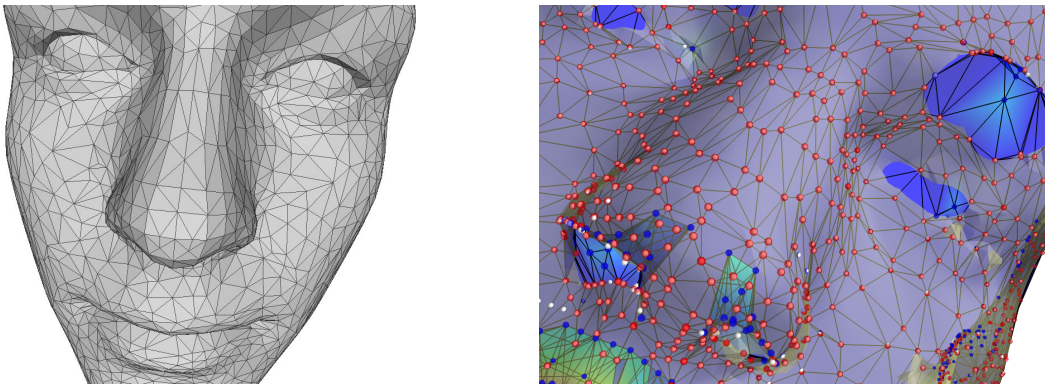
5.1.2. Caveat of Minimizing a Point-to-Point Error

Table 4.1 states the quantitative results of the registration. The point-to-point, point-to-plane and angular error are very low after the registration and would suggest a good registration result. We discovered that low error rates do not represent another factor that describes the quality of a registration: The deformation of the mesh topology itself is not represented by these measures. Following we shortly discuss reasons for this. Potential solutions are presented in the future work section.

NICP-A minimizes the point-to-point error between the model's mean face and the target surface. A non-rigid registration changes the mesh topology to some extent, but it should preserve the mesh topology as well as possible.

Optimizing for a point-to-point error directly minimizes the distance between two corresponding vertices. This leads to problems when having meshes with different densities.

Figure 15 shows an example of the previously mentioned mesh topology deformation. Figure 15(b) shows the deformed template and Figure 15(a) a clipping of the model's mean face. The mesh topology in Figure 15(a) is symmetric and smooth. Figure 15(b) shows a clipping of the mean face registered to a sample. The topology changed in a non-smooth way. The influence of mesh topology deformations on the fitting result is subject to further investigations.



(a) *Mesh topology of the model's mean face:* Shows the template before registration. The vertice topology is quite uniform.

(b) *Strong point clustering* after the registration in the model's mean face.

Figure 15.: *Unnaturally Deformed Mesh:* Influence of the registration to the mesh topology of the model's mean face.

Without detailed analysis we still state the following assumption: The purpose of the registration process is to find correspondences. The template gets deformed in a non-rigid fashion to accomplish this. When the registration is performed on a sparse target mesh, multiple template vertices get deformed to the same target point. This results in a clustering or even congruence of multiple template vertices towards a single target point.

Moving along to the actual fitting, the outcome is that biased correspondences are supplied to the fitting process.

5.1.3. Unnatural Deformations of Missing Regions

NICP-A uses the template's mesh topology regularization. In iterations with low stiffness weights, unnatural deformations of regions with wrong correspondences might happen.

Figure 16 shows an example of this. The sample's ear contains large missing regions and the template's ear got badly deformed towards the remaining parts of the sample's ear and parts of the cheek.

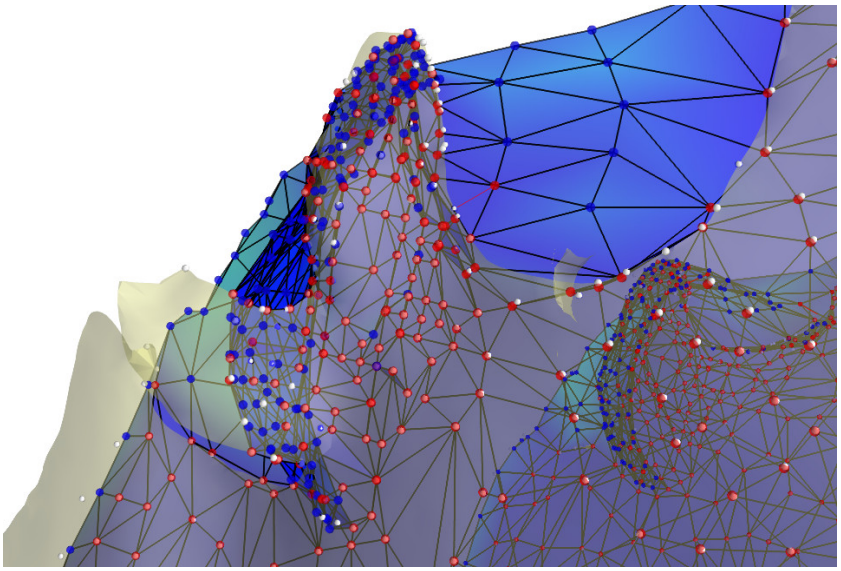


Figure 16.: *Unnaturally deformed ear due to wrong correspondences:* The blue mesh is the registered template, light gray the sample. The red points have correspondences, the blue points were rejected.

The red dots are correspondences with a non-zero weight for the registration. The blue dots are excluded from the current iteration. The white dots are corresponding vertices on the sample. Figure 16 also shows that many template vertices have been transformed in a way that made them congruent with sample vertices. This issue is discussed in detail in Section 5.1.2. Registration results like this are avoided as much as possible by the correspondence rejecting and weighting scheme, explained in Section 3.4.1 and 3.4.2.

5.2. Discussion of Fitting Results

Following we discuss the robustness of the fitting and the influence of different error metrics in the cost function.

5.2.1. Robustness of the Fitting

A non-convex optimization runs the risk of getting stuck in local minima. Good initial guesses of the parameters reduce this risk greatly.

We initialize the model coefficients $\boldsymbol{\theta} \in \mathbb{R}^m$ randomly using a standard normal distribution. This is reasonable, because the construction of the 3DMM can be interpreted as fitting multivariate Gaussian distribution (see Section 2.1). Translation and rotation is initialized with zero. We empirically evaluated that these are suitable starting conditions.

5.2.2. Results in Terms of Different Error Measures

In Section 4.1 we discussed the difficulties of finding a good error metric for facial model fitting. In this Section we discuss by suitable examples and state the influence of different factors to them.

We want explain different error measures on the results of sample 1, which is the most complete scan in our dataset. Figure 17 shows sample 1 overlayed with the fitted model.

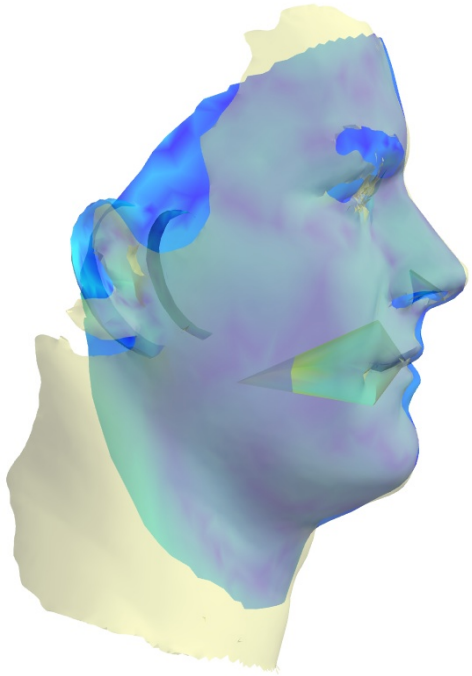
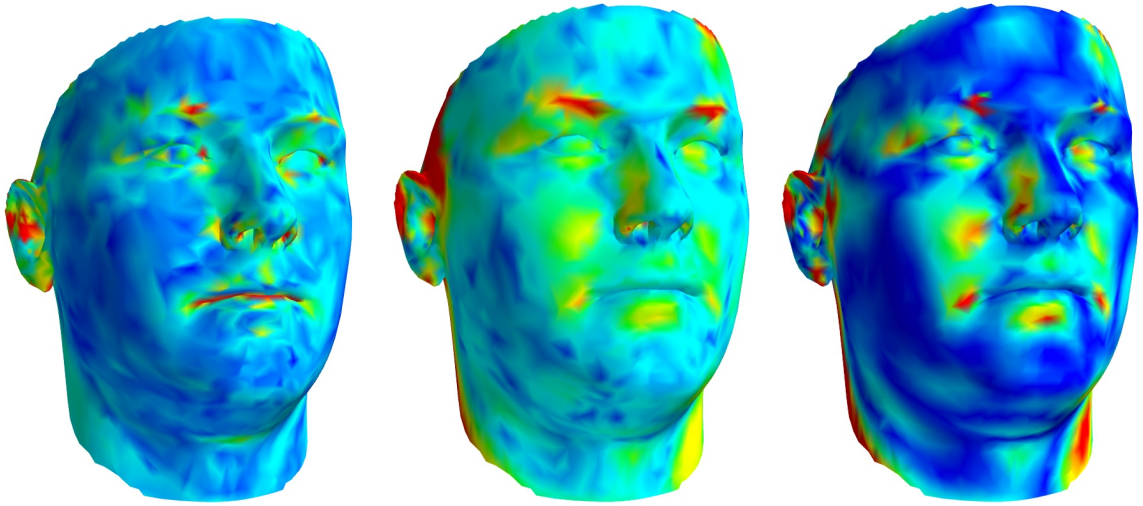


Figure 17.: *Sample 1 overlayed with fitted model*: The lips could not be modeled accurately.

Figure 18 illustrates the errors as colormap on the fitted model. Following we discuss the informations that each of the error measures gives.



(a) Angular Error **(b)** Point-to-point Error **(c)** Point-to-plane Error

Figure 18.: *"Error faces" for different error measures:* The angular error informs about the surface similarity between fit and input mesh. Point-to-point and point-to-plane are rather similar and show the distance to the closest point.

The mean angular error in Figure 18(a) does not directly express the distance of two corresponding points. This is clearly visible in the mouth and forehead area of Sample 1. Despite the fact that there is clearly a bigger distance between the sample and the fitted model, the surface curvature is similar. The aim of the angular error is to give a statement about the similarity of the curvature. The angular difference does not reflect minor translations of a connected area. Not only the similarity of the surface, but also its position defines the accuracy of a fit. The red areas in the "angular error face" are areas that contain the highest 15 % of angular errors.

The point-to-point (Figure 18(b)) and point-to-plane (Figure 18(c)) distance overcome the issue of the angular error measure, but do not indicate the similarity of the curvature. Looking at the images, the area under the lips is now clearly marked as erroneous. We see that the error in the point-to-point "error face" is significantly higher than in the point-to-plane images. The meaning of the red area is the same as for the angular error images.

Concluding we believe that none of the evaluated error measures on its own gives a strong enough statement about the fitting quality of a face. We think that a combination of the angular error and the point-to-plane error measure gives a better judgment.

5.2.3. Weighted Least-Squares Optimization

The accuracy of the fitting result greatly depends on the quality of the correspondence. Our cost functions minimize a least-square error, which is sensitive to outliers.

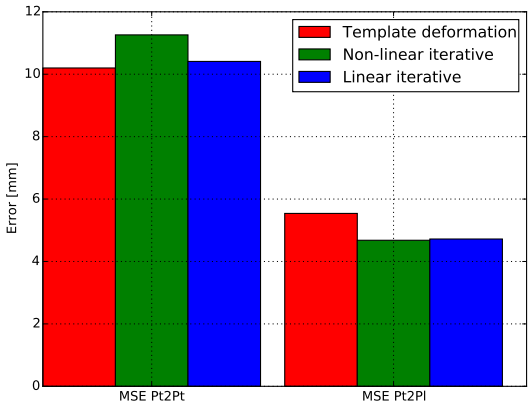
Experiments showed that correspondences weighting is not necessary for the template deformation based fitting method described in Section 3.3. This is not surprising, as the correspondences for this method are given by the non-rigidly registered model mean shape.

The non-linear iterative and linear iterative fitting methods presented in Section 3.7, respectively Section 3.8 require correspondence weighting. We observed that these methods fail without it and return degenerated fits.

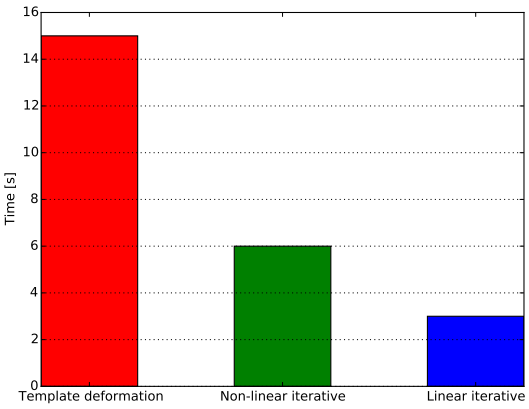
Others used *robust functions* to increase the robustness of ICP [20, 35, 53, 58]. These methods can be employed to increase the robustness of our fitting methods. Another approach can be the usage of sparsity inducing norms [12].

5.2.4. Comparing the Fitting Methods

We presented three different fitting methods in this work. The first method (Section 3.3) acquires correspondences by non-rigid registration [2]. A 3DMM is fitted by performing a non-linear least-squares optimization. The second method (Section 3.7) iteratively acquires new correspondences and refits the model, using the method as above. As third method (Section 3.8) we solve the rigid-pose estimation and model fitting in a fully linear way.



(a) *Error comparison*: The point-to-plane error of our methods is about 18 % lower than of the original method.



(b) *Time comparison:* Our proposed iterative methods are up to 5 times faster than the original method.

Figure 19.: *Time and error comparison of the evaluated methods:* Our proposed iterative methods are both more accurate and faster.

Figure 19(a) compares the point-to-point and point-to-plane errors of our proposed methods with the original method. Our iterative methods perform about 18 % better in terms of point-to-plane error than the original method from [2]. The curvature of the

resulting models does not differ significantly between the different methods. Therefore the mean angular error is similar for all methods.

The original method, requiring a non-rigid registration, need about 15 seconds to fit a model. The non-rigid registration takes the most time of the process. Method 2 takes about 6 s and method 3 about 3 seconds for fully fitting a model. Figure 19(b) shows a plot of the timings. Currently, the framework is implemented in Python. We think that using a faster implementation, method 3 can reach real-time speed. Especially faster correspondence search and better convergence checks for the algorithm can give a big speedup. Fitting methods 2 and 3 are at risk of fitting a degenerated model, due to the lack of a regularization. We stated this issue in Section 4.6 and present solutions in Section 5.3.2 as future work.

5.3. Future Work

In the previous two sections we discussed results and findings of this thesis. Consecutively we state and motivate potential future work.

5.3.1. Automated Rough Alignment

To achieve a fully automated model fitting, an adequate rough alignment has to be provided. In the current pipeline version this step has to be done manually. For automating this step, many efficient methods exist.

If texture data is available as well, it is possible to run a fast 2D facial landmark detection on it. The detected landmarks can be projected into 3D space and used to estimate a transformation to align the sample with the model's mean face. Running the 2D facial landmark detection on several frames and then estimate the transformation in a least-squares manner increases the robustness.

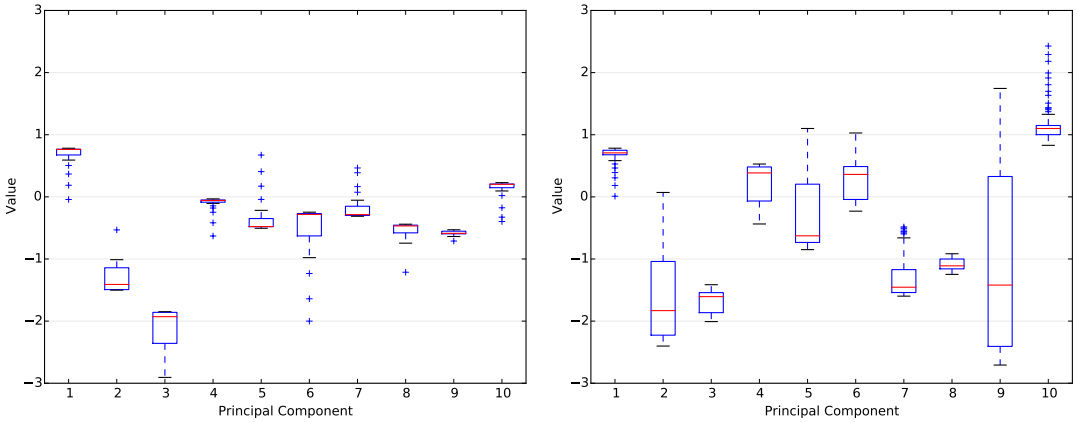
When having only range data, other approaches have to be pursued. The nose is a very distinctive part of a face. Algorithms for detection of the nose tip are quite robust and fast. Peng et al. [42] presented state of the art results in their recent publication. Their approach is accurate and the authors state that a real-time implementation could be possible.

5.3.2. Different Model Parameter Regularization

Not every k-sized parameter vector $\boldsymbol{\theta} \in \mathbb{R}^k$ models a realistic face. The statistical properties of a 3DMM provide opportunities for parameter regularization (see Section 2.4). The fitting methods incorporating non-linear model parameter estimation use a parameter penalty term of the form $\lambda\|\boldsymbol{\theta}\|^2$. This is also called Tikhonov regularization or ridge

regression. It penalizes big parameters, which is desirable due to the model properties (see Section 2.1).

Figure 20(a) shows the final parameters of the non-linear iterative fitting method for Sample 6. The parameters values are quite small in comparison to Figure 20(b), which was fitted without regularization. Furthermore the parameters changed within a bigger interval than the regularized fitting.



(a) *Boxplot of the first ten model parameters of Sample 6 fitted with regularization:* The parameters are small and jumped less during the optimization.

(b) *Boxplot of the first ten model parameters of Sample 6 without regularization:* Fitted using the linear iterative method. The high values suggest unrealistic parameters. Strong parameter changes during the optimization.

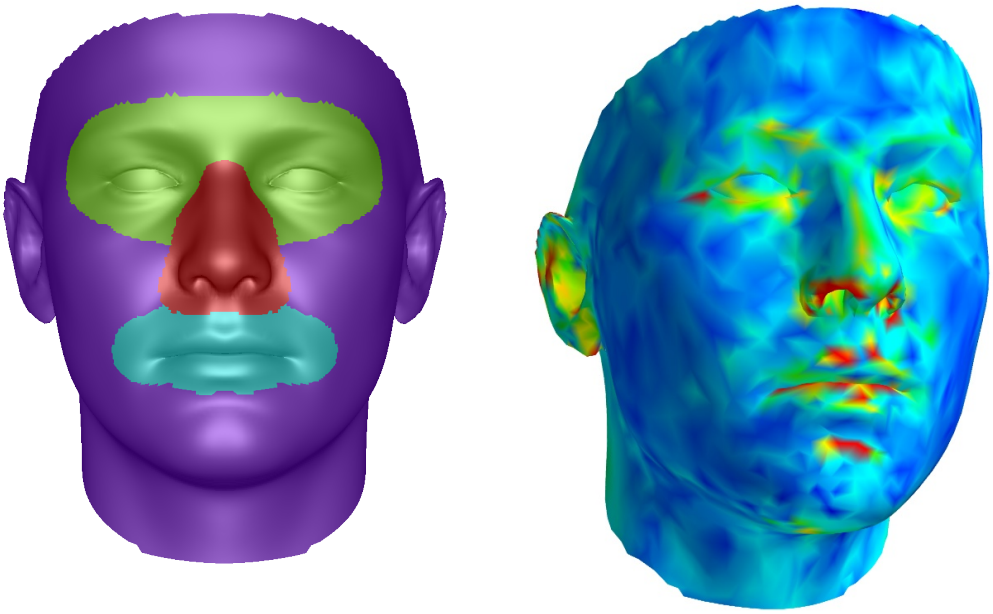
Figure 20.: *Comparison of the first ten parameters of two fitting methods:* The boxplots show the resulting parameters and the value range they had during the fitting.

The principal components $\boldsymbol{U}$ of the 3DMM are sorted by the variance in the dataset. Considering how the model describes faces, α_i with a lower component index i models stronger deformation. Section 2.1.4 showed that 25 components describe over 90 % of the model's variance. This means that high i just add more details. Intuitively we want a sparse parameter solution, meaning that a solution with as few components as possible should be found. Tikhonov regularization penalizes big values, but the parameters will still be non-zero. The *Lasso* regularization method adds a term $\lambda\|\boldsymbol{\theta}\|_1$ which causes more and more parameters driven to be zero [52].

Employing a *Lasso* regularization can increase the fitting robustness in presence of outliers. Additionally better results without changing the regularization method could be possible by the prior distribution presented in Equation 2.37.

5.3.3. Segmenting the Model

Figure 21(b) shows that nose, eyes and mouth areas have in comparison to other areas quite high errors. The 3DMM described in Section 2.1 has m degrees of freedom. The expressiveness of the model can be increased by dividing the face model into different subregions. Figure 21(a) shows the segmentation of the BFM. These subregions can be morphed individually. To achieve a good result, the transitions between the individually fitted regions have to be smooth. The authors of [49] successfully used the method from [50] to ensure a smooth segment region transition by an additional regularization.



(a) *Segmented Face Model [40]:* Segmentation increases the model's flexibility.

(b) *Angular error image sample:* High errors are visible in the nose, mouth and eyes area.

Figure 21.: *Increased flexibility by a segmented model:* Segmenting the model decouples the individual regions and enables individual fitting in regions with high errors.

5.3.4. Better Acquisition of Dense Correspondence

The quality of the dense point-to-point correspondence is a crucial factor for the fitting quality. In Section 5.1.2 we explained caveats of minimizing a point-to-point based cost function. We further discussed issues caused by unnatural template deformation in Section 5.1.3.

To tackle the first issue, a different registration method with a point-to-plane based cost function could be used, *e.g.* [33]. Furthermore the model fitting can be embedded into the registration algorithm as prior, like [11, 48].

Direct model fitting without acquiring dense point-to-point correspondence is also possible [6, 49]. The fitting performance is degraded when there are too many outliers in the target mesh. To address this problem, we would like to replace the one-to-one correspondence assumption by a one-to-many and fit for fuzzy correspondences [17, 22].

The second issue is closely related to the point-to-point based registration. A mesh simplification that preserves the symmetric properties [37] of the face model could be beneficial to improve this issue.

5.3.5. Benchmarking

Benchmarking and comparing different approaches and methods requires good ground truth data. The dataset (see Section 4.3) can be considered as noisy real world dataset. On the other hand it is not possible to conduct a fair comparison between the samples, because the quality of the scans varies a lot. For future evaluations it would be advantageous to use a publicly available dataset, like the UND database [14]. We were unable to obtain one of the public face databases for this thesis.

5.4. Conclusion

We presented an extendable framework to fit 3DMMs for faces. We extended an existing non-linear cost function by point-to-plane constraints. Lower fitting errors are achieved by *alternatively* fitting the 3DMM and search for new correspondences. For faster computing, we linearized the cost function and embedded the model fitting in a point-to-plane ICP framework. A robust correspondence weighting and rejection scheme for an existing non-rigid registration algorithm was proposed. We carefully evaluated all used methods, stated issues and possible solutions to them.

Summarizing, the proposed fitting methods are 18 % more accurate and 5 times faster than the existing method [2].

A. Derivation of Jacobians

The Jacobian of a real valued function $F : \mathbb{R}^m \to \mathbb{R}^n$ is defined by the $n \times m$ matrix $\boldsymbol{J}$. The partial derivates of the component functions of F with respect to the variables $x_1, \ldots, x_m$ are organized as follows:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_m} \end{bmatrix} \tag{A.1}$$

A.1. Analytic Derivative Point-to-Point Error Function

In this section we give the derivation of the analytic Jacobian for the point-to-point fitting function. LM optimizes an objective function F in the following least-squares manner:

$$\min_{x \in \mathbb{R}^m} \|F(x)\|_2^2$$

Therefore we do not have to derive the least-squares function, just the inner part of the least-squares problem. We ignore the regularization part of the objective function for now to make the derivation simpler. F_i is the evaluation of f for the i-th vertex. Hence,

$$\begin{aligned} F_i &= \boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\boldsymbol{\alpha} + \boldsymbol{t}' - \boldsymbol{R}'\boldsymbol{v}_i \\ \boldsymbol{R}' &= \boldsymbol{R}^{-1} \qquad \boldsymbol{t}' = \boldsymbol{R}^{-1}\boldsymbol{t} \end{aligned} \tag{A.2}$$

$$\frac{\partial F_i}{\partial \boldsymbol{\alpha}} = \boldsymbol{U}_i \operatorname{diag}(\sigma) \qquad \frac{\partial F_i}{\partial \boldsymbol{t}'} = \boldsymbol{I}_3 \qquad \frac{\partial F_i}{\partial r_i} = \frac{\partial \boldsymbol{R}'_{r_1,r_2,r_3}}{\partial r_i}\boldsymbol{v}_i \tag{A.3}$$

The derivative of the regularization part is given by:

$$\frac{\partial \boldsymbol{\alpha}}{\partial(\boldsymbol{\alpha}, \boldsymbol{t}', r_i)} = \left[\begin{array}{cc|ccc} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{array}\right] \tag{A.4}$$

In equation A.3 we see directly that only the derivatives of the rotation change in every iteration. We might use this to split up the Jacobian into the following static and dynamic part. We rewrite the above in matrix formulation and add the derivation of the regularization part at the correct positions.

$$\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{U}\,\mathrm{diag}(\sigma) & \mathbf{1} \otimes \boldsymbol{I}_3 \\ \boldsymbol{I} & \mathbf{0} \end{bmatrix} \tag{A.5}$$

$$\boldsymbol{J}_d = \begin{bmatrix} \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}^\intercal & \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}^\intercal & \boldsymbol{I} \otimes \frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}^\intercal \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A.6}$$

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_c & \boldsymbol{J}_d \end{bmatrix} \tag{A.7}$$

The full Jacobian has the dimensions $(3n + m) \times (m + 6)$, where m denotes the number of used PCA dimensions and n the number of vertices. The static part $\boldsymbol{J}_c$ has to be calculated only once.

Detailled Jacobian Derivation

The following equations state the derivation of the Jacobian. F_i is the evaluation of f for the i-th vertex. m denotes the number of used principal components (see Section 2.1.1) and n the number of vertices.

The Jacobian $\boldsymbol{J}$ has the dimensions $(3n + m) \times (m + 6)$ and is derived as follows:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial F_1}{\partial \boldsymbol{\theta}} & \frac{\partial F_1}{\partial \boldsymbol{t'}} & \frac{\partial F_1}{\partial r_i} \\ \vdots & \vdots & \vdots \\ \frac{\partial F_n}{\partial \boldsymbol{\theta}} & \frac{\partial F_n}{\partial \boldsymbol{t'}} & \frac{\partial F_n}{\partial r_j} \\ \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{t'}} & \frac{\partial \boldsymbol{\theta}}{\partial r_i} \end{bmatrix} \tag{A.8}$$

$$= \begin{bmatrix} \frac{\partial F_{1_x}}{\partial \theta_1} & \cdots & \frac{\partial F_{1_x}}{\partial \theta_m} & \frac{\partial F_{1_x}}{\partial t_x} & \frac{\partial F_{1_x}}{\partial t_y} & \frac{\partial F_{1_x}}{\partial t_z} & \frac{\partial F_{1_x}}{\partial r_1} & \frac{\partial F_{1_x}}{\partial r_2} & \frac{\partial F_{1_x}}{\partial r_3} \\ \frac{\partial F_{1_y}}{\partial \theta_1} & \cdots & \frac{\partial F_{1_y}}{\partial \theta_m} & \frac{\partial F_{1_y}}{\partial t_x} & \frac{\partial F_{1_y}}{\partial t_y} & \frac{\partial F_{1_y}}{\partial t_z} & \frac{\partial F_{1_y}}{\partial r_1} & \frac{\partial F_{1_y}}{\partial r_2} & \frac{\partial F_{1_y}}{\partial r_3} \\ \frac{\partial F_{1_z}}{\partial \theta_1} & \cdots & \frac{\partial F_{1_z}}{\partial \theta_m} & \frac{\partial F_{1_z}}{\partial t_x} & \frac{\partial F_{1_z}}{\partial t_y} & \frac{\partial F_{1_z}}{\partial t_z} & \frac{\partial F_{1_z}}{\partial r_1} & \frac{\partial F_{1_z}}{\partial r_2} & \frac{\partial F_{1_z}}{\partial r_3} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F_{n_x}}{\partial \theta_1} & \cdots & \frac{\partial F_{n_x}}{\partial \theta_m} & \frac{\partial F_{n_x}}{\partial t_x} & \frac{\partial F_{n_x}}{\partial t_y} & \frac{\partial F_{n_x}}{\partial t_z} & \frac{\partial F_{n_x}}{\partial r_1} & \frac{\partial F_{n_x}}{\partial r_2} & \frac{\partial F_{n_x}}{\partial r_3} \\ \frac{\partial F_{n_z}}{\partial \theta_1} & \cdots & \frac{\partial F_{n_y}}{\partial \theta_m} & \frac{\partial F_{n_y}}{\partial t_x} & \frac{\partial F_{n_y}}{\partial t_y} & \frac{\partial F_{n_y}}{\partial t_z} & \frac{\partial F_{n_y}}{\partial r_1} & \frac{\partial F_{n_y}}{\partial r_2} & \frac{\partial F_{n_y}}{\partial r_3} \\ \frac{\partial F_{n_z}}{\partial \theta_1} & \cdots & \frac{\partial F_{n_z}}{\partial \theta_m} & \frac{\partial F_{n_z}}{\partial t_x} & \frac{\partial F_{n_z}}{\partial t_y} & \frac{\partial F_{n_z}}{\partial t_z} & \frac{\partial F_{n_z}}{\partial r_1} & \frac{\partial F_{n_z}}{\partial r_2} & \frac{\partial F_{n_z}}{\partial r_3} \\ \frac{\partial \theta_1}{\partial \theta_1} & \cdots & \frac{\partial \theta_1}{\partial \theta_m} & \frac{\partial \theta_1}{\partial t_x} & \frac{\partial \theta_1}{\partial t_y} & \frac{\partial \theta_1}{\partial t_z} & \frac{\partial \theta_1}{\partial r_1} & \frac{\partial \theta_1}{\partial r_2} & \frac{\partial \theta_1}{\partial r_3} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \theta_m}{\partial \theta_1} & \cdots & \frac{\partial \theta_m}{\partial \theta_m} & \frac{\partial \theta_m}{\partial t_x} & \frac{\partial \theta_m}{\partial t_y} & \frac{\partial \theta_m}{\partial t_z} & \frac{\partial \theta_m}{\partial r_1} & \frac{\partial \theta_m}{\partial r_2} & \frac{\partial \theta_m}{\partial r_3} \end{bmatrix} \tag{A.9}$$

$$= \begin{bmatrix} U_{11_x} & \cdots & U_{1m_x} & 1 & 0 & 0 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_1^\intercal\right)_x & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_1^\intercal\right)_x & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_1^\intercal\right)_x \\ U_{11_y} & \cdots & U_{1m_y} & 0 & 1 & 0 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_1^\intercal\right)_y & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_1^\intercal\right)_y & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_1^\intercal\right)_y \\ U_{11_z} & \cdots & U_{1m_z} & 0 & 0 & 1 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_1^\intercal\right)_z & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_1^\intercal\right)_z & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_1^\intercal\right)_z \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ U_{n1_x} & \cdots & U_{nm_x} & 1 & 0 & 0 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_n^\intercal\right)_x & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_n^\intercal\right)_x & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_n^\intercal\right)_x \\ U_{n1_y} & \cdots & U_{nm_y} & 0 & 1 & 0 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_n^\intercal\right)_y & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_n^\intercal\right)_y & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_n^\intercal\right)_y \\ U_{n1_z} & \cdots & U_{nm_z} & 0 & 0 & 1 & \left(\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}_n^\intercal\right)_z & \left(\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}_n^\intercal\right)_z & \left(\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{v}_n^\intercal\right)_z \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \tag{A.10}$$

$$= \begin{bmatrix} \boldsymbol{U} & 1 \otimes \boldsymbol{I}_3 & \frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{v}^\intercal & \frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{v}^\intercal & \frac{\partial \boldsymbol{R}'}{\partial r_3 3}\boldsymbol{v}^\intercal \\ \boldsymbol{I} & \boldsymbol{0} & 0 & 0 & 0 \end{bmatrix} \tag{A.11}$$

A.2. Analytic Derivative Point-to-Plane Error Function

There are only minor changes on the analytical derivatives. Hence the reader is referred to Section A.1 for a more detailed derivation.

We start again with the inner function F_i

$$\begin{aligned} F_i &= (\boldsymbol{\mu}_i + \boldsymbol{U}_i \operatorname{diag}(\sigma)\boldsymbol{\alpha} + \boldsymbol{t}' - \boldsymbol{R}'\boldsymbol{v}_i)\,\boldsymbol{n}_i \\ \boldsymbol{R}' &= \boldsymbol{R}^{-1} \qquad \boldsymbol{t}' = \boldsymbol{R}^{-1}\boldsymbol{t} \end{aligned} \tag{A.12}$$

and split up the derivatives into a static ($\boldsymbol{J}_c$) and dynamic part ($\boldsymbol{J}_d$).

$$\frac{\partial F_i}{\partial \boldsymbol{\alpha}} = \boldsymbol{M}_i \operatorname{diag}(\sigma)\boldsymbol{n}_i \qquad \frac{\partial F_i}{\partial \boldsymbol{t}'} = \boldsymbol{I}_3\boldsymbol{n}_i = \boldsymbol{n}_i \qquad \frac{\partial F_i}{\partial r_i} = \left(\frac{\partial \boldsymbol{R}'_{r_1,r_2,r_3}}{\partial r_i}\boldsymbol{v}_i\right)\boldsymbol{n}_i \tag{A.13}$$

$$\boldsymbol{J}_c = \begin{bmatrix} \operatorname{inner}_3\left(\boldsymbol{U}\operatorname{diag}(\sigma), \boldsymbol{N}\right) & \boldsymbol{N} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \tag{A.14}$$

$$\boldsymbol{J}_d = \begin{bmatrix} \operatorname{diag}\left(\boldsymbol{v}\frac{\partial \boldsymbol{R}'}{\partial r_1}\boldsymbol{N}\right) & \operatorname{diag}\left(\boldsymbol{v}\frac{\partial \boldsymbol{R}'}{\partial r_2}\boldsymbol{N}\right) & \operatorname{diag}\left(\boldsymbol{v}\frac{\partial \boldsymbol{R}'}{\partial r_3}\boldsymbol{N}\right) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{A.15}$$

The inner_3 operator is a special multiplication operator. For it's definition refer to Section A.2.1. The calculation of the component-wise inner product of two $3 \times n$ matrices is formulated as computing the dot product and extracting the diagonal. The diagonal

represents the component-wise dot product. We derived this method by modifying the Kronecker product.

The reader might notice that it is rather inefficient to compute a component-wise inner product like this. We implemented this efficiently using numpy.

Detailled Jacobian Derivation

The derivation of the Jacobian for Point-to-Plane metric follows the same scheme as Point-to-Plane.

The main difference is that the resulting Jacobian matrix is a $(n+m) \times (m+4)$ matrix. The dot product between the vertices and the normals "reduces" a vertex to a scalar, therefore reduces the number of rows of the Jacobian by a factor of 3. Furthermore

$$\frac{\partial F_i}{\partial \boldsymbol{t}'} = \boldsymbol{I}_3 \boldsymbol{n}_i = \boldsymbol{n}_i$$

causes a reduction by two columns.

All other derivation steps are analogous to Point-to-Point metric.

A.2.1. Inner3 Operator

We define the operator inner_3 as follows:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1_x} & \dots & a_{1,m_x} \\ a_{1,1_y} & \dots & a_{1,m_y} \\ a_{1,1_z} & \dots & a_{1,m_z} \\ \vdots & \ddots & \vdots \\ a_{n,1_x} & \dots & a_{n,m_x} \\ a_{n,1_y} & \dots & a_{n,m_y} \\ a_{n,1_z} & \dots & a_{n,m_z} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} b_{1_x} & b_{1_y} & b_{1_z} \\ \vdots & \vdots & \vdots \\ b_{n_x} & b_{n_y} & b_{n_z} \end{bmatrix}$$

$$\underset{3}{\text{inner}}(\boldsymbol{A}, \boldsymbol{B}) = \begin{bmatrix} s_{1,1} & \dots & s_{1,m} \\ \vdots & \ddots & \vdots \\ s_{n,1} & \dots & s_{n,m} \end{bmatrix}$$

$$\begin{aligned} s_{1,1} &= a_{1,1_x} b_{1_x} + a_{1,1_y} b_{1_y} + a_{1,1_z} b_{1_z} \\ s_{1,m} &= a_{1,m_x} b_{1_x} + a_{1,m_y} b_{1_y} + a_{1,m_z} b_{1_z} \\ s_{n,1} &= a_{n,1_x} b_{n_x} + a_{n,1_y} b_{n_y} + a_{n,1_z} b_{n_z} \\ s_{n,m} &= a_{n,m_x} b_{n_x} + a_{n,m_y} b_{n_y} + a_{n,m_z} b_{n_z} \end{aligned}$$

Acronyms

3DMM 3D Morphable Model.

AAM Active Appearance Model.
ASM Active Shape Model.

BFM Basel Face Model.

CCCP Concave-Convex Procedure.
CSM cross-side matching.

EM Expectation-Maximization.

ICIA Inverse Compositional Image Alignment.
ICP Iterative Closest Point.

LiST Linear Shape and Texture.
LM Levenberg-Marquardt.

MAP maximum a posteriori probability.
MFF Multi-Feature Fitting.
MSE mean squared error.

NICP non-rigid ICP.

PCA Principal Component Analysis.
PDF Probability Density Function.

RA-3DMM Resolution-Aware 3DMM.

SNO Stochastic Newton Optimization.
SVD Singular Value Decomposition.

Bibliography

[1] O. Aldrian and W. Smith. "Inverse rendering of faces with a 3D Morphable Model." In: *PAMI* (2013).

[2] B. Amberg, R. Knothe, and T. Vetter. "Expression invariant 3D face recognition with a Morphable Model." In: *FG*. 2008.

[3] B. Amberg, S. Romdhani, and T. Vetter. "Optimal step nonrigid ICP algorithms for surface registration." In: *CVPR*. 2007.

[4] B. Amberg. "Editing faces in videos." PhD thesis. University of Basel, 2011.

[5] D. Anguelov et al. "SCAPE: Shape Completion and Animation of People." In: *SIGGRAPH*. 2005.

[6] C. Arellano and R. Dahyot. "Robust bayesian fitting of 3D Morphable Model." In: *CVMP*. 2013.

[7] R. Beichel et al. "Robust Active Appearance Models and their application to medical image analysis." In: *MI* (2005).

[8] P. Besl and N. D. McKay. "A method for registration of 3-D shapes." In: *PAMI* (1992).

[9] V. Blanz, K. Scherbaum, and H.-P. Seidel. "Fitting a Morphable Model to 3D scans of faces." In: *ICCV*. 2007.

[10] V. Blanz and T. Vetter. "A Morphable Model for the synthesis of 3D faces." In: *SIGGRAPH*. 1999.

[11] S. Bouaziz and M. Pauly. "Dynamic 2D/3D Registration for the Kinect." In: *ACM SIGGRAPH Courses*. SIGGRAPH. 2013.

[12] S. Bouaziz, A. Tagliasacchi, and M. Pauly. "Sparse Iterative Closest Point." In: *Computer Graphics Forum (Symposium on Geometry Processing)* (2013).

[13] J. Bustard and M. Nixon. "3D Morphable Model construction for robust ear and face recognition." In: *CVPR*. 2010.

[14] K. I. Chang, K. W. Bowyer, and P. J. Flynn. "An evaluation of multimodal 2D+ 3D face biometrics." In: *PAMI* (2005).

[15] Y. Chen and G. Medioni. "Object modelling by registration of multiple range images." In: *IVC* (1992).

[16] H. Chui and A. Rangarajan. "A feature registration framework using mixture models." In: *MMBIA*. 2000.

[17] H. Chui and A. Rangarajan. "A new point matching algorithm for non-rigid registration." In: *CVIU* (2003).

[18] T. F. Cootes et al. "Active shape models-their training and application." In: *CVIU* (1995).

[19] G. J. Edwards, T. F. Cootes, and C. J. Taylor. "Face recognition using Active Appearance Models." In: *ECCV*. 1998.

[20] A. W. Fitzgibbon. "Robust registration of 2D and 3D point sets." In: *IIC* (2003).

[21] S. Flöry and M. Hofer. "Surface fitting and registration of point clouds using approximations of the unsigned distance function." In: *Computer Aided Geometric Design* 27.1 (2010), pp. 60–77.

[22] S. Granger and X. Pennec. "Multi-scale EM-ICP: A fast and robust approach for surface registration." In: *ECCV*. 2002.

[23] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009.

[24] G. Hu et al. "A facial symmetry prior for improved illumination fitting of 3D Morphable Model." In: *ICB*. 2013.

[25] G. Hu et al. "Resolution-aware 3D Morphable Model." In: *BVMC*. 2012.

[26] I. T. Jolliffe. *Principal Component Analysis*. 2nd ed. Springer, 2002.

[27] O. van Kaick et al. "A Survey on Shape Correspondence." In: *CGF* (2011).

[28] B.-N. Kang, H. Byun, and D. Kim. "Multi-resolution 3D Morphable Models and its matching method." In: *ICPR*. 2008.

[29] M. Kirby and L. Sirovich. "Application of the Karhunen-Loeve procedure for the characterization of human faces." In: *PAMI* (1990).

[30] R. Knothe. "A global-to-local model for the representation of human faces." PhD thesis. University of Basel, 2009.

[31] H. Li, T. Weise, and M. Pauly. "Example-based facial rigging." In: *SIGGRAPH)* (2010).

[32] H. Li et al. "Realtime facial animation with on-the-fly correctives." In: *SIGGRAPH Asia* (2013).

[33] H. Li et al. "Robust single-view geometry and motion reconstruction." In: *SIGGRAPH Asia* (2009).

[34] K.-L. Low. "Linear least-squares optimization for point-to-plane icp surface registration." In: *Chapel Hill, University of North Carolina* (2004).

[35] T. Masuda and N. Yokoya. "A robust method for registration and segmentation of multiple range images." In: *CVIU* (1995).

[36] S. C. Mitchell et al. "3-D Active Appearance Models: segmentation of cardiac MR and ultrasound images." In: *MI* (2002).

[37] A. Patel and W. A. P. Smith. "Simplification of 3D Morphable Models." In: *ICCV*. 2011.

[38] A. Patel and W. A. P. Smith. "3D morphable face models revisited." In: *CVPR*. 2009.

[39] P. Paysan. "Statistical modeling of facial aging based on 3D scans." PhD thesis. University of Basel, 2010.

[40] P. Paysan et al. "A 3D face model for pose and illumination invariant face recognition." In: *AVSS*. 2009.

[41] P. Paysan et al. "Face reconstruction from skull shapes and physical attributes." In: *DAGM*. 2009.

[42] X. Peng, M. Bennamoun, and A. S. Mian. "A training-free nose tip detection method from face range images." In: *Pattern Recognition* (2011).

[43] H. Pottmann et al. "Geometry and convergence analysis of algorithms for registration of 3D shapes." In: *IJCV* (2006).

[44] S. Romdhani and T. Vetter. "Efficient, robust and accurate fitting of a 3D Morphable Model." In: *ICCV*. 2003.

[45] S. Romdhani. "Face image analysis using a multiple features fitting strategy." PhD thesis. University of Basel, 2005.

[46] S. Romdhani, V. Blanz, and T. Vetter. "Face identification by fitting a 3D Morphable Model using linear shape and texture error functions." In: *ECCV*. 2002.

[47] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm." In: *3DIM*. 2001.

[48] D. C. Schneider and P. Eisert. "Algorithms for automatic and robust registration of 3D head scans." In: *JVRB* (2010).

[49] D. C. Schneider and P. Eisert. "Fitting a Morphable Model to pose and shape of a point cloud." In: *VMV*. 2009.

[50] D. Schneider and P. Eisert. "Fast nonrigid mesh registration with a data-driven deformation prior." In: *ICCV*. 2009.

[51] R. W. Sumner, J. Schmid, and M. Pauly. "Embedded deformation for shape manipulation." In: *SIGGRAPH*. 2007.

[52] R. Tibshirani. "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society* (1996).

[53] E. Trucco, A. Fusiello, and V. Roberto. "Robust motion and correspondence of noisy 3-D point sets with missing data." In: *Pattern Recognition Letters* (1999).

[54] M. Turk and A. Pentland. "Eigenfaces for recognition." In: *Journal of cognitive neuroscience* (1991).

[55] T. Weise et al. "Realtime performance-based facial animation." In: *SIGGRAPH* (2011).

[56] A. Weiss, D. Hirshberg, and M. Black. "Home 3D body scans from noisy image and range data." In: *ICCV*. 2011.

[57] A. L. Yuille and A. Rangarajan. "The concave-convex procedure." In: *Neural Computation* 15.4 (2003).

[58] Z. Zhang. "Iterative point matching for registration of free-form curves and surfaces." In: *IJCV* (1994).

[59] M. Zollhöfer et al. "Automatic reconstruction of personalized avatars from 3D face scans." In: *CASA* (2011).