Graz University of Technology

Institute for Computer Graphics and Vision

Master's Thesis

3D Morphable Models

Christoph Gratl

Graz, Austria, August 2012

Thesis supervisor

Prof. Dr. Horst Bischof

Thesis advisor

Dr. Martin Urschler

Abstract

In photographs, the three-dimensional world is reduced to a two-dimensional image. In this master's thesis, we try to restore some of this lost information for the special case of human faces. Using the prior knowledge of human heads incorporated in a 3D Morphable Face Model, it is possible to recover the shape and texture of the displayed face as well as its pose and lighting in the photograph up to a certain extent. First we show the building process of this 3D Morphable Model. The base for the model is formed by a large set of unregistered human 3D head scans. With the Non-Rigid ICP algorithm, the headscans are brought into correspondence. The result is a high-dimensional Face Space with a lot of redundancy. On this Face Space a Principal Component Analysis is applied to reduce dimensionality and extract the more significant base vectors. As outcome we get a statistical three-dimensional face model that can describe human faces in an elegant and compact way by only a few parameters. To find the parameters of the 3D Morphable Face Model for a given face in a photograph, an Analysis-by-Synthesis algorithm is used. Starting from a reasonable point (e.g. the mean face), the model coefficients are adjusted incrementally. The randomized residual between input and ajdusted model as well as the derivative of the residual influence those adjustments. The algorithm stops when the projection of the 3D model matches the face on the input photograph close enough and returns the model and rendering parameters. The whole framework described in this master's thesis is evaluated both with regard to pose estimation and 3D face reconstruction on several image databases.

Keywords. 3D Morphable Model, Principal Component Analysis, 3D reconstruction

Kurzfassung

Fotos reduzieren die dreidimensionale Welt auf ein zweidimensionales Bild. In dieser Diplomarbeit wird für den speziellen Fall von Gesichtern versucht, einige dieser verlorenen Informationen wiederherzustellen. Indem das Wissen über das menschliche Antlitz in ein 3D Morphable Face Model integriert wird, ist es möglich, sowohl die Form und Farbe des abgebildeten Gesichts als auch seine Ausrichtung und die Beleuchtung im Bild bis zu einem gewissen Grad zu rekonstruieren. Als Erstes wird die Erstellung des 3D Morphable Models beschrieben. Als Basis für das Modell dient eine große Datenbank von unregistrierten Scans von menschlichen Köpfen. Der Non-Rigid ICP Algorithmus stellt die Korrespondenz zwischen diesen Scans her. Das Resultat ist ein hochdimensionaler Gesichtsraum, der noch viel Redundanz beinhaltet. Auf diesen Gesichtsraum wird eine Hauptkomponentenanalyse durchgeführt, um die wichtigen Basisvektoren zu extrahieren und die Dimensionalität zu verringern. Das Ergebnis ist ein statistisches dreidimensionales Gesichtsmodell, das mit nur wenigen Parametern menschliche Gesichter kompakt und elegant beschreiben kann. Um die Parameter des 3D Morphable Face Models zu einem Gesicht in einem Foto zu bestimmen, wird ein Analyse-durch-Synthese-Algorithmus verwendet. Ausgehend von einem sinnvollen Startpunkt (z.B. dem Mittelwertgesicht) werden die Modellkoeffizienten inkrementell angepasst. Die randomisierte Differenz zwischen Foto und angepasstem Modell und deren Ableitung dienen als Richtungsgeber für diese Anpassungen. Der iterative Vorgang stoppt, sobald die Projektion des 3D-Modells dem Gesicht am Foto ähnlich genug ist, und der Algorithmus liefert die Modell- und Darstellungsparameter als Resultat. Auf mehreren Gesichtsdatenbanken wird das in dieser Diplomarbeit beschriebene Framework dann sowohl in Bezug auf Pose Estimation als auch auf 3D Gesichtsrekonstruktion evaluiert.

Schlagwörter. 3D Morphable Model, Hauptkomponentenanalyse, 3D Rekonstruktion

Acknowledgments

First and foremost, I want to express my deep gratitude to my parents. They supported me under all circumstances and gave me the opportunity to do my studies at University.

Many thanks go to Prof. Horst Bischof, who supervised this master's thesis with great patience and also initiated the contact to Prof. Stan Z. Li.

Dr. Martin Urschler guided me throughout this work. He always had a sympathetic ear for my questions and provided helpful answers and interesting discussions. Martin also deserves credit for the proof-reading of this thesis. And I would like to extend my thanks to Markus Storer and all the other members of the Institute of Computer Graphics and Vision.

The members in the team of Prof. Stan Z. Li and he himself offered me a warm welcome to the Center for Biometrics and Security Research and I appreciate their hospitality and support during my time in Beijing a lot.

Last but not least I am very grateful for all the friends that accompanied me all along my way.

Statutory Declaration

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 marked all material which has been quoted either literally or by content from the used sources.

Graz, Austria, August 2012 (signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, Austria, August 2012 (Unterschrift)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Photographs capture an image of the reality that is reduced to two dimensions. While humans can easily reconstruct the lost information of the third dimension in their mind [53], [32], to recover an objects three-dimensional shape from a two-dimensional image is a classical problem in computer vision [38], [34], [62]. This information of the third dimension is of great value for an abundant number of tasks in computer vision. One special area that has been of interest to researchers is the human face. It is a non-trivial three dimensional shape with which we humans interact often and easily, but for computers it poses several challenges.

A first common task concerning human heads is the estimation of their pose. The orientation of a face transports much information in inter-personal communication, and humans absorb this information with ease. But for a computer algorithm, having available only the two dimensions from a photograph, the changing shape of the projected head, varying illumination and different identities make it difficult to determine the correct pose of a human head [56].

Even harder it is to identify and to recognize a person seen from different viewpoints and under varying illumination, if the computer algorithm can use only the two-dimensional information of the photograph [31]. In contrast to that, the three-dimensional representation of the persons face is invariant to changes in light and camera position. Although there are still many factors that impede those tasks (facial expression, occlusions, ...), both identification and recognition as well as the pose estimation can be facilitated if an exact 3D model of a persons head is available.

With the three-dimensional information at hand, it is also easy to produce virtual photographs of a person by rendering the 3D head to an image under arbitrary poses and

with freely configurable illumination conditions. This can be used to generate normalized pictures that are for example suitable for passports [39] or can be easier compared by traditional 2D face recognition algorithms [63]. Another important area where those virtual renderings can be of help is the editing of images that contain human faces. With this underlying shape, adding a hat for example with correctly rendered shadows and occlusions becomes trivial compared to the traditional image editing process. Furthermore, using computer animation, realistic movies with human faces can be created on the basis of a 3D head model.

Research in computer vision has opened many possibilities to recover the shape from two-dimensional photographs. The most obvious way is to use two or more images of the same object that are taken under different viewpoints [6], [23]. This stereoscopic reconstruction searches for corresponding points in the images and uses them to compute the missing information of the third dimension. But also the blur that results from focus depth gives cues to infer the full shape of a pictured object [44].

Shape from shading is part of the technique that is used in this work to recover 3D information from a 2D image. Horn [34] first described how to recover an objects normals (and therefore its shape) from its shading (the perceived color in the 2D image) within some constraints (one point light source, smooth opaque object, Lambertian illumination model, ...). But for realistic images it is difficult to find unique solutions, and many directions have been explored starting from seminal work of Horn [80], [24]. Related to this method is photometric stereo, where illuminating the object with a light from different directions also allows to infer its normals.

To derive the three-dimensional information from an image, humans use the knowledge of already seen objects of the same category. Also in computer vision, a database of examples can provide the basis for an algorithm to perform tasks and decisions. Active Appearance Models (AAMs) [25] use a set annotated images of a specific category to match an unseen instance from the same type. The knowledge is captured in a model that linearly combines and compresses the example images, which guides the fitting of a new image. Blanz and Vetter introduced the 3D Morphable Model [13], which extends this idea to the third dimension. They use a number of 3D laser scans as the knowledge basis for their model.

This master's thesis is based on the seminal work of Blanz and Vetter [13]. The goal was to build a 3D Morphable Model of the human face and develop a framework for its testing and evaluation. One major contribution of this work is a 3D Morphable Model

that has been created from a large database of head scans. As the optical flow method to establish correspondence between the raw head scans did not function well for our database, an Optimal Step Nonrigid ICP algorithm [3] is used to perform the registration of the database examples. The implementation of the complete framework (from building the model to image fitting) is another major part of this work. Finally, the model is examined carefully and evaluated on a number of datasets both for pose estimation and shape and texture estimation.

The master's thesis is grouped into four main chapters. The broader topic of Statistical Shape and Appearance Models and their areas of application are covered in chapter 2. The creation of a 3D Morphable Face Model from a database of human head scans is shown in detail in the next chapter 3. In chapter 4 we describe the algorithm to adjust the parameters of this constructed model to a given two-dimensional human face image. The last major chapter 5 addresses the evaluation of the built model and the fitting framework. Experiments are conducted for pose estimation and for shape and texture estimation on artificial and real datasets. Also the implementation is described in this part. An outlook as well as a summary are given in the final chapter 6.

Chapter 2

Statistical Shape and Appearance Models

Contents

Computer vision often has to deal with objects that, although pertaining to a common category, have great variability in both shape and appearance. This wide range is not only characteristic for categories that are produced by nature (trees, dogs, faces, mountains, clouds, ...) but also for classes of man-made objects (i.e. cars, buildings, cogwheels, etc.). This diversity introduces difficulties for computers when they try to recognize, detect, classify or segment objects in images. In contrast to that, humans easily recognize the objects and classify them into the corresponding category. Only a few examples are needed by the human visual system to learn a new category. Statistical shape and appearance models try to adopt this human behavior by describing a category through a combination of some of its elements.

In this chapter, first some roots of statistical template models are shown. Then two major approaches in computer vision using those models are presented. The Active Appearance Model is the most prominent representative and introduced in the first section. An overview of 3D Morphable Models is given in the next section, and the last part is

concerned with their areas of application.

2.1 History

The basic idea of statistical template models is to describe new members of a category as linear combinations of samples that are already known. A new face can for example be specified by

$$face_c = 0.2 * face_x + 0.8 * face_y \tag{2.1}$$

Using those prototypes it is possible to span a category vector space, and a category member is specified by its coordinates in this space. It allows for variability within a class and furthermore gives a compact representation. A prerequisite for this linear combination formulation of a class is that its samples are in a corresponding vectorized form.

This first naive approach has some drawbacks. Many base samples are needed to account for all the possible variation within a category, which results in a high dimensionality of the category vector space. They furthermore can introduce a lot of redundancy into the model. And third, it is hard to describe the intra-class variation consistently and efficiently.

Principal Component Analysis [59] [41] is a classical statistical method that is able to deal with all three of those mentioned problems. It gives a more compact and statistical meaningful representation of the category by projecting the original vector space to a new basis with associated probabilities and plausibilities. Kirby and Sirovich [43] first applied this method to the domain of human faces by simply vectorizing the training images and applying a PCA to them. Also Turk and Pentland [74] worked with PCA on faces. They called the resulting eigenvectors "eigenfaces" as they look like "a sort of ghostly face" , and used them to represent, detect and recognize faces in images. As the eigenfaces are directly calculated from the vectorized images, this approach is prone to changes in shape, pose and expression.

Those changes in shape were previously addressed by Kass et al. [42]. Their Active Contour Models are a mathematical formulation where on one hand the bending energy of a shape defined by splines is to be minimized. On the other hand, the formulation contains also a term that results in the shape adapting itself to image contours. From this behavior the method gets its common name "Snakes", as the contour actively snuggles into the image-based gradients as real snakes do with obstacles. However, as this model

does not incorporate any knowledge about the underlying object category.

Cootes et al. [21] incorporate this knowledge of category in their Active Shape Model. The basis of the knowledge is formed by a set of examples having hand-labeled feature-points. The rigid and scale deformations between those examples are removed by a Procrustes analysis and then a PCA is computed onto the feature points to generate a Point Distribution Model. In the paper they show models for a human hand, resistors and heart ventricles. To match the model to an image, first the strongest edges in the normal direction are determined for every model point. Then the global parameters (pose, scale) are adjusted so the distances between the model points and the image contours are as small as possible. Finally, also the shape parameters are adapted to deform the model towards the image edges. The disadvantage of this approach is that it incorporates only part of the gray-level information of the image - just the gradient information along a line in the neighborhood of the model points is used to fit the model to it.

Ezzat and Poggio [26] generate synthetic new views of a face from a set of example views by learning linear combinations of the inter-example-correspondences. Using analysis-by-synthesis, they also fit the model to an unseen view within a stochastic optimization procedure. However this works reliably only for the same person that the model has been generated with - it does not generalize to unseen faces.

This is addressed by Vetter and Poggio [75]. First the linear coefficients by which an unseen input image can be represented through a set of examples with the same orientation are determined. Those coefficients are then used to generate a synthetic view with a different desired orientation. Images of the same example objects but with the desired pose form the basis for the linear combination that finally generates a new synthetic view of a given input image.

The overview over some of the earlier statistical models shows already some of their properties: Every model is category-specific. A model trained from a set of examples that belong to a specific category can not be used for the analysis of objects from different categories. But as the model is trained for a special class, it pertains its specific characteristics while still allowing for intra-category variability. And once a model is established for a specific category, it usually has a quite good representational power for this class, whilst pertaining a very compact description for a particular instance: A few model parameters characterize a certain object in a very detailed way. Depending on the fitting algorithm, their calculation can be quite efficient and fast, whilst providing a good adoption to shape and texture.

2.2 Active Appearance Models

Active Appearance Models (short AAM) are the main result of an evolution of different deformable template models, and their development has mainly been driven by Cootes, Edwards and Taylor [20].

The basis of an AAM is formed by a set of m labeled training examples with sparse corresponding feature points x. Using the coordinates of those feature points in the image, a new shape can be generated by a parametrized model of the form.

$$x_{new} = M(b) \tag{2.2}$$

Instead of using a simple linear combination that has several drawbacks, a PCA is used to optimize the shape vector space. Given the results of the analysis, a new shape can now easily be calculated as:

$$x_{new} = \bar{x} + \sum b_j \hat{x}_j \tag{2.3}$$

As the PCA furthermore provides information on the statistics of the examples, it is possible to conclude how probable a shape is or how large its deviation is from the mean.

Using this shape model, the texture of the examples are warped into a shape-free form (usually the mean shape). For the areas in between the feature points, the warping is interpolated (piecewise affine transformation, TPS, ...). The effect of this warping is that the shape-free images are now approximately in pixel-wise correspondence. Therefore it is possible to represent them as vectors by simply rasterizing the warped image. Performing a PCA on this data results in the texture model:

$$g_{new} = \bar{g} + \sum a_j \hat{g}_j \tag{2.4}$$

To control the complete appearance model with only one unified parameter c, the correlations between shape and texture are learned by another Eigen-analysis from the training set. Using this single parameter, ideally the whole range of category items can be produced.

$$x_{new} = \bar{x} + Q_s c \qquad g_{new} = \bar{g} + Q_g c \tag{2.5}$$

The "Active" part of Active Appearance Models is the fitting of an appearance model as described before to an input image that contains a specific instance of the category.

This is done by searching for an optimal parameter c, such that a model-generated item matches the image as close as possible. The Analysis-by-Synthesis method generates a first model by using initial parameters. Based on those parameters, the shape is calculated, and the input image g_{im} is back-warped into a shape-free form.

$$g_s = T_u^{-1}(g_{im}) \tag{2.6}$$

In this domain it is possible to calculate a residual between the generated model instance and the actual input image

$$r(c) = g_s - g_m \tag{2.7}$$

The goal is to find a parameter update δc that minimizes this residual. In contrast to usual optimization strategies, the update does not use a derivative of the image residual that has to be recalculated for every set of parameters, but a simple linear model $\delta c = Ar(c)$. This linear relationship is determined in the training phase by perturbing the model parameters in several ways. As long as the parameter update does not lead to a satisfactory model ($r(c) < \epsilon$), the model search is continued in an iterative manner.

Active Appearance Models have been successfully applied to many computer vision tasks (cardiac MRIs [55], human faces [25], X-ray images of bones - robust AAMs [7], ...) and have several advantages. They are fast, include a model-specific knowledge and have few parameters to select by an expert. A drawback is that they need distinctive features which the model adaption can use - so dealing with amorphous objects like trees or clouds is hard for AAMs. Another drawback is that they are only defined for two-dimensional images (although extensions to the third dimension exist [55]), and the used model usually is a sparse one.

2.3 3D Morphable Models

The 3D Morphable Model was introduced by Blanz and Vetter [13] and is a parametrized statistical model that has mainly been applied to human faces. With a few shape and texture parameters it is possible to generate the whole range of plausible objects as high-resolution 3d models. The basic idea is to build a vector space of objects. Specific instances of the object category are generated as a linear combination of the basis vectors. The main difference to Active Appearance Models is the extension to the three-dimensional space and a much denser model of corresponding points.

In this section only an overview is given, and the details are treated in the subsequent chapters.

Building From a large database of raw object examples the morphable model is built. The essential condition for a usable vector space of objects (in this section we refer to human faces, although other object categories could be used analogously) is that the basis faces are in correspondence. This constraint assures that linear combinations generate valid and plausible faces. Blanz and Vetter employed a hierarchical Optical Flow algorithm on the radial height and texture maps to establish correspondence between the raw training examples. After this step, every training example can be represented by a shape vector $\tilde{S} = [x_1, y_1, z_1, x_2, ..., y_n, z_n]^T \in \mathbb{R}^{3n}$ of n vertices (x, y, z) and a texture vector $\tilde{T} = [R_1, G_1, B_1, R_2, ..., G_n, B_n]^T \in \mathbb{R}^{3n}$ containing RGB-color for each of the vertices.

Using a linear combination, a new face can be described by its coefficients for shape a_i and texture b_i for the q training examples:

$$\begin{array}{ll} \tilde{S}_{new} = \sum_{i=1}^{q} a_i \tilde{S}_i & \text{where} \quad \sum_{i=1}^{q} a_i = 1 \\ \tilde{T}_{new} = \sum_{i=1}^{q} b_i \tilde{T}_i & \text{where} \quad \sum_{i=1}^{q} b_i = 1 \end{array} \tag{2.8}$$

Valid generated faces are located in the unit-hypersphere of a and b, respectively. This face vector space is very high-dimensional (q dimensions, as for every example face in the training database, another dimension is added), and it contains quite a lot of redundancy. Using a Principal Components Analysis, this vector space is transformed to a representation where the first dimensions contain the most variation. The least significant $q - m$ dimensions can be discarded to obtain a lower dimensionality.

Quite similar to 2.8 new faces can be described now through shape coefficients α_i and texture coefficients β_i (where S_i and T_i are the new shape and texture basis vectors generated by the PCA):

$$S = \bar{S} + \sum_{i=1}^{m} \alpha_i S_i \qquad T = \bar{T} + \sum_{i=1}^{m} \beta_i T_i \tag{2.9}$$

Because the initial Optical Flow estimate for correspondence might contain errors, a bootstrap approach is applied to improve the model. The most significant parameters of the current model are used to fit the training scans, and these fitted estimates are used as initialization for a next round of optical flow correspondence estimation. This is repeated with an increasing number of model coefficients, until the quality of the model is satisfying.

Fitting Similar to the "Active" part in AAMs, a 3D Morphable Model can be fitted to a two-dimensional image. This is again done in an Analysis-by-Synthesis manner: The iterative process is started from a set of initial model parameters. Then a model instance (an actual face) is calculated from these parameters, and an error between generated instance and the input image is estimated. Based on this error measure, the parameters of the model are updated, so it represents the input image more closely. This process is repeated until the fit is good enough.

A comparison with AAMs shows some major differences. The number of vertices that define the topology of the object is much higher for 3DMM. Blanz & Vetter use tens of thousands, while the sparse mesh of Active Appearance Models usually contains only some dozens of feature points. Furthermore, 3DMMs contain information about the third dimension of the object, whilst AAMs only model two dimensions. This enables the separation of appearance, shape and the influence of light, shadow and different poses. Another difference is how the parameter update is calculated. For 3DMM it is based on the derivatives of the residual that are recalculated every step, while AAMs use only an approximated, precalculated linear matrix.

2.3.1 Approaches & Improvements

Stochastic Newton In the original paper ([13]), Blanz and Vetter already proposed an improvement for the fitting process. Instead of using a simple gradient descent optimization, a stochastic variant of Newton's algorithm is introduced. By randomly selecting only a subset of all vertices for computing the error measure and the parameter update (and resampling this selection from time to time), both some local minima are avoided and the computation is accelerated.

Multiple Feature Fitting Rhomdani proposes in his thesis [68] not only to use the pixel color feature to calculate the error measure, but also to include the information extracted from edges, specular highlights and manual anchor points. The advantage is that the overall cost function has better characteristics for optimization.

Sparse Features → Shape Parameters (and Dense Model) Quite often a set of sparse feature points at the interesting locations of a face is available. Several works try to compute the shape and texture parameters and therefore the dense model of the underlying face from those few points. Hu et al. [37] derive only the shape parameters from some 80 feature points, and refine the resulting mesh using interpolation. The texture is extracted

only from the input image, and the major shortcoming of their method is that it only works for a predefined pose.

Blanz et al. [12] provide several improvements. Through regularization the reconstruction is stable w.r.t. noise in the feature points, and can be calculated directly in one step. Pose estimation (translation, rotation and scale) is included by adding linear and linearized (for rotation) coefficients to the model - due to this approximation usually two iterative steps are needed for the final result. In addition to [37], not only discrete feature points, but also directional constraints are allowed to describe the face.

Combination with AAM's Several researchers explored the combination of Active Appearance Models with 3DMM. Faggian et al. [27] trained two AAM's with data varying in pose and identity. Those training examples were synthesized using a 3DMM with random shape and texture parameters and yaw rotation steps of 5 degrees. Input images are then fitted by the so-generated AAM's and the full 3D model is reconstructed using the method described in [12] and the previous paragraph.

Xiao et al. explore the possibilities of AAMs to encode the same information as 3DMM do. They come to the conclusion that with at most 6 times more parameters Active Appearance Models are able to model the same information/objects/faces as 3DMM do. But furthermore, those models also generate non-plausible instances, and so they use 3D information (either obtained from a structure-from-motion algorithm or from a 3DMM) to restrict the fitting of AAMs. The speed for fitting is even faster than for a regular AAM because of the restricted flexibility.

ICIA applied to 3DMM Baker and Matthews [5] developed a fast algorithm for the fitting of Active Appearance Models called Inverse Compositional Image Alignment. The usual parameter update in the iterative fitting step is an additive one. It combines an incremental parameter update with the current parameter and afterwards computes a new model instance. In contrast to that, the compositional approach first computes a model update and composes it with the current model. By furthermore inverting the role of template (model instance) and input image, it is possible to precompute the Jacobian and allow faster iterations.

Romdhani and Vetter adopted the ICIA for 3DMM in [69]. They showed that some simplifications made in [5] do not hold for dense meshes, and modify the original algorithm w.r.t. those shortcomings. Furthermore, also the texture parameters are included in the algorithm. The disadvantages of this algorithm are that only weak perspective is

considered, and only a very simple illumination model (ambient light) is allowed.

2.4 Areas of Application

In this section an overview is given of the areas statistical shape models (and 3D Morphable Models in particular) are used. For most of those areas also alternative approaches exist and are therefore mentioned too.

2.4.1 3D Face Recognition

Face recognition is one of the most obvious, but also most challenging tasks that a computer may perform. Humans easily recognize and identify other human faces very fast and under varying conditions.

For computers two tasks are distinguished: Identification (recognition) is the matching of an unidentified person against a database that contains a set of registered persons. Verification (authentication) is the process of checking if a person is the one she claims to be. Usually a face recognition system is not able to perform well in both scenarios, because they have oppositional requirements.

Although there exist a lot of algorithms for face recognition [61], [1], most of them perform well only under standard conditions. A major influence on the performance plays the illumination of the subject. Recognition at frontal pose works quite well, but if the head turns by more than a few degrees, a significant performance degradation can be observed [31]. Also variations of the subject such as expression, aging and facial hair affect most algorithms. Occlusions of the subject (sun glasses, hats, ...) usually have a large impact on the recognition rate. So the robustness of an algorithm is greatly influenced by these conditions, and the "ideal" algorithm would perform well under all of them.

In this section, an overview only over 3d recognition algorithms is given. 2d algorithms work on image data that is "flat" - i.e. the usual images - and are treated in the next chapter 2.4.2. In contrast 3D algorithms use the three-dimensional information that represents the shape of a face. For shape-only algorithms, this is the only information they use to perform recognition, while multi-modal algorithms use both the shape and a 2D texture (the color of the face).

Most of the face recognition algorithms follow this sequence:

Data acquisition The necessary face data is captured by a 3D scanner. Widely used are laser scanners, stereo camera pairs and structured light cameras. Laser scanners project a

laser beam light sheet onto the object, and using a camera the coordinates of the profile can be triangulated [40]. The stereo camera technique uses two cameras placed slightly apart, and the 3D coordinates of the viewed object are computed using stereoscopic algorithms [6]. Structured light scanners project different regular patterns of light on the object and can derive the shape by the way the patterns are deformed [66].

Preprocessing The raw data obtained from 3d scanners usually contains several artifacts. Laser scanners produce spikes and holes that can be cleaned by using (for example) median cut or radial basis functions respectively. Some algorithms are sensitive to the data resolution, which requires the resampling of the data to a specified raster. If the adopted algorithm needs normalized pose, the rigid alignment of the face data is also done in this step.

Extract information The core part of all algorithms is to extract the metadata from the preprocessed scans. In the ideal case this metadata characterizes the captured face uniquely and is robust / invariant to distortions (expression, lighting, facial hair, etc. - see above). Research literature brought up a wide variety of approaches to address this topic (see survey [70]). Some examples are:

- Curvature: Based on the Gaussian curvature, the image is segmented and the regions are characterized by Extended Gaussian Images [72]

- ICP: By using the Iterative Closest Point algorithm (see section 3.4.1.1) the best match in the database is determined. To account for non-rigid deformations, Thin Plate Splines have been used [50].

- Profile: Both horizontal and vertical profiles are used for the description of the face.

- Point signatures: The region around some points of interest is characterized by signatures [18].

- Template matching: The parameters of a deformable template are iteratively adjusted to fit the given data. A member of this algorithm family is the 3D Morphable Model that is detailed in this thesis. Another variant is given by Ansari and Abdel-Mottaleb [4] where the parameters of the 3D model Candide are tuned.

Database comparison The information extracted in the previous step is compared to the samples in the gallery. For recognition (identification), the best match is searched, whereas for authentication (verification) the match has to surpass a certain threshold. Many algorithms perform this database search implicitly during the previous step of meta-data extraction.

3D Morphable Models can be used in a variety of ways for 3D face recognition. The direct approach to fit a 3DMM directly to the cylindrically parametrized data has been shown in [13]. In contrast, Amberg et al. [2] use the 3DMM only to regularize a non-rigid ICP variant (OSNRICP, see section 3.4.4) that matches the data. Another use scenario is to first reconstruct the 3D shape of a face given in a 2D image by the Morphable Model Method, and then do the actual recognition with another algorithm. But of course also the actual model parameters easily lend themselves as the basis for recognition.

2.4.2 Normalization (ICAO) & 2D Face Recognition

Most of the data that represents human faces is available in the form of two-dimensional images. Having evolved from the analog pin-hole camera to high quality digital cameras nowadays, the type of devices that produces this kind of data is widely present and lots of images are readily available. But nevertheless, there are some significant problems when dealing with human faces in 2d images. Several distortions occur in the process of image acquisition, which can either be introduced by the subject or the acquisition process [22]. A first variation is introduced by the relative pose of the person and the camera. Also human expressions add significant differences to portraits (especially around the mouth occur large deformations), as do accessories such as sunglasses, hats, ... If some time passes between capturing two images of the same person, also aging has an influence. Illumination is one of the most important conditions in the acquisition process, others are the sharpness of the photograph, the resolution and representation of the image.

It would be ideal to have the face images in a normalized form. Normalization means that the pose of the face is defined (usually frontal view), the person presenting a neutral expression and the scene is shot under specified illumination. The ICAO (International Civil Aviation Organization) division for Machine Readable Travel Documents laid the foundations for the ISO/IEC standard 19794-5:2005 [39] that provides a detailed specification of how facial images should look like. The eyes are taken as fixed points in a defined coordinate system, and the pose deviation has to be less than 5° from frontal. Furthermore, conditions for valid pictures are concerned with facial expression, lightning

conditions, the eye-looking direction and more.

Normalized faces are used in a wide variety of areas. Their main advantage is that those photos are easier to compare. Therefore they are deployed in passports and similar official documents to allow easier recognition and verification both for humans (e.g. border control, police, ...) and computers (automated 2D face recognition).

The computer identification or verification of faces in two-dimensional images can be roughly separated into 3 key parts: (a) face detection / segmentation, (b) feature extraction and (c) recognition / identification. Face detection and segmentation finds the place where the face is located in the 2D photograph and segments it from a probably cluttered background. Often detection and feature extraction happen simultaneously. Features are both local characteristics such as lines of fiducial points and facial features such as mouth, eyes and nose. The extracted features are then used to compare the input to a database of previously stored templates. The categorization in holistic, feature-based (local, structural) and hybrid matching methods has been suggested from psychological studies [73]. The oldest approach is the feature-based comparison, where the locations of feature points are evaluated and matched against the faces in a database. Bledsoe [15] for example used the normalized distances between fiducial points such as the corners of the eyes or the mouth. In contrast the holistic or global approach observes the face as an entity and tries to match the whole template against the stored database. Examples of this category are the "Eigenfaces" [74] or "Fisherfaces" [8]. Hybrid methods finally merge both local and global information of a face to perform a reliable matching to the database. The Flexible Appearance Model [45] as a representative of this category is specified both by its overall shape as by local gray-level profiles at the model points. Most modern face recognition algorithms use hybrid methods as this potentially provides the most information to identify or verify human faces. An extensive overview of 2D face recognition is given by Zhao et al. [81], and Abate [1] also tries to compare the performance of the most important algorithms. To compare current face recognition systems is also the goal of the Face Recognition Vendor Test [64].

3D Morphable Models can be used both to generate and evaluate normalized faces. For generation, the model is fitted to an input image. Because the illumination and pose are separated from the model itself, the generated 3d model can be rendered under standardized lightning and pose. And if the model also contains expression modalities, the mimic of the model could be neutralized too. On the other hand, to evaluate whether an image fulfills standards, only the fitted parameters (expression, rendering, light, pose)

have to be analyzed.

In face recognition, 3D Morphable Models can be used to provide normalized images to recognition algorithms, which have problems with variations in illumination, pose, etc. Tests in the FRVT 2002 show, that this use of 3DMM significantly improves the performance of traditional face recognition systems (Phillips et al. [63], report up to 62 percent better identification performance). But also the direct use of the model parameters that are determined by fitting a 3DMM to a photograph can be used for face recognition [14].

2.4.3 Pose Estimation & Tracking

The pose of the human head plays an important role in the communication between humans. From active interaction by nodding to answer somebody's question, over subconscious signals that signal (dis-)agreement, up to noticing a persons focus of attention by the pose of her head and eyes, computers are excluded from an essential area of interaction and observation if they are not able to estimate a humans head pose.

The research community has brought up a lot of algorithms that try to solve the problem, and the survey of Murphy-Chutorian [56] provides an up-to-date starting point into this topic.

When a 3D Morphable Model is matched to an image, the pose parameters (position, rotation angles) are explicitly fitted. This allows a quite exact estimate of the pose, also because the model accounts for inter-subject differences. One problem can be the region of convergence of the original fitting algorithm that usually does not span the whole domain (e.g. vertical rotation from -90° to 90°). Either several initializations from where the fitting is started, or a fitting algorithm with a broader convergence region should be tried in this case.

2.4.4 Training & Test Data Synthesis

In computer vision, a lot of algorithms draw the knowledge to identify or discriminate objects from a training data set. They interpolate between different objects, learn the common or distinguishing features of object categories and use the data set to account for varying conditions. Some selected examples are artificial neural networks, Bayesian learning - also 3D Morphable Models are constructed from a learning data set. But all these approaches have a large disadvantage in common: It is a tedious and labor-intensive process to build a training database, especially if the "objects" are humans.

Another equally tedious task is to build databases to test and evaluate algorithms

- the reader may just think of face recognition algorithms, where several thousands of individuals can be necessary to assess the quality of an algorithm [64].

3D Morphable Models can be used to generate synthetic images of faces. Via shape and texture parameters, different identities can be created, and the rendering and light parameters account for variations in pose and illumination. If the model includes expression parameters, even images with varying mimics can be synthesized. The advantages are a very fast creation of such databases, and the known "ground-truth" parametrization of light and pose.

Heisele et al. [78] used 3DMM to provide images of one person under different pose and illumination. They fitted the model to three input images of every person, and then generated many training images under varying pose and illumination. On components of those training images, a SVM for every person learned to distinguish between the person and all the other subjects in the database.

2.4.5 3D Face Animation

Three-dimensional animations play a more and more important role in our lives. Most cinema movies contain at least some animations, realistic images are generated from computer models and instead of doing sports in the real world, people sit on their couches playing interactive games. To realistically model humans in those virtual worlds, the animation of the face plays a very important role.

But it is quite difficult to describe expressions in terms that are understandable for computers. One approach that is taken by the 3DMM is to annotate some images with the expressions they represent, and let an algorithm learn them. For every expression one or more coefficients are added to the model. Due to this representation, the model can not only "fit" the exactly same expressions that it learned, but also their linear variations and combinations - e.g. a little bit of smiling whilst forming the vocal "o". By fitting an unknown face in an image, the computer can therefore derive the expression (and other modes such as male-female, facial weight, ... learned by the model) as numeric coefficients. The next possibility is to transfer such an expression to a different face. This is done by adding the expression coefficients of the intended "mimic" to the neutral target face. But also a new expression can be generated by numerically describing its coefficients.

There are a lot of practical applications that emerge from this use of 3DMM. The animation of still images is one of the most obvious. Who has not tried to capture a group photograph where on each image, a person has its eyes closed or another looks grimly. But

having an animated 3D model of a person, it is also possible to produce a whole movie without her personal attendance (e.g. after her dead). In interactive games, the animated alter ego is generated from one captured image. By using this technique also for video conferences, it has another advantage: the representation of a face and its expression by 3DMM coefficients is also very compact and can therefore be transmitted very effectively.

2.5 Conclusion

In this chapter we give an overview over Statistical Shape and Appearance Models, and point out some historical stations on the way of their evolution. Then the concept of both Active Appearance Models and the 3D Morphable Model is laid out, and some of their properties are compared. Finally, the some of the many areas where 3D Morphable Models prove to be useful are elaborated.

3D Morphable Models are studied in detail in the rest of this work, because they have some interesting and promising properties: The extension to the third dimension allows to explicitly describe and calculate the pose and lighting of a face. This is not only important for the generation and rendering of a new face, but especially for the analysis and the fitting of a persons photograph with unknown pose and illumination. Furthermore the curvature of a persons face (an implicit property of a three-dimensional model) is useful for its identification and characterization. In contrast to several other 2D and 3D models, the level of detail of 3DMMs (density of the underlying triangle mesh) is much higher, which makes the creation of very realistic faces possible. The holistic approach allows a very efficient representation of a complete human face with only a few numerical coefficients. This representation can also be used to transfer and modify identities or expressions from one head to another.

Chapter 3

Building a 3D Morphable Model

Contents

The statistical model is one basic pillar of the 3DMM framework. It contains the condensed knowledge learned from the samples that it is built of. The goal of this chapter is to describe the algorithm to automatically construct an accurate and well-formed 3D morphable face model from a database of raw 3D scans of human heads.

The complete procedure from the raw scans to the final model involves several areas of computer vision. It begins with the collection of raw head data in form of shape coordinates and a corresponding texture (section 3.1). The raw data is preprocessed for artifact correction in the next section. The triangle mesh defining the topology of a head scan is cropped and simplified in section 3.3. The main step in constructing a 3DMM consists of establishing dense and accurate 3D correspondence (section 3.4) between all the head scans. Using the registered head scans, mean shape and texture are calculated. Further statistic properties are determined by the Principal Component Analysis in section 3.5, which removes redundancy from the data and transforms the model to a new, more efficient coordinate system.

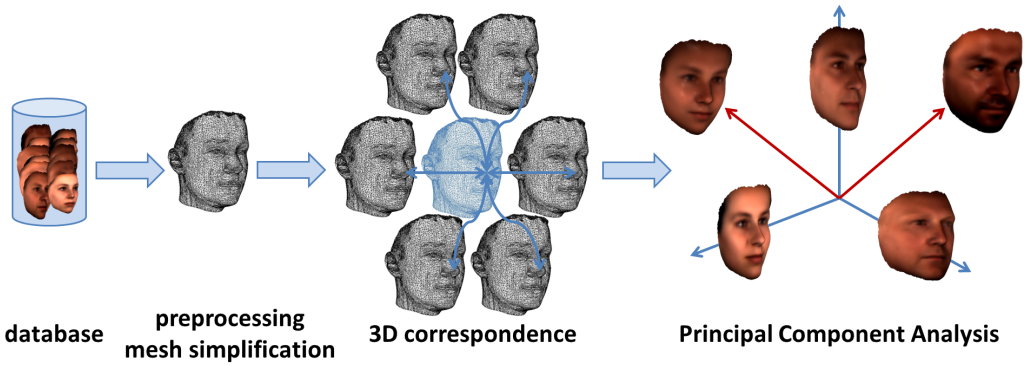


Figure 3.1: Building a 3D Morphable Model

3.1 Head Scan Database

The basis of the 3D Morphable Model is formed by a large database of human head scans that comprises more than 9.000 subjects.

As part of the "Steirische Landesausstellung 2000 comm.gr2000az", a Cyberware 3030 laser scanner* was used to capture high-resolution models. This device produces radial range maps $r(h, \phi)$ with 512 angle steps ϕ and 512 height steps h. Furthermore, for each data point RGB color information $R(h, \phi)$, $G(h, \phi)$, $B(h, \phi)$ is given. Figure 3.2 shows an example of a height and texture map. The resolution of the scan per data point is 8 bit for the height map and 8 bit per color channel and data point.

(a) (b) (c)

Figure 3.2: (a) Height map, (b) the corresponding texture and (c) a rendering of a head scan

*Cyberware 3030 3D Color Scanhead `http://www.cyberware.com/products/scanners/3030.html`. Accessed on November 7, 2008

This laser scanner obtains the data by projecting a laser light stripe on the facial surface and capturing the resulting profile using a digital camera. Along the stripe it calculates the 3D locations of points, and simultaneously extracts the texture color for each of these points under uniform illumination.

There are two main alternative techniques to the laser scanner for 3D face data acquisition. Using a pair of cameras, the stereoscopic system captures images of the face from two different viewpoints and calculates the 3D shape by a triangulation of corresponding points. The Geometrix system[†] uses this method. The other option for capturing three-dimensional images of faces uses varying structured light patterns. These patterns are projected on the face and can be used to deduce its 3D shape. The Minolta scanner[‡] is a representative of this category. To improve the quality and density of corresponding points, the 3dMD face system[§] uses both structured light patterns and a pair of cameras to derive the shape of a face.

Our database contains one sample of every subject. These subjects vary with respect to age (from child to grandpa), color skin (majority is white) and gender. Some of the male subjects have beard or mustache, and some subjects wear ear rings or studs. Only a small minority is smiling or has the mouth opened, the majority poses with neutral expression and closed mouth. Eyeglasses are taken off during the scan acquisition.

Due to the radial (cylindrical) way of capturing the data, some problems can not be avoided. Subjects that are displaced from the radial axis are sampled non-uniformly. Another issue occurs with non-convex structures, which can not be sampled completely by a radial laser beam. This is intuitively understandable for the region of nostrils or the ear. For our model, the region behind the chin (called "submandibular triangle") has greater influence, but is not recorded correctly for some persons with a pointy chin (see Fig. 3.3).

Hair influences the scans in two ways. It is nearly impossible to be captured correctly due to its fine structure. And some subjects cover parts of their facial region because of a special hairstyle (fringes, emo-style, ...).

Another inaccuracy occurs due to the movement of subjects during the capture process. If the subjects head is displaced when the scan head rotates around it, incorrect measurements are made. Sometimes this can be detected, if the vertical scan line at the start and end (0° and 360° respectively) are not equal.

[†]Geometrix. `http://www.geometrix.com/`. Accessed on March 24, 2010

[‡]KONICA MINOLTA VIVID 9i-Non-contact 3D Digitizer. `http://www.konicaminolta.com/instruments/products/3d/non-contact/vivid9i/index.html`. Accessed on March 24, 2010

[§]3dMDface System: `http://www.3dmd.com/3dmdface.html`. Accessed on March 24, 2010

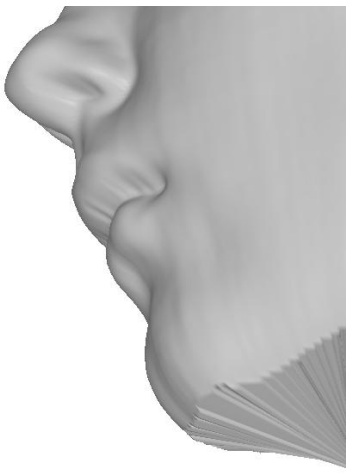


Figure 3.3: Incorrectly captured chin due to its non-convexity and the radial sampling method of the laser

The scans are stored in the proprietary echo file format[¶]. The first part of an echo file is constituted by an ASCII header that describes the parameters of the scan. The angular increments between the horizontal scan profiles, the height of horizontal start, end and increment of the data points, and meta data such as recording date, scan name etc. are stored. The second part (following the DATA tag in the header) contains the actual measurements. For every angular increment there is a group of data points describing the profile of the object. The radius in microns at a data point consists of a 4 byte long integer (big-endian) that is compressed by cutting off the last two bytes. For further processing, a simple cylinder mesh was used to tessellate the laser scan.

3.2 Preprocessing

The raw result given by the scanner contains artifacts such as spikes and holes in the mesh. Spikes are mostly caused by highly reflective areas (e.g. the eyeball or earrings), while holes occur both in the area of non-reflective parts and regions that can not be reached by the laser beam. Although there exist algorithms for automated correction of those errors ([28], [79], ...), they have been removed manually after the scanning process using a mesh editing tool in this case.

[¶]Echo File Format. `ftp://ftp.cyberware.com/pub/echoFormat.tar.gz`. Accessed on June 28, 2009

3.3 Mesh Simplification

The data points constituting the mesh of the head scan are more or less spatial uniformly tessellated (see section 3.1). To reduce the computational effort in the following steps, the mesh can be simplified at regions with little details.

Most algorithms work by iteratively merging vertices, and thereby simplify not only the mesh but also the topology. The used qslim [30] is an iterative algorithm. First for all vertex pairs that can be merged, an optimal new vertex location $\bar{v}$ is determined by using a quadric error norm $\Delta(\bar{v})$.

$$\Delta(\bar{v}) = \bar{v}^T Q \bar{v} \tag{3.1}$$

The vertex pair with lowest cost of contraction (error at the optimal location) is merged and for all neighboring vertices, we update their error measures. This error norm represents the squared sum of the distances between the vertex and a set of relevant planes. Its matrix Q is initially calculated for all vertices using their adjoining planes. The update of the error measure is approximated by summing up the Q-matrices:

$$\bar{Q} = Q_1 + Q_2 \tag{3.2}$$

This contraction of vertices is repeated until the desired level of detail is reached. Figure 3.4 shows the simplified mesh of a head with 20.000 triangles and 10.424 vertices. The higher density of the mesh at important regions such as mouth and eyes can be seen clearly.

qslim has been chosen because the algorithm preserves structural details ("high quality simplification"), is fast, fully automatic and there exists a free implementation[‖].

This algorithm is applied to one head scan that is chosen as the template for establishing correspondence in section 3.4. The disadvantage of this approach is that the simplification is based on one actual template. Therefore the final model may contain a dense mesh at regions that are characteristic only for that specific instance (e.g. birthmark), but can have a sparse mesh at other regions that are specific to various samples in the database (e.g. at the cheek, where other persons may have dimples). This shortcoming can be adressed by basing the simplification on all subjects in the database [77], although the technique described there requires manual labeling for every scan.

[‖] QSlim Simplification Software. `http://mgarland.org/software/qslim.html`. Accessed on March 24, 2010

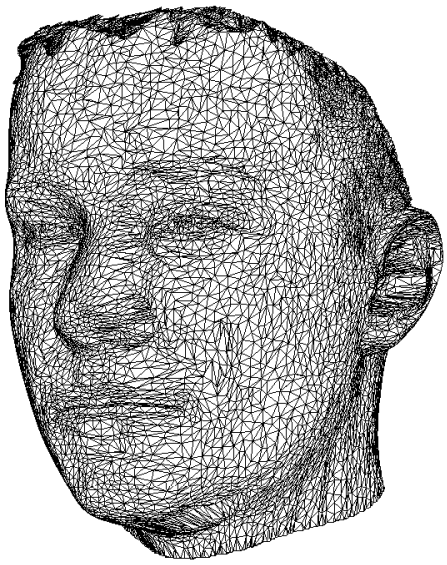


Figure 3.4: Simplified mesh with 20.000 triangles and 10.424 vertices .

3.4 3D Correspondence

Two head scans (represented by a number of vertices) can informally said to be in correspondence, when all the vertices in these two scans lie at the same semantic position of the face - e.g. in both scans the fiftieth vertex is located exactly at the tip of the nose.

Then a scan can be represented by a shape vector $\tilde{S} = [x_1, y_1, z_1, x_2, ..., y_n, z_n]^T \in \mathbb{R}^{3n}$ where x_i, y_i and z_i are the coordinates for all n vertices, and a texture vector $\tilde{T} = [R_1, G_1, B_1, R_2, ..., G_n, B_n]^T \in \mathbb{R}^{3n}$ that contains the red, green and blue color values for each vertex. The topology of the vertices is defined by a generic mesh for all scans.

A more formal definition first requires the introduction of landmark points. They are prominent characteristic locations that can be identified in every instance of an object. In the category of faces for example, the corners of eyes and the mouth, and the tip of the nose lend themselves to be used as landmarks.

Based on those landmarks, correspondence can be defined as the spatial mapping of landmark vertices between two scans. To find this elastic transformation of one prototype onto the other, several approaches have been developed. In the following subsections, Thin Plate Splines, Optical Flow and a non-rigid variant of the Iterative Closest Point algorithm are discussed. For building a 3D Morphable Model, it is necessary to not only establish the correspondence between the sparsely distributed landmark points, but also

a dense correspondence (i.e. between all the vertices of a densely sampled model) in all regions of the face. Therefore the Thin Plate Splines are not usable for our task. The dense correspondence can be difficult to determine exactly for structures that have no counterpart in another face (like birthmarks, scars, ...) or for "low contrast areas" (as the cheek). The algorithm chosen (OSNRICP, section 3.4.4) smoothly interpolates the correspondence field for such cases. Optical Flow can also be adapted to provide this interpolation, but the data of the head scans proved to be to noisy (dangling hair, etc.) for this approach. In contrast, the non-rigid ICP variant works well also for those distorted samples and has therefore been used in the practical implementation.

3.4.1 Rigid Alignment

Most algorithms that establish correspondence between two scans need a rough initialization, and rigid alignment provides a sufficient starting point. A transformation denoted as rigid is constituted only from translation and rotation. In our case, the framework's rigid alignment step also includes a scaling transformation, which by the exact definition is not a rigid transformation. But the non-rigid correspondence finding that follows after the rigid alignment needs the two scans to resemble each other up to a certain degree, which can be only achieved by bringing them to the approximately same scale.

The raw head scans in the used database are coarsely aligned with respect to the vertical position, but are arbitrarily rotated around the vertical axis. So in a first step a face detector that employs AdaBoost [71] has been applied onto the texture of the scans to determine the facial region (eyes and mouth). In this region the nose can be identified in the range map as the point with maximum elevation and serves as a first landmark point for rough alignment.

Starting from this rough initialization, the Iterative Closest Point Algorithm (ICP) is used to provide a finer rigid alignment and a good estimate of scale.

3.4.1.1 Iterative Closest Point Algorithm

The Iterative Closest Point algorithm (ICP) is a popular algorithm to align two free-form surfaces (S_1, S_2) that are represented by a set of points (X_1, X_2). It has first been described by Besl and McKay [10].

The algorithm consists of two steps. First, for every point x_1 in the template mesh X_1, the closest point to it on the target surface S_2 is searched by finding the point x_2 with minimal Euclidean distance $min(\| x_1 - x_2 \|)$. To speed up the search time, usually

a k-d-tree is built from the template points X_1 and used to discover the closest points.

The second step is to find a rigid / affine transformation A that minimizes the squared distance between the two point sets based on the temporary correspondence (closest points) established in the first step. The Procrustes alignment 3.4.1.2 can be used to compute this transformation. It is then applied to the target surface $S_{2,new} = A(S_2)$.

Those two steps are iterated until the distance between the two surfaces falls below a predefined threshold. All the iterative transformations in the second step are combined to determine the absolute transformation that best aligns the two surfaces. The ICP algorithm has a very fast convergence, but the region of convergence is limited - it is not guaranteed to find a global minimum. Therefore either a rough initialization (the nose was used for the head scans in our case) or multiple runs with different starting points have to be used.

3.4.1.2 Procrustes Algorithm

The algorithm named after a cruel Greek legend is used to determine the rigid transformation A that minimizes the Procrustes distance d between two sets of corresponding points X_1 and X_2 (usually those correspondening points have been determined in the previous step of the ICP algorithm 3.4.1.1).

$$d = \sqrt{(x_{1,1} - x_{2,1})^2 + (x_{1,2} - x_{2,2})^2 + \ldots + (x_{1,n} - x_{2,n})^2} \tag{3.3}$$

The Procrustes analysis is often used as second step in the ICP algorithm. First both point sets X_1 and X_2 are aligned to their center of mass.

$$X_{1,centered} = X_1 - mean(X_1) \qquad X_{2,centered} = X_2 - mean(X_2) \tag{3.4}$$

Then a Singular Value Decomposition (SVD) is used to extract the rotation that minimizes the distance d. It is computed on the covariance matrix H of the two centered point sets

$$H = X_{1,centered}^T X_{2,centered} \tag{3.5}$$

SVD decomposes this matrix H into an orthonormal matrix U, a diagonal matrix Σ that contains the singular values of H in the main diagonal and the transpose of another orthonormal matrix V

$$svd(H) = U\Sigma V^T \tag{3.6}$$

The rotation R that minimizes the Procrustes distance between the two point sets (best alignment in a least squares sense) can then be computed using the matrix

$$D = diag([1, 1, det(V * U^T)]) \tag{3.7}$$

as follows

$$R = V * D * U^T \tag{3.8}$$

In our case not only the rigid alignment is determined, but also a similarity transform (uniform scaling s)

$$s = \frac{\sum RX_{1,centered}\dot{X}_{2,centered}}{\sum X_{1,centered}\dot{X}_{1,centered}} \tag{3.9}$$

To complete the Procrustes Analysis (in the original analysis, scale s is not used here), the translation t can finally be computed by

$$t = mean(X_2) - sRmean(X_1) \tag{3.10}$$

3.4.2 Thin Plate Splines

Thin Plate Splines (TPS) [17] provide a smooth interpolation of a deformation defined at some fixed points. The technique is inspired by the physical analogon of bending a thin metallic plate that is clamped to a few anchors. For our 3D correspondence, those anchors $\mathbf{p}_i$ are an otherwise (i.e. through manual labeling) given set of sparse correspondences. Between them, the TPS generates a smooth field of intermediate correspondences.

The smoothness constraint of the displacement field is fulfilled by minimizing a functional $u(\mathbf{x})$ of the following form (for the 3-dimensional case):

$$u(\mathbf{x}) = \sum_{\nu=1}^{4} a_\nu \phi_\nu(\mathbf{x}) + \sum_{i=1}^{n} w_i U(\mathbf{x}, \mathbf{p}_i) \tag{3.11}$$

This functional represents the added bending energy of thin plates that are fixed at the defined sparse correspondence points. The kernel $U(\mathbf{x}, \mathbf{p}_i)$, a special radial basis function, is the central part of this formulation. The other part of the functional is constituted by

the affine transformations ϕ_ν. For further details see the chapter on TPS in [67].

The big advantage of this formulation is that there exists a closed-form solution for the minimization, that therefore can be calculated efficiently.

But TPS need a sparse set of correspondence points to interpolate the dense mapping. A first approach would be to manually define the landmark points in all examples. But for large databases, this method is not feasible. Therefore, for our framework it has not been used. Patel et al. [58] nevertheless did that manual work. They located the landmark points in all scans they use for the construction and build the dense correspondence using the here described Thin Plate Splines. The approach of Chui and Rangarajan [19] is much more robust with respect to outliers in the correspondence points by using deterministic annealing. Therefore it could probably be applied to a set of unreliable correspondence points that have been automatically located by a feature detector.

3.4.3 Optical Flow

Optical Flow estimates the correspondence between two images by making the basic assumption, that pixel intensity does not change over time and motion. It has been used to track the motion of an object through a time-series of images, or to determine the deformation of an object between two images. The basic assumption can be formulated as follows (where I is the image intensity at a given pixel location x, y at a given time t):

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{3.12}$$

The problem with this basic formulation is that it is underconstrained and can therefore not be solved in a deterministic way. There exist two major algorithmic directions that introduce additional constraints to allow the Optical Flow problem to be solved.

The optical flow algorithm developed by Lucas and Kanade [51] assumes, that the intensity not only of one pixel but also of the region around it stays constant over an incremental time and motion step. Using this constraint, the optical flow field can be computed in a non-iterative manner between two images. The problem of this approach is that at areas with little intensity variation, the correspondence is not very well defined.

Horn and Schunk [35] on the other hand tackle the aperture problem by introducing a smoothness condition for the optical flow field. This has the advantage of providing a dense correspondence field also at regions with low contrast.

In practice, both algorithms are applied in a coarse-to-fine manner to avoid being trapped in local minima and for faster computation. This solves some of the problems of

Optical Flow with large displacements and aliasing. But other conditions such as varying illumination or heavy noise pose difficulties for many OF approaches. Hair dangling in the face was one of the main "noise" sources in the scans of our database, and therefore, another algorithm detailed in the next section is used to establish 3D correspondence in our framework.

Blanz and Vetter [13] use a hierarchical optical flow algorithm [9] to establish correspondence between their scans. This algorithm is based on the idea of Lucas and Kanade, but applies the aforementioned coarse-to-fine steps. Furthermore, Blanz and Vetter apply a smoothing step in every iteration to the flow field to obtain a continuous correspondence. From a first temporary correspondence, a model is built, the scan is approximately matched, and the optical flow is computed again to get a more precise model.

3.4.4 Optimal Step Non-Rigid ICP

In search for an algorithm to establish a dense 3D correspondence between a 3D template V and a target T that is also robust to holes, Amberg et al. [3] extended the idea of the Iterative Closest Point algorithm to non-rigid deformations.

The main idea is to assign an affine transformation X_i to every vertex v_i of the template V – instead of one global transformation for all vertices in case of the conventional ICP. Those transformations should warp the template in a way such that it matches the target surface T as close (defined by an error norm, see equation 3.13) as possible. At the same time, the transformations have to fulfill some constraints such as minimizing a landmark term and a smoothness regularization. The main steps of the algorithm are presented here, but for a more detailed description see the original paper [3].

As the regular ICP algorithm, OSNRICP consists of two steps that are iteratively repeated. The first step is to establish a temporary point correspondence (u_i, v_i) between the target vertices $u_i \in T$ and the template vertices $v_i \in V$. This is again done by a nearest neighbor search, assisted by a k-d-tree built from the template points. The second step is to deform the template based on this temporary correspondence in the most optimal way. The error measure $E(X)$, parametrized with the local affine transformations X, needs to be minimized to find this optimal deformation.

$$E(X) = E_d(X) + E_s(X) \tag{3.13}$$

The first part of this error function E_d describes the distance between the two surfaces. The reliability of each correspondence between two points is denoted by w_i or $W =$

$diag(w_1, \ldots, w_n)$ for all of them.

$$E_d(X) := \sum_{v_i \in V} w_i \parallel X_i v_i - u_i \parallel^2 = \|(W \otimes I_3) \left(\begin{bmatrix} X_1 & & \\ & \ddots & \\ & & X_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} - \begin{bmatrix} u_1 \\ \vdots \\ u_2 \end{bmatrix} \right) \|^2 \tag{3.14}$$

Using this formulation, it is not easy to differentiate the error function (for minimization). By reformulating it, it is easier to compute the first derivative, from which the minimum can be determined by setting it to 0.

$$D := \begin{bmatrix} v_1^T & & & \\ & v_2^T & & \\ & & \ddots & \\ & & & v_n^T \end{bmatrix} \tag{3.15}$$

$$E_d(X) = \|W(DX - U)\|_F^2 \tag{3.16}$$

The second part of the error function is an adjustable stiffness term that regularizes the difference of transformations in the vertices neighborhood. Thereby, the deformation changes are smoothed out.

$$E_s(X) = \alpha \sum_{i,j \in edges} \|(X_i - X_j)G\|_F^2 \tag{3.17}$$

By varying the stiffness parameter α the regularization can be adjusted. A high value only allows low local deformations and therefore the behavior converges to the traditional ICP algorithm. Lowering this value allows the template to fit the target more accurately. The equation can again be reformulated to simplify differentiation.

$$E_s(X) = \alpha \|(M \otimes G)X\|_F^2 \tag{3.18}$$

The matrix M describes the mesh of the template (one column per vertex, one row per mesh edge, edge r connects vertices i and j: $M_{ri} = -1$, $M_{rj} = 1$). The matrix $G = diag(1, 1, 1, \gamma)$ can be used to weight the rotational against the translational part by a factor γ.

So the overall error function that has to be minimized can be written as

$$E(X) = \| \begin{bmatrix} \alpha M \otimes G \\ WD \end{bmatrix} X - \begin{bmatrix} 0 \\ WU \end{bmatrix} \|_F^2 = \|AX - B\| \tag{3.19}$$

Given that the matrix A has full rank (for a proof see [3]), its pseudoinverse is $(A^T A)^{-1} A^T$ and the optimal transformations are $X = (A^T A)^{-1} A^T B$.

The procedure is started using a high stiffness value α, and iterating over the two described steps until there are no significant changes anymore. Then α is gradually lowered whilst carrying out the described iterations until the template resembles the target as close as needed.

The advantage of this formulation is that for every iterative step (and temporary fixed correspondence), the optimal local deformations have a closed form solution. Disadvantage are the huge sparse matrices that are computationally costly with a growing number of vertices.

Another nice property is that for areas that are missing in the target surface, the corresponding areas in the template are deformed only by the existing neighbor vertices through the regularization term. That allows a nice hole-filling characteristic that preserves the prior shape of template. Furthermore, automatic downsampling is carried out if the template mesh has a lower resolution (less points) than the target.

The formulation also allows the easy inclusion of a landmark term. This can be useful, when the data has already some labeled feature points.

Disadvantages of the algorithm are the huge sparse matrices and the therefore costly computations on them.

After applying this algorithm to the heads in the database, we obtain a set of registered scans that form the basis of a face space.

3.4.5 Face Space

When face i is brought into correspondence, it can be described by shape vector $\tilde{S}_i = [x_1, y_1, z_1, x_2, ..., y_n, z_n]^T \in \mathbb{R}^{3n}$ of n vertices (x_i, y_i, z_i), a texture vector $\tilde{T}_i = [R_1, G_1, B_1, R_2, ..., G_n, B_n]^T \in \mathbb{R}^{3n}$ containing RGB-color for each of the vertices and a generic triangle mesh that represents the topology of the vertices. Using all the q faces in correspondence, it is possible to span a space of faces. Every registered training face i from the database adds another dimension to the space, and a valid new synthetic face (shape and texture vector) is created by barycentric coordinates in it.

$$\begin{aligned} \tilde{S}_{new} &= \sum_{i=1}^{q} a_i \tilde{S}_i \quad \text{where} \quad \sum_{i=1}^{q} a_i = 1 \\ \tilde{T}_{new} &= \sum_{i=1}^{q} b_i \tilde{T}_i \quad \text{where} \quad \sum_{i=1}^{q} b_i = 1 \end{aligned} \tag{3.20}$$

These linear combinations of the original faces create realistic synthetic ones. Figure 3.5 shows some examples from the two-dimensional face space formed by the faces at the left and right. The second face from the left in Figure 3.5 with the face space coordinates $a = b = (0.75, 0.25)$ is formed by

$$\tilde{S}_{new} = 0.75 \cdot \tilde{S}_1 + 0.25 \cdot \tilde{S}_2 = \begin{bmatrix} 0.75x_1^1 + 0.25x_1^2 \\ 0.75y_1^1 + 0.25y_1^2 \\ 0.75z_1^1 + 0.25z_1^2 \\ 0.75x_2^1 + 0.25x_2^2 \\ \vdots \\ 0.75y_n^1 + 0.25y_n^2 \\ 0.75z_n^1 + 0.25z_n^2 \end{bmatrix} \tag{3.21}$$

Figure 3.5: Examples from the two-dimensional face space formed by the faces at the left and right. The coordinates a and b of the faces (equal for this figure) are $(1, 0); (0.75, 0.25); (0.5, 0.5); (0.25, 0.75); (0, 1)$

To remove the constraint of the barycentric coordinates, the space can be slightly modified by removing the mean from the dimensions. Then realistic faces are in the unit sphere, and coordinates with greater distance from the center represent caricatures.

Disadvantages of this representation are its high dimensionality and the redundancy it contains - if two similar faces form the basis of two separate dimensions, the same face can be generated by the contribution of either one or the other dimension. A solution to both of these problems is presented in the next chapter.

3.5 Principal Component Analysis

Principal Component Analysis is a technique to reduce the dimensionality of data whilst preserving as much information as possible. This is achieved by transforming the data to a set of new orthogonal basis vectors and discard the most insignificant dimensions in the new coordinate system (those dimensions where the data contains the least variance). It has been invented independently by Pearson [59] in 1901 and Hotelling [36] in 1933 and generalized by Karhunen-Love [49]. The book of Jolliffe [41] provides a comprehensive treatment of the subject.

In science, it happens quite often that the data gathered is of very high dimensionality. This can be inconvenient, because the computational cost and complexity rises with every additional dimension. Furthermore, the data in some dimensions is often highly correlated, so those dimensions add little information. Figure 3.6 shows a two-dimensional example where the two variables are clearly correlated.

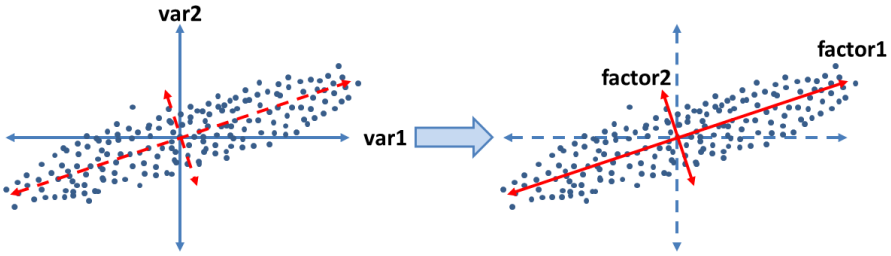


Figure 3.6: Dataset with two correlated variables

The principal components (PC) describe the directions in which a data set contains the highest variance. In our example, the first PC is clearly the diagonal direction shown in the diagram. The principal components are ordered by the amount of variation the data contains in their respective direction, with the first PC containing most and the last PC containing the least information. An important property of the principal components is that they are orthogonal. They form the basis vectors of a new coordinate system to which the data is transformed.

To find the principal components, several steps are necessary. First the mean has to be removed from the data.

$$\bar{S} = \frac{1}{q}\sum_{i=1}^{q} \tilde{S}_i \qquad \hat{S}_i = \tilde{S}_i - \bar{S} \tag{3.22}$$

Then these centered measurements $\hat{S}_i$ are combined into a matrix X

$$X = \begin{Bmatrix} \hat{x}_{1,1} & \hat{y}_{1,1} & \hat{z}_{1,1} & \hat{x}_{1,2} & \cdots & \hat{y}_{1,n} & \hat{z}_{1,n} \\ \hat{x}_{2,1} & \hat{y}_{2,1} & \hat{z}_{2,1} & \hat{x}_{2,2} & \cdots & \hat{y}_{2,n} & \hat{z}_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{x}_{q,1} & \hat{y}_{q,1} & \hat{z}_{q,1} & \hat{x}_{q,2} & \cdots & \hat{y}_{q,n} & \hat{z}_{q,n} \end{Bmatrix} \tag{3.23}$$

This matrix X contains our q observations (faces from the database that have been brought into correspondence) with each $3n$ variables (mean-stripped vertex coordinates $\hat{x}$, $\hat{y}$ and $\hat{z}$). The main step in calculating the Principal Components is done by a Singular Value Decomposition. It factorizes X into

$$svd(X) = U\Sigma V^T \tag{3.24}$$

The columns of the matrix V^T, the eigenvectors of X, are the principal components that we are interested in. Those Principal Components form the basis of the new coordinate system that our data is transferred to. The variance of the data σ^2 is proportional to the square root of the diagonal elements of Σ, the singular values of X. By convention, those singular values are in descending order. Therefore, the first Principal Component (first column of V^T) is the most descriptive, as the data contains the most information (variance) in its direction.

$$\sigma_i^2 = \left(\frac{\Sigma_{i,i}}{\sqrt{q-1}} \right)^2 \tag{3.25}$$

To calculate the coordinates of the original faces X in the space of the principal components, they are multiplied by the principal components V^T. The same coordinates result from weighting U by the singular values in Σ.

$$A = XV^T = U\Sigma \tag{3.26}$$

The main goal of PCA is to reduce dimensionality without loosing too much information. Before the transformation, the variance in the data is usually more or less equally distributed. That means the different faces contain a lot of redundancy. After the PCA transformation the most variance of the data is now represented by the first PC and the least by the last PC. So we can discard the last few $q - m$ Principal Components, while making as little error as possible for the reconstruction $\hat{y}$.

$$a = \hat{x}V_{1\ldots m}^T \qquad \hat{y} = aV_{1\ldots m} \tag{3.27}$$

Principal Component Analysis theoretically (assumptions at the end of this section) finds the optimal transform for a given dataset X and a given dimensionality m with respect to the error made in a least-squares sense.

$$e = \hat{x} - \hat{y} \tag{3.28}$$

How big this reconstruction error is for new data can be estimated using the singular values of the corresponding dimensions

$$E(\mathrm{e}) = \sum_{j=m+1}^{q} \sigma_j^2 \qquad \mathrm{e} = \|e\|^2 \tag{3.29}$$

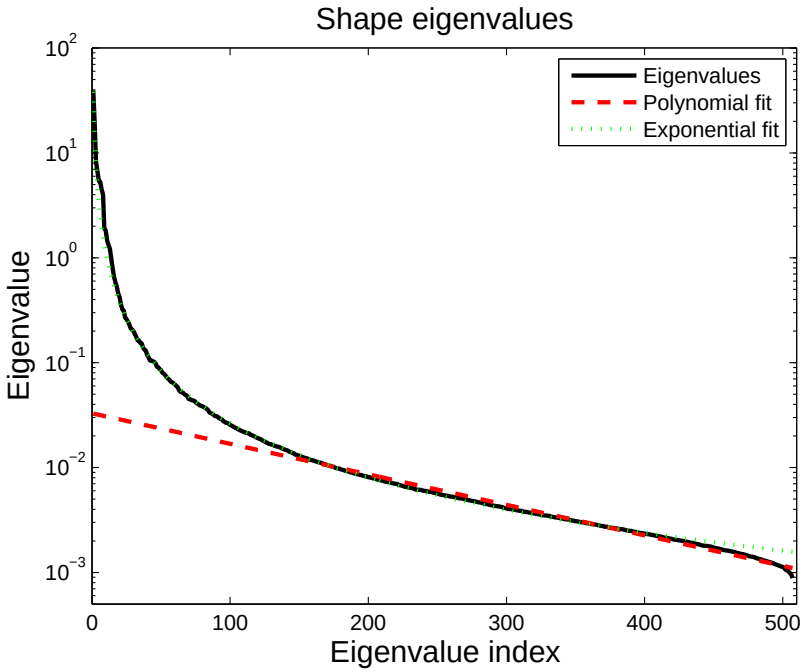
To find a tradeoff between information loss (due to leaving out the last PCs) and a lower dimensionality of the data representation, the usual method is to inspect the diagram of variances. Figure 3.7(a) shows LEV-diagram (logarithmic eigenvalues) of the shapes of our database.

An empirical method to chose the number of dimensions is to find the point where the decrease of variance changes from exponential to polynomial (i.e. the point where the LEV graph goes over in a more or less straight line) (see Jolliffe [41], chapters 6.1.3 and 6.2.2). But those methods rely only on the data and may not capture and encode the information that is needed for a specific task. This is the reason why Meytlis and Sirovich [54] try to determine the number of dimensions needed for face identification by making experiments with humans. They suggest the use of about 100 dimensions.

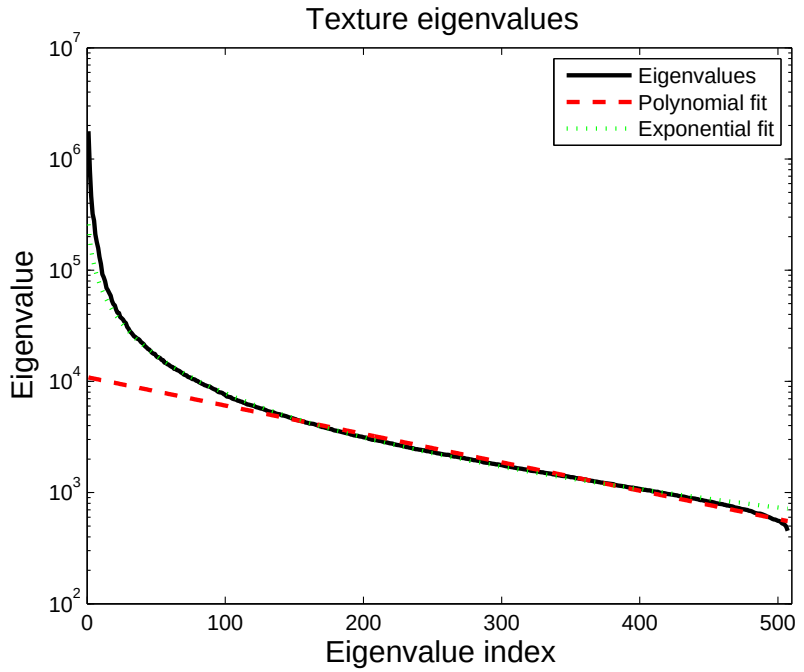
In the case of our 3D Morphable Model, PCA was performed separately on shape and texture of the registered 3D head scans (see Figure 3.8). The mean face (that is subtracted from the dataset before performing the SVD) is shown at the left. The variance of the data decreases sharply after a few components. The first and second principal component are shown also in Figure 3.8. The first PC of texture gives the face either a light or a dark complexion, whereas the second texture PC surprisingly models the gender of the face. The main shape principal component varies the "fullness" of the face and the size of the nose, and the third component generates narrow or round faces. A new face ($\tilde{S}_{new}$, $\tilde{T}_{new}$) can be transformed into the PCA-Face-Space by removing the mean from it and multiplying it with the m first model components S_i that are stacked in the matrix $V_{1\ldots m}^T$.

$$\alpha_{new} = (S_{new} - \bar{S})V_{1\ldots k}^T \tag{3.30}$$

PCA is based on several premises. The assumption of linearity is made, and if the data



(a)



(b)

Figure 3.7: Logarithmic eigenvalue diagram of both (a) shape and (b) texture

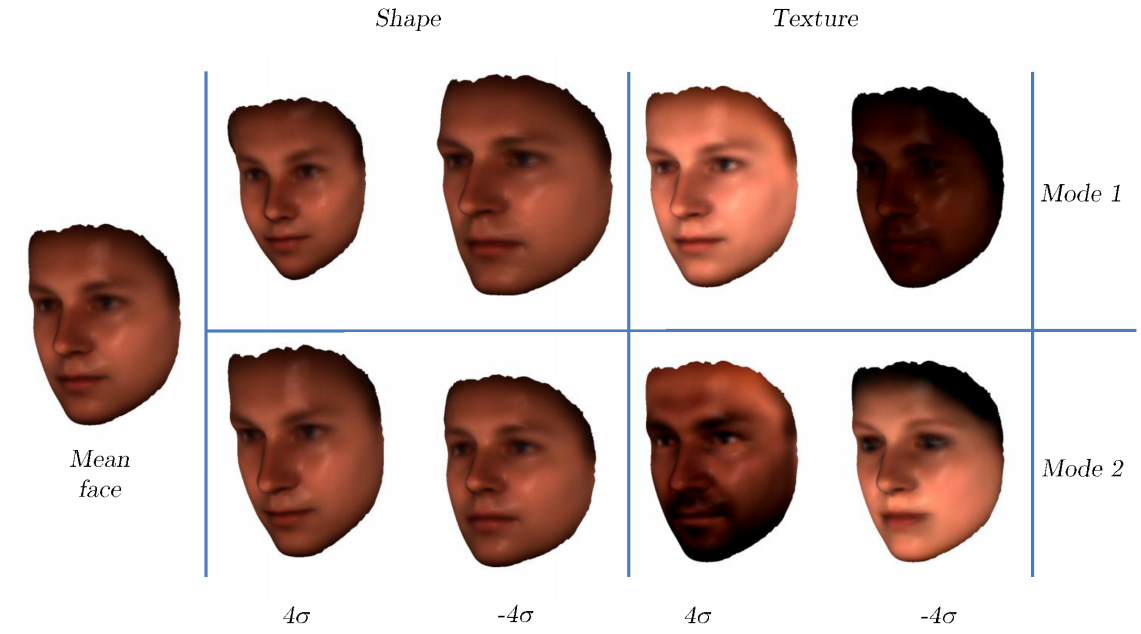
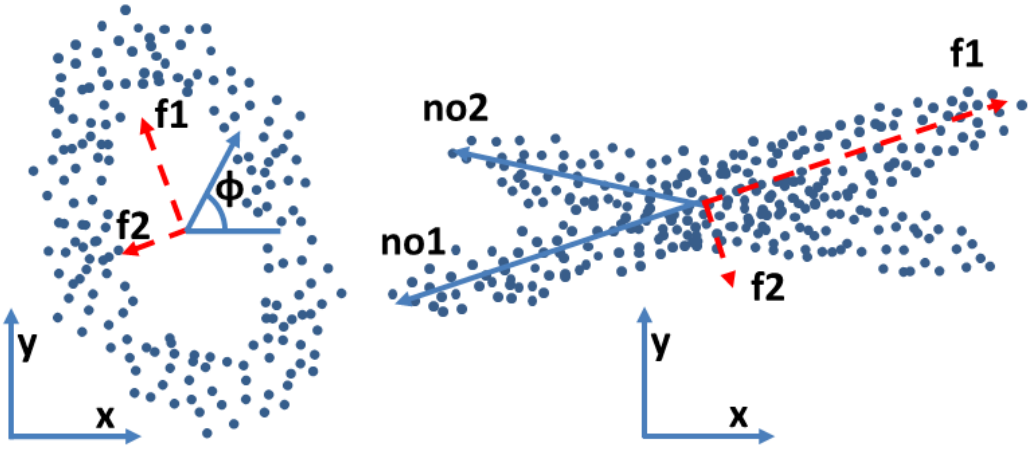


Figure 3.8: Overview over the first two modes of the generated 3D Morphable Model

does not support this assumption, the analysis can not produce useful results. Consider the measurements of a persons position on a ferris wheel - this could be easily described by the phase of the wheel - but is a nonlinear combination of the original basis (Figure 3.9(a)). Also if the data does not fulfill the orthogonality constraint, PCA has difficulties to generate a satisfying new basis (Figure 3.9(b)). The distribution of the data needs to be Gaussian, or else the transformation may not be optimal in a least-squares sense. Finally PCA fails too, if the importance of the data can not be derived from the variance.

But using PCA we were able to significantly reduce the dimensions in the “Face Space” described in section 3.4.5. This reduces both computational costs and storage size of the model, and emphasizes important modes. Other advantages of the Principal Component Analysis are that it does not have any parameters to adjust, and it gives a plausibility value for a face. How probable a face is can be calculated combining the variance (singular values) with the coordinates of a face in the principal components space.



(a) (b)

Figure 3.9: Cases for which PCA fails: (a) nonlinearly dependent variables, (b) non-orthogonal variables

3.6 Conclusion

In this chapter we show how to build a 3D Morphable Face Model from a database of raw head scans. Except for the preparation of the template face (mesh simplification, cropping) and the choice of some parameters (number of PCA modes, stiffness parameters for OSNRICP, ...) this is done in a completely automated manner - manually dealing with every single head scan (e.g. selection of landmark points, ...) is not necessary. Using the model, it is possible to generate realistic 3D models of human faces by choosing only a few parameters for shape and texture.

Chapter 4

Fitting a 3D Morphable Model to an Image

Contents

The goal of this chapter is to find a three-dimensional model that represents a face given in a two-dimensional image as close as possible. Per se this is an ill-posed problem, as during capture both depth-information is lost and also special lighting can result in unrecognizable faces. But by using a 3D Morphable Model it is often possible to recover this information. The "Fitting" in our 3DMM framework tries to estimate the model coefficients α for shape and β for texture, along with the illumination of the scene and the pose of the head with respect to the camera. This is done in an Analysis-by-Synthesis manner. Starting with an initialization (the mean face), the actual estimate of the face is rendered (section 4.1). The residual between the input image and the rendered face (section 4.2) is calculated not on the whole face, but only on a randomly selected subset of the face triangles. This randomized residual is derived with respect to the shape and texture parameters, illumination and pose (section 4.3). An optimization routine (section 4.4) uses those derivatives and updates the model and rendering parameters, iteratively

generating a model that approximates the input face more and more. For a further refinement of the generated face, an illumination-corrected texture (last section, 4.1.5) is extracted from the input image and blended with the estimated model.

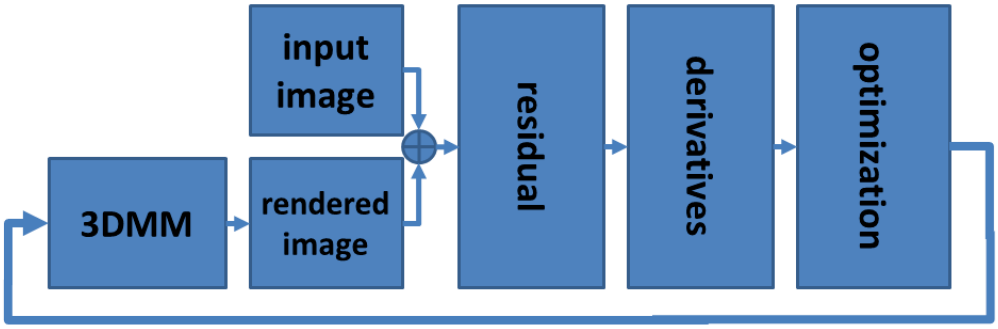


Figure 4.1: Structure diagram of 3DMM fitting process

4.1 Rendering Process / Image Synthesis

The Rendering Process generates a two-dimensional pixel image from a set of 3D vertices under some given conditions. Its first step is the Modeling Transformation, where the vertices are moved into the world coordinate system. Then the color of the 3d points is determined given some light and reflectance properties. The rendering process tries to model a pinhole camera, so a perspective projection makes more distant objects smaller in the Viewing Transformation, which also "squeezes" the image into the desired dimensions / format. Still represented by a list of vertices, the "Rasterisation" step transforms our model into the pixel image format. The color of all those pixels is determined in the final Fragment Shading step.

In general, rendering can be quite an involved topic, if the result is intended to be very realistic. From lens distortions over global illumination (ray tracing) up to more authentic reflectance models, there are a lot of details that one has to deal with. In this work, a simpler way was chosen because of easier mathematical handling (see derivatives), faster computation and also because it gives sufficient realistic results.

In our framework a face or head is represented by a shape vector S calculated from a linear combination of the shape components S_i

$$S = \sum_{i=1}^{m} \alpha_i S_i = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \end{bmatrix}$$

of n three-dimensional vertices given in local coordinates, a triangle mesh

$$M = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ v_{31} & v_{32} & \cdots & v_{3n} \end{bmatrix}$$

that defines the topology of the head and a texture vector T

$$T = \sum_{i=1}^{m} \beta_i T_i = \begin{bmatrix} R_1 & R_2 & \cdots & R_n \\ G_1 & G_2 & \cdots & G_n \\ B_1 & B_2 & \cdots & B_n \end{bmatrix}$$

containing a RGB color for every vertex (also calculated as a linear combination of the texture components T_i.

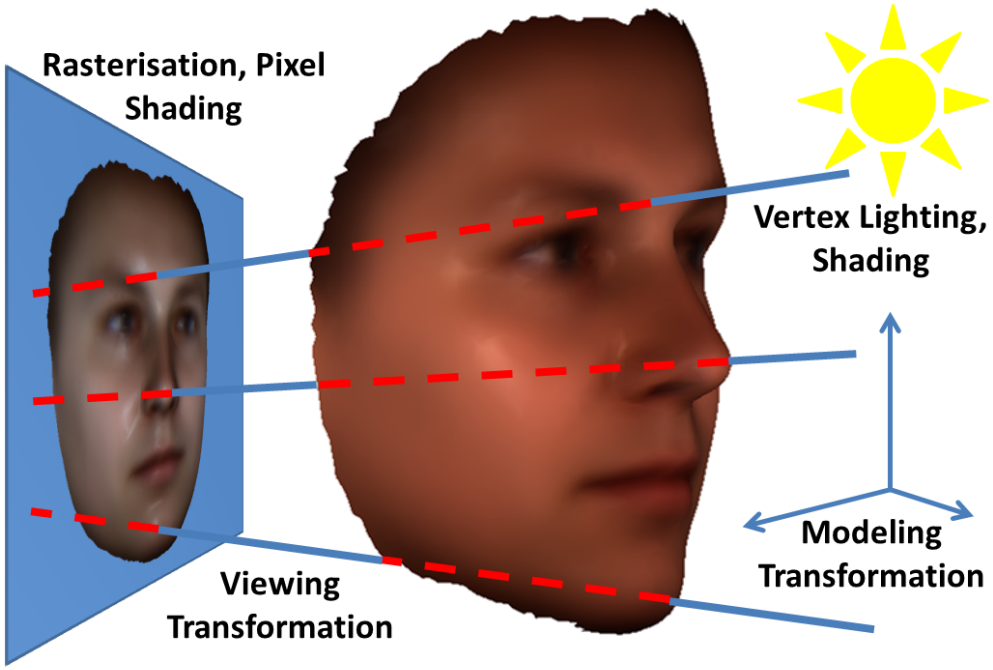


Figure 4.2: Structure diagram of rendering process

4.1.1 Modeling Transformation

The modeling transformation scales the face and moves it to a pose defined in world coordinates. It only affects the shape vector S. First, the size of the face is modified by the uniform scaling factor s - this operation can be represented as a homogeneous multiplication matrix SC.

$$SC = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Then the orientation is determined by three rotations. The first rotation around the x-axis is also known as yaw (ϕ_x), pitch (ϕ_y) denotes the second rotation around the y-axis, and the rotation with respect to the z-axis is called roll (ϕ_z). Those three rotations are applied to the scaled shape vector by multiplying it with the following homogeneous matrices.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi_x & -\sin\phi_x & 0 \\ 0 & \sin\phi_x & \cos\phi_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

$$R_y = \begin{bmatrix} \cos\phi_y & 0 & \sin\phi_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi_y & 0 & \cos\phi_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$R_z = \begin{bmatrix} \cos\phi_z & -\sin\phi_z & 0 & 0 \\ \sin\phi_z & \cos\phi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

The last step in the modeling transformation is the translation. It moves the scaled and rotated face to its final position. The translation again can be written as a matrix, with t_x, t_y and t_z being the movements along the x-, y- and z-axis respectively.

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Homogeneous coordinates allow the combination of transformations by multiplying their matrices. So the whole modeling transformation can be combined into one matrix that is multiplied with the vertices homogeneous coordinates

$$S_h = \begin{bmatrix} S \\ \mathbf{1}_{1\times n} \end{bmatrix} \tag{4.6}$$

to perform all of the scaling, rotation and translation steps at once.

$$MV = T \cdot R \cdot SC \qquad R = R_z \cdot R_y \cdot R_x \tag{4.7}$$

The final shape in the world coordinate system S_w is then obtained by

$$S_w = MV \cdot S_h \tag{4.8}$$

4.1.2 Vertex Lighting / Shading / Lighting / Reflection Models

For a realistic rendering, the next important step is to add light to the scene and model its interaction with the face. Two major approaches are distinguished: While local illumination only considers the interaction of the light that comes directly from the light source, global illumination also includes the light that is reflected from other objects.

The Blinn-Phong reflection model is a local illumination model easy to model and to compute, and it produces reasonable results. The color of a lit vertex is constituted of three terms.

$$T_i^{ill} = T_i^a + T_i^d + T_i^s \tag{4.9}$$

The first term T_i^a represents the influence of the ambient lighting on the ith vertex. It depends neither on the position of the viewer nor on the position of the light sources and models the ambient light that results from diffuse, scattered light in the scene. The resulting color is influenced both by the color of the ambient light $L^a = diag(L_R^a, L_G^a, L_B^a)$ and the texture color $T_i = [R_i, G_i, B_i]^T$.

$$T_i^a = \begin{pmatrix} L_R^a & 0 & 0 \\ 0 & L_G^a & 0 \\ 0 & 0 & L_B^a \end{pmatrix} T_i \tag{4.10}$$

The second term T_i^d models the diffuse properties of the surface. The diffuse lighting is independent of the viewer's position, but depends on both the direction of the light d and the surface normal n around the vertex. Lambert observed the diffuse part to be brightest, if the normal vector and the light direction are identical, and that no diffuse reflection happens if they are perpendicular. Between these two extremes the reflection intensity decreases with the cosine of the angle between light and surface normal vector (see equation 4.12). The color is influenced by the color of the directional light L^d and again the texture color T_i.

$$T_i^d = \begin{pmatrix} L_R^d & 0 & 0 \\ 0 & L_G^d & 0 \\ 0 & 0 & L_B^d \end{pmatrix} \Big(< n, d > T_i \Big) \tag{4.11}$$

$$< n, d >= |n||d| cos(\sphericalangle(n, d)) \tag{4.12}$$

The specular reflectance T_i^s is influenced by all three of light direction d, viewers direction v and surface normal n. An ideal specular (mirror-like) surface would reflect the light only when the incidence angle is the same as the emergent angle: $\sphericalangle(d, n) = \sphericalangle(v, n)$. Rougher surfaces tend to have broader reflection cones. The specular coefficient ν describes how sharp the drop-off is. So for dull surfaces this parameter is rather small, and bigger for more mirror-like surfaces. In the original Phong model [65] the specular reflectance term is modeled by

$$T_i^s = \begin{pmatrix} L_R^d & 0 & 0 \\ 0 & L_G^d & 0 \\ 0 & 0 & L_B^d \end{pmatrix} \Big(k_s < r, v >^{\nu} 1_{3\times 1} \Big) \tag{4.13}$$

where the angle of the emergent light $r = 2 < n, d > n - d$ is calculated exactly. In contrast to this original model, the "halfway vector" $h = \frac{(d+v)}{\|d+v\|}$ has been introduced by Blinn [16] for a more efficient computation if light and viewer are placed at infinity. Nowadays, this method is still widely used in basic 3D renderings (e.g. OpenGL) and also used in our framework.

$$T_i^s = \begin{pmatrix} L_R^d & 0 & 0 \\ 0 & L_G^d & 0 \\ 0 & 0 & L_B^d \end{pmatrix} \left(k_s < h, n >^{\nu} 1_{3\times 1} \right) \tag{4.14}$$

The specular material k_s describes how big the contribution of the specular reflection to the overall vertex color. Its color is determined only by the color of the directional light L^d.

In Figure 4.3 the influence of all three terms can be observed both combined and separately.

Figure 4.3: Ambient + Diffuse + Specular Component = Blinn-Phong reflection model

4.1.3 Viewing Transformation, Projection and Viewport Transformation

The Viewing Transformation moves the scene into the camera coordinate system. It only includes translations and rotations. In the framework, this step is omitted as the world and camera coordinate system are the same, with the camera located at the origin looking in the direction of the negative z-axis.

Projection is the process of mapping a three-dimensional model on a two-dimensional surface. Various techniques have been developed to model the reality more or less closely. The simplest approach is the orthographic projection, where all lines of projection are orthogonal to the projection plane. The results are not that realistic, but the method is often used in technical drawings because it preserves parallel lines.

Using perspective projection, the lines of projection emanate from the center of projection (focal point). As a result, distant objects are smaller than near objects. This model resembles more closely our experience (pin-hole camera, human eye). The distance

between the projection plane and the focal point is called focal length.

In computer graphics, projections are realized by transforming the viewing volume (frustrum) to the unit cube and omitting the z-coordinates. For the perspective transformation, the homogeneous multiplication matrix that transforms the frustrum to the unit cube is the following:

$$PR = \begin{bmatrix} \frac{f}{a} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{zN+zF}{zN-zF} & \frac{2zNzF}{zN-zF} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{4.15}$$

The viewing frustrum is defined by the field of view $fovy$ in the y direction ($f = \cot(fovy/2)$) and by the ratio of width (x direction) to height (y direction) a. In z-direction this pyramid is truncated by the planes located at zN (near plane) and zF (far plane). Primitives outside of the volume are not displayed on the final projection, and those on the border are clipped against it.

For displaying the scene in a window on a screen, the viewport transformation includes a translation along the x- and y-axis ($t_{x,offset}$, $t_{y,offset}$) and scaling (v_x, v_y). All four parameters are already in pixel units.

$$VP = \begin{bmatrix} v_x/2 & 0 & 0 & v_x/2 + t_{x,offset} \\ 0 & v_y/2 & 0 & v_y/2 + t_{y,offset} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

To finally obtain the coordinates in the Cartesian system again, every vertex $\mathbf{x}_{h,i}$, which is still represented in the homogeneous coordinate system, is normalized by its fourth homogeneous component w_i:

$$\mathbf{x}_i = \text{norm}\left(\mathbf{x}_{h,i}\right) = \begin{bmatrix} \frac{x_i}{w_i} \\ \frac{y_i}{w_i} \\ \frac{z_i}{w_i} \end{bmatrix} \qquad \mathbf{x}_{h,i} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ w_i \end{bmatrix} \tag{4.17}$$

All the previously explained transformations are applied to the original shape S_h of the head and summarized in the following function:

$$P(\alpha, \delta) = \text{norm}\,(VP \cdot PR \cdot MV \cdot S_h) \tag{4.18}$$

The function P describes the combination of modeling, projection and viewport transformation and a following homogeneous coordinate normalization. These transformations are applied to the vertices of the head using matrix multiplication. The shape, as detailed in the previous chapter, is generated by $S = S(\alpha) = \bar{S} + \sum_{i=1}^{n} \alpha_i \hat{S}_i$ and extended to homogeneous coordinates. The parameter δ is a stacking of the rotation angles ϕ_x, ϕ_y and ϕ_z, the translation parameters t_x ,t_y and t_z, the scale s, the projection parameters $fovy$, a, zN and zF and the viewport parameters v_x, v_y, $t_{x,offset}$ and $t_{y,offset}$.

4.1.4 Rasterisation

The rasterisation step finally transforms the vertex list into a pixel image. For every pixel it is determined to which triangle it belongs and where its relative position in the triangle is. As our face is already defined as a set of triangles, every pixel has to be checked against all triangles. This method called Ray Casting is very time and resource consuming. Therefore in Computer Graphics the rasterisation usually is done not in image order (pixel by pixel) but in object order (triangle by triangle).

A popular rasterisation algorithm is the scanline conversion. It is somewhere in between image and object order as a scanline moves down the rows of the image and is intersected with the triangles. Pixels between consecutive intersections belong to the respective triangle and are filled with the adequate color in the next step.

As a lot of different problems can occur in this step (triangle border, sub-pixel-size primitives, aliasing, ...), a lot of research and development has been done on the topic. For a more-in-depth treatment the reader is referred to [29].

In the rasterisation step, also the visibility of the triangles needs to be determined. One of the standard algorithms, the z-buffer-algorithm, also stores for every pixel the depth-value of the corresponding primitive. If another primitive would correspond to the same pixel, the depth value in the z-buffer is compared, and if the new primitive is closer to the camera than the already stored one, the pixel gets updated. Another algorithm renders primitives in back-to-front-order. Therefore, closer objects are automatically painted over the more distant ones.

Our heads are non-convex objects, and depending on the position of the light, parts of the face (nose, etc.) cast shadows over other parts. In the shaded regions, the color of the pixels is governed only by the ambient term of equation 4.9. Those regions can be

determined using a modified z-buffer-algorithm, where a fictive camera is placed at the position of the light. The areas that are not visible by the fictive light camera are the ones in shade.

4.1.5 Pixel Shading, Illumination Correction

Shading is the process of calculating the pixel colors. Up to this step, it is known to which triangle a pixel belongs, and also the colors of the vertices have been determined. For coloring all the pixels, there exist three main shading algorithms. Using flat shading, each pixel corresponding to a triangle gets the same color - the mean of the three vertex colors. When this technique is used for objects with non-flat surfaces, they may look faceted, as the triangle borders are visible as hard color changes. Gouraud shading avoids those sharp edges by interpolating the corner vertex colors. A pixels location in the triangle is calculated and then the color is bilinearly interpolated. This technique produces smoother looking objects compared to flat shading whilst still being reasonable fast. The disadvantage of Gouraud shading is that, as lighting is only calculated at the vertices, some special specular highlights (small ones in the middle of a triangle) are not rendered correctly and appear to be varying in intensity when moving. This problem is addressed by Phong shading (not to be confused with Phong lighting). The algorithm recalculates the complete lighting not only at the vertices but for every pixel by interpolating the surface normals over the triangle.

Further illumination correction for the now rasterized image of the face is necessary as real images often vary in contrast and have different color gains and offsets. For example images captured under the light of fluorescent lamps often have a blue tint. Grayscale images, as another example, have no color contrast.

$$T^{illc} = MT^{ill} + o \tag{4.19}$$

$$M = \begin{bmatrix} g_r & 0 & 0 \\ 0 & g_g & 0 \\ 0 & 0 & g_b \end{bmatrix} \cdot \left[cI + (1-c) \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \\ 0.3 & 0.59 & 0.11 \end{bmatrix} \right] \tag{4.20}$$

g_r, g_g and g_b are the gains for red, green and blue, $o = [o_r o_g o_b]$ is the color offset for each of the color channels and c varies the color contrast of the image. The luminosity coefficients 0.3, 0.59 and 0.11 are chosen for typical RGB primary colors.

At the end of the whole rendering process, we obtain a two-dimensional pixel image

from a three-dimensional head model seen under a defined pose with specified illumination. In the next section this image is compared to a real input image.

4.2 Residual Function

For finding the model parameters of a 3D face that resembles an input image as close as possible, the first step is to define a (dis)similarity measure. This measure allows the algorithm to judge the quality of the current parameter estimate by comparing the input image and the generated image.

It is defined as the sum of squared differences between input image and the rendered face based on current model parameters. The first intuitive approach is to calculate this error in the domain of the two-dimensional input image for all visible face pixels H.

$$e_c(x) = I^{in}(x) - T^{illc}(P^{-1}(x, \alpha, \delta), \alpha, \beta, \gamma) \tag{4.21}$$

$$E = \sum_{x \in H} \parallel e_c(x) \parallel^2 = e_c^T(H) e_c(H) \tag{4.22}$$

$T^{illc}(\tilde{x}, \alpha, \beta, \gamma)$ is the color of a 3D point $\tilde{x}$ that assigned to it during the synthetic rendering. $P^{-1}(x, \alpha, \delta)$ is the inverse projection that, for a given 2D pixel x, results to the location on the 3D head which is rendered to that pixel. This formulation brings up several difficulties that, although not making it impossible to work with nevertheless make it more involved. First the computation of the domain of head pixels H in the synthetic image is required to calculate the inverse projection. Further difficulties occur in the calculation of the inverse projection derivatives. By simply calculating the error measure in the 3D vertex domain instead of the 2D image space, most of the foregoing problems are avoided. Furthermore to obtain a cleaner formula for the derivative, the error function is scaled by a factor of $1/2$.

$$e_c = I^{in}(P(u, \alpha, \delta)) - T^{illc}(u, \alpha, \beta, \gamma) \tag{4.23}$$

$$E = \frac{1}{2} \sum_{u \in U} \parallel e_c(u) \parallel^2 = \frac{1}{2} e_c^T(U) e_c(U) \tag{4.24}$$

U is the set of vertices u that are visible under the current parameters and would get rendered in the synthetic image. $I^{in}(P(u, \alpha, \delta))$ represents the input image back-warped

into the domain of the three-dimensional head under the actual parameters. This formulation is fundamentally the same as 4.21, but it is evaluated at different locations (vertices u rather than pixel centers H). The projection function $P(u, \alpha, \delta)$ usually does not result in integral pixel locations, so the image I^{in} is sampled using bilinear interpolation.

The illuminated texture function $T^{illc}(u, \alpha, \beta, \gamma)$ takes both the vertices $S(\alpha)$ and their color $T = T(\beta) = \bar{T} + \sum_{i=1}^{n} \beta_i \hat{T}_i$ as arguments. In the parameter γ, all the illumination variables are pooled.

The computation of the preceding formula (and especially its derivatives in the next section) is still expensive. It is evaluated for all visible vertices on one hand, and on the other hand, the normal of each vertex depends on all surrounding triangles. Therefore two further simplifications are introduced: First, the error measure in equation 4.24 is computed at the center of the triangles instead at the vertices. This simplifies the calculation of the normals and especially the deduction of the formulas derivatives. Second, as using all visible triangles introduces a lot of redundance, only a randomly selected subset is used in the determination of the residual. So the computational cost of the error function is reduced significantly. Another advantage of selecting only the subset (easier optimization) is explained in chapter 4.4.3.

4.3 Derivatives

As we want our generated face to resemble the input image as close as possible, the error measure 4.24 has to be minimized. The derivative of the error measure can be used to guide the optimization described in the next chapter. The derivatives are deduced with respect to shape, texture, pose and illumination parameters, as all of them need to be adjusted. Only an overview is given here, some more details can be found in Appendix A.

4.3.1 Shape

The shape parameters do not only influence the first part $I^{in}(P(u, \alpha, \delta))$ of the residual function, but also the second part $T^{illc}(u, \alpha, \beta, \gamma)$, because the illumination depends on the normal vectors, which themselves change as the shape is deformed.

So the derivative of the error measure 4.24 with respect to the shape components α is the following

$$\frac{\partial E}{\partial \alpha} = \frac{1}{2}\frac{\partial (e_c^T e_c)}{\partial \alpha} = \frac{1}{2}(\frac{\partial e_c^T}{\partial \alpha} e_c + e_c^T \frac{\partial e_c}{\partial \alpha}) = \frac{\partial e_c^T}{\partial \alpha} e_c \tag{4.25}$$

For the derivative of the cost function, it is necessary to use the chain rule

$$\frac{\partial e_c}{\partial \alpha} = \frac{\partial I^{in}(P(U,\alpha,\delta))}{\partial \alpha} - \frac{\partial T^{illc}(U,\alpha,\beta,\gamma)}{\partial \alpha} \tag{4.26}$$

$$\frac{\partial I^{in}(P(U,\alpha,\delta))}{\partial \alpha} = \frac{\partial I(\bar{x},\bar{y})}{\partial(\bar{x},\bar{y})}|_{P(U,\alpha,\delta)} \frac{\partial P(U,\alpha,\delta)}{\partial(U)}|_{S(\alpha)} \frac{\partial S(\alpha)}{\partial \alpha} \tag{4.27}$$

The first part of the derivation are the image gradients in both $\bar{x}$ and $\bar{y}$ direction of the input image, evaluated at the locations of the triangle centers of the head image coordinates rendered with the current parameter estimates α and δ. The next part derived using the chain rule is the derivative of the transformation P (equation 4.18) with respect to the triangle centers in 3D. The last one is the derivative of the shape derived with respect to the shape parameters α.

In the derivative of the illuminated texture the ambient term falls out completely, as it only depends on the parameter β. It is mainly based on the derivative of the normal vectors.

$$\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \alpha} = M\left(L^d \frac{\partial < n,d >}{\partial \alpha} T + L^d k_s \nu < h,n >^{(\nu-1)} < h, \frac{\partial n}{\partial \alpha} >\right) \tag{4.28}$$

4.3.2 Texture

The only terms that vary with the texture parameters are the ambient and the diffuse illumination. So the simple derivation of the cost function with respect to the parameters β is

$$\frac{\partial e_c}{\partial \beta} = -\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \beta} \tag{4.29}$$

$$\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \beta} = M\left(L^a \frac{\partial T}{\partial \beta} + L^d < n,d > \frac{\partial T}{\partial \beta}\right) \tag{4.30}$$

4.3.3 Pose

Both the $I^{in}(P(u,\alpha,\delta))$ and the $T^{illc}(u,\alpha,\beta,\gamma)$ part of the residual function depend on the pose parameters ρ. For the first part it is obvious that it depends on the pose parameters, but also the illuminated texture varies with changes in pose. If the light direction is given in world coordinates, the diffuse reflection term is influenced by a pose change, as the angle

between the normal vectors and the light direction changes. The specular term depends also on the viewers direction with respect to the face - also this relation changes as the pose is altered. Here the derivatives with respect to the rotation angles are shown. For the other pose parameters, they are similar.

$$\frac{\partial e_c}{\partial \rho} = \frac{\partial I(\bar{x},\bar{y})}{\partial(\bar{x},\bar{y})}|_{P(\rho,S(\alpha))} \frac{\partial P(\rho,x,y,z)}{\partial \rho}|_{S(\alpha)} - \frac{\partial T^{illc}}{\partial \rho} \tag{4.31}$$

$$\frac{\partial P(\rho,x,y,z)}{\partial \rho} = \begin{bmatrix} \frac{\partial P^{\bar{x}}}{\partial \rho_x} & \frac{\partial P^{\bar{x}}}{\partial \rho_y} & \frac{\partial P^{\bar{x}}}{\partial \rho_z} \\ \frac{\partial P^{\bar{y}}}{\partial \rho_x} & \frac{\partial P^{\bar{y}}}{\partial \rho_y} & \frac{\partial P^{\bar{y}}}{\partial \rho_z} \end{bmatrix} \tag{4.32}$$

$$\frac{\partial T^{illc}}{\partial \rho} = M\left(L^d < \frac{\partial n}{\partial \rho}, d > T + L^d k_s \nu < h, n >^{(\nu-1)} < h, \frac{\partial n}{\partial \rho} >\right) \tag{4.33}$$

4.3.4 Illumination

Just the illuminated corrected texture term T^{illc} has to be concerned for changing lighting conditions. Here is its derivation with respect to the direction of light d. The derivatives for the other illumination parameters are similar.

$$\frac{\partial T^{illc}}{\partial d} = M\left(L^d \frac{\partial < n,d >}{\partial d} T + L^d k_s \nu < \frac{(d+v)}{\parallel d+v \parallel}, n >^{(\nu-1)} < \frac{\frac{(d+v)}{\|d+v\|}}{\partial d}, n > 1_{3\times 1}\right) \tag{4.34}$$

4.4 Optimization

The goal of our framework is to minimize the residual E (equation 4.24) between the given input image and our estimated model. Within the vocabulary of optimization this function is called the "objective function". Optimization is the process of finding the parameters for the objective function where it reaches a minimum (or maximum). Various algorithms exist for different types of objective functions (linear vs. non-linear, discrete vs. continuous, ...) and side criteria (robustness, global vs. local optimization, ...).

On the residual function described in section 4.2 several observations can be made: It is a very high-dimensional function - the residual depends on the pose parameters (ρ, 7 dimensions), illumination parameters (γ, up to 18 dimensions), and up to several hundreds of shape and texture parameters (α, β). The residual is also not linear (illumination, input image, SSD, ...) and not convex.

In this section first the Gradient Descent optimization (section 4.4.1) is described,

and then an overview over alternative algorithms is given. Stochastic optimization is introduced in section 4.4.3, and finally some specialties of the 3DMM fitting are covered in the last part.

4.4.1 Gradient Descent

One of the simplest optimization techniques is the gradient descent algorithm. For finding a local minimum x_{min} of a function $f(x)$, iterative steps into the direction of the steepest gradient at the current position x_i are taken.

$$x_{i+1} = x_i - \lambda * \nabla f(x_i) \tag{4.35}$$

$\nabla f(x)$ is the gradient of the objective function evaluated at x_i. An important parameter in this algorithm is the step size λ. Its value is always a trade-off between speed of convergence and stability. With a small λ, small steps towards the minimum cause a slow convergence. But if the step size is chosen too big, the steps can produce an oscillating behavior.

Usually, the iterations are halted when the difference of the objective function at two consecutive parameter sets is falling below a fixed threshold, and the current parameter set is assumed to be roughly the minimizer. An alternative is to stop the optimization after a predetermined number of steps within which the minimum is reached under all starting conditions.

The gradient descent algorithm is very simple, but also has some disadvantages. It is not guaranteed that the global minimum is found - the algorithm very often gets trapped in a local minimum. Only if the objective function is convex, the convergence to the global minimum is ensured. The speed that the minimum is approached with is only linear, and can be even slower for some pathological function (e.g. in a valley of the Rosenbrock function)

4.4.2 Alternative Optimization Algorithms

There exists a vast amount of optimization algorithms for different areas of application, with various advantages and shortcomings. Only a few of them are mentioned here.

Newton's Algorithm is based on the observation, that the first derivative is zero at an extrema of a function. Assuming that the objective function can be approximated by a quadratic function, the Taylor expansion for the gradient is $\nabla f(x + \Delta x) = \nabla f(x) +$

$\Delta x H(f(x))$. $\nabla f(x)$ is again the gradient, and $H(f(x))$ is the Hessian (the second-order partial derivatives) of the objective function evaluated at x. The optimal parameter update can be computed as

$$x_{n+1} = x_n - \lambda[H(f(x_n))]^{-1}\nabla f(x_n) \tag{4.36}$$

This approach shows quadratic convergence near the minimum, but when starting far away from the solution, the Newton's method might not even find the final solution. Another problem is the expensive computation of the inverse Hessian.

Quasi-Newton As the computation of the Hessian in the Newton's algorithm is expensive, several algorithms try to avoid its update. Instead the Hessian is approximated from function evaluations and the Jacobian around the current point. One example of this family of algorithms is the L-BFGS algorithm [57].

Gauss-Newton is a modification of Newton's method that can be applied to a least-squares-problem when the objective function is given as a sum of squared function values. A first advantage of the formulation is that the costly Hessian does not need to be computed. And near the solution it has, similar to the Newton's method, quadratic convergence, whilst away from the solution, it takes steps like the steepest descent method with a bigger region of convergence. Nevertheless, it is not guaranteed to converge to a (local) minimum.

$$x_{n+1} = x_n - (\nabla f(x)^T \nabla f(x))^{-1} \nabla f(x)^T f(x) \tag{4.37}$$

Levenberg-Marquardt is a method that interpolates between Gradient Descent and Gauss-Newton. It has the advantage of being more robust than the Gauss-Newton algorithm, trading a bigger region of convergence for a lower convergence speed. A damping value is adjusted at each iteration, and regulates the influence of the Gradient Descent and the Gauss-Newton influence according to the previous steps made. It has first been introduced by Levenberg [46] and rediscovered and improved by Marquardt [52] and belongs to the trust-region approaches.

Lipschitz Optimization An optimization method that finds the global minimum of a function is the Lipschitz optimization. It basically optimizes the brute-force method of sampling the function over the complete parameter domain and is based on observations

on the objective function. The iterative method evaluates the objective function and compares the resulting value with a previously determined temporary minimum. Using the knowledge about the maximum gradient the function can have (requires smoothness), an area around the currently evaluated parameter can be guaranteed to not contain a minimum smaller than the temporary minimum. An overview over this branch-and-bound method is given by Hansen et al. in [33].

In this framework the Lipschitz optimization has been used to make a rough pose estimation for an input image is given. The implementation loosely follows the ideas given in [47].

4.4.3 Randomized Optimization

Stochastic (or randomized) optimization was introduced to deal with cost functions that contain a lot of local minima (as does the residual formulated in this framework, see section 4.2). By adding random noise to the objective function, the optimization process is (metaphorically speaking) being "pushed" out of the unwanted basins that we do not want to get stuck within.

A common method that has been proposed by Viola in his thesis [76] to add this noise to the objective function is to use only a subset of the available data points. The error made by leaving out a part of the datapoints can be used as this noise. In order to avoid overfitting of the optimization w.r.t. special subset, it is newly selected every or every few iterations.

In the case of the 3D Morphable Model, we randomly choose a set of triangles from those that are visible under the current parameters. The difference between the full and the reduced residual is the random noise that helps the optimization to avoid the local minima. A new subset of triangles has to be selected every few iterations to avoid overfitting. Ideally, this subset would be renewed every iteration, but its selection is computationally expensive. For every triangle, its visibility needs to be checked. So the number of iterations, after which a new combination of contributing triangles is randomly chosen, is a tradeoff between the point where overfitting happens (and no progress towards the final solution is made) and the time necessary to determine the new subset.

In our case this stochastic method has two further advantages: First, the contribution of all (visible) triangles would be redundant, and secondly, the computational effort is significantly lower if the cost function (and the gradient) is evaluated only at a few vertices.

Stochastic optimization sometimes does not take the optimal steps toward the min-

imum, and therefore has a convergence properties a bit slower than the equivalent "deterministic" method. Risk of overfitting exists if the selection of the datapoint subset is not optimal (non-uniform selection of datapoints) or to small. Also the stopping criteria has to be modified sometimes, as after a subset variation the optimized parameters may change again after coming to rest for a previous set.

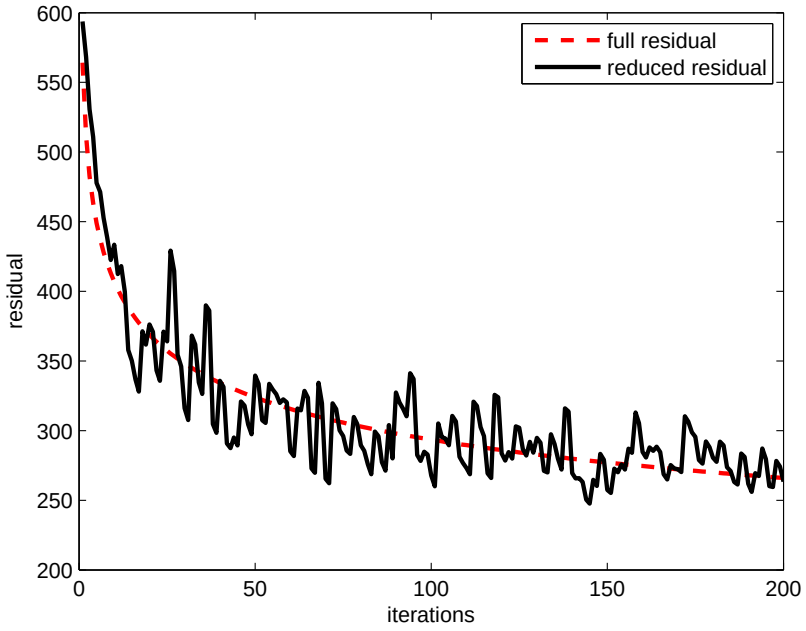


Figure 4.4: Stochastic vs. normal optimization example (reduced vs. full image space residual) (selection of new triangles every 2^{nd} iteration)

4.4.4 Further details on the optimization procedure

The objective function $E(\alpha, \beta, \rho, \gamma)$ of our 3D Morphable Model Framework has several peculiarities. One of the main observations is that the impact of pose and illumination on the residual are larger than the influence of shape and texture parameters. Therefore it is difficult to estimate correct values for α and β if the rough pose and light estimation is wrong. Also, pose and illumination have a high grade of interdependence. For this reason, a sequential optimization strategy was used to first search for rough pose and illumination parameters based on a mean shape and texture face. Only after that, the first ten shape and texture parameters are adjusted. Finally, for all parameters the optimal values are determined within the closer neighborhood.

The election of the triangles used to calculate the derivatives and guide the optimization process is weighted by their visible size. The probability for triangles in areas important for the distinctiveness of a face (nose, mouth, eyes) to be selected can be increased by adding a weight mask to them. When adjusting the rough shape (first ten components of α), another area of focus is on triangles at the silhouette (determined by the dot product between triangle normal and view direction).

4.5 Inverse Texture Extraction

Ideally, the 3D Morphable Model fits an exact three-dimensional face to a given 2D face image with all its peculiarities. But to do this successfully, all peculiarities have to be in the training set of the model, and the PCA needs to embrace and include them. This would result in a large and complex 3D Morphable Model. It can be observed that almost all of those specialties of individuals occur in the texture domain. So instead of using a complex model that captures all exceptional features, the input image is used to extract a corrected texture that obviously includes all those anomalies.

Inverse Texture Extraction first determines the location of all the visible vertices in the input image ($P(u, \alpha, \rho)$) and then inverts both illumination correction (4.1.5) and vertex lighting (4.1.2).

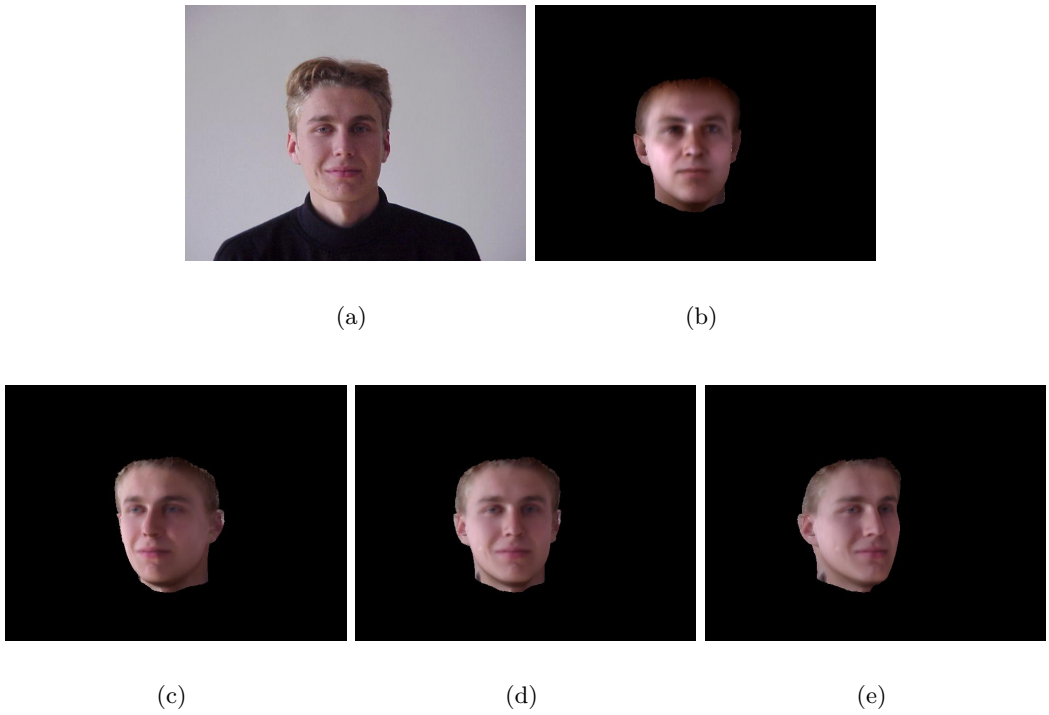
$$T^{ill}(u) = M^{-1}(I^{in}(P(u, \alpha, \delta)) - o) \tag{4.38}$$

$$T^{e}(u) = \frac{T^{ill}(u) - L^{d} k_s < h, n >^{\nu} 1_{3\times 1}}{L^{a} + < n, d > L^{d}} \tag{4.39}$$

The extracted texture T^e includes all visible peculiarities and fine details from the input image. It is illumination-invariant, and the face can be synthesized under any new light and pose conditions. It gives a more realistic 3D model of the given image (see Figure 4.5).

But as the texture can only be extracted for visible regions, the hidden parts of the face in the input image need to be improved otherwise. To avoid sharp borders between the extracted and the fitted texture, they can be interpolated based on the relation of normal vector and view direction. Furthermore the albedo can be mirrored from the visible side of the face, if there are no special features on this side - this is the case if the fitted and the extracted texture do not differ substantially.

Other problems occur if the estimated pose, shape or light parameters are wrong -



(a) (b)

(c) (d) (e)

Figure 4.5: (a) Subject from the CVL database and (b) the face fitted to it. The second row shows the fitted face with extracted texture at (d) the same pose, (c) a rotated yaw angle (-20°) and (e) another rotated yaw angle (20°)

then for example the background is included in the extracted texture, the real color is mis-estimated and misplaced (the texture of the nose is used for the mouth, etc.). Also the compact representation of a face only by its parameters α and β can not be used anymore with the extracted texture.

4.6 Conclusion

In this chapter, it is shown in detail how the parameters of a previously built 3D Morphable Model (chapter 3) are fitted to a given input image. This is done in an Analysis-by-Synthesis approach, iteratively adapting the parameters of the model and other conditions (light, pose) until the generated face resembles the template close enough. A stochastic variant of the gradient descent algorithm makes this optimization more robust and also speeds up the algorithm, and if a more detailed model is needed, an illumination-invariant texture is extracted from the input image. To judge the quality of the whole fitting approach, in the next chapter evaluations on several datasets are conducted.

Chapter 5

Evaluation, Experiments, Discussion

Contents

In computer vision usually it is hard to prove that an approach to solve a problem is completely correct. This is because the data we deal with is usually of very high dimensionality and has a lot of variance. But it is possible to demonstrate examples where a technique succeeds to solve the problem statement. And if not only a few examples but a large, realistic dataset is used for evaluation, those thorough tests can provide enough evidence that the algorithm works well for the specified task.

To judge how well an approach performs for a specific example, an error measure is needed. This makes the experiments comparable and provides the basis for a quantitative evaluation.

In this chapter, the building step of the 3D Morphable Model is given a close examination in the first section 5.1. Then the pose estimation capabilities of the 3D Morphable Model framework are tested (section 5.2), and also the shape and texture estimation for artificial and real 2D input images are inspected (section 5.3). Finally we show some details of the implementation (section 5.4).

5.1 3DMM evaluation

The different stages of building a morphable model are evaluated separately. On 8532 texture images of the head scan database (section 3.1) that were evaluated by the AAM face detector, 7370 face regions have been successfully detected. The scans where no face has been found are not used in the following stages, as there are abundant scans. One face detection needs about 1.4 seconds on a system with an Intel Core2 Duo CPU @ 2.6GHz (T9550) and 4GB of main memory on Microsoft Windows Vista x64 SP2. Figure 5.1 shows the detection of the face region with eyes and the mouth highlighted.

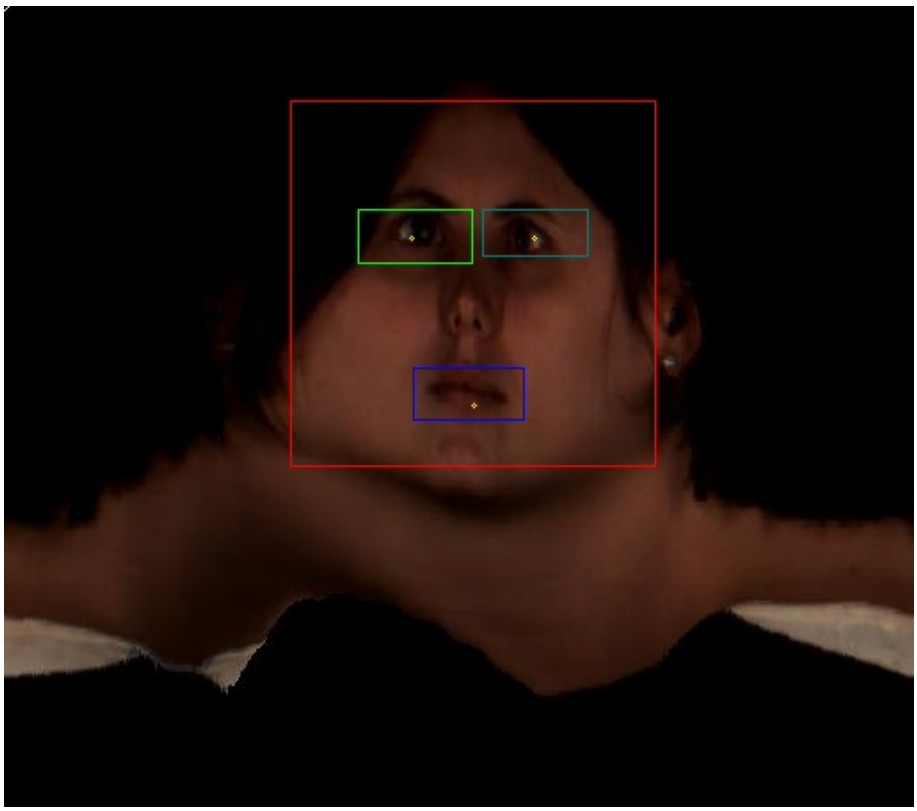


Figure 5.1: Face detection on the texture of a head scan

Then the Optimal-Step Non-Rigid ICP algorithm found dense 3D correspondence for the scans. One run of the Matlab implementation for a head with 230400 vertices against the template with 10424 vertices needs about 1 minute. The stiffness was lowered in steps of 20 from 100 to 10 and then with a step size of one down to $\alpha = 1$. Figure 5.2 shows the residual over the course of iterations.

The quality of the OSNRICP correspondence finding was evaluated using the recon-

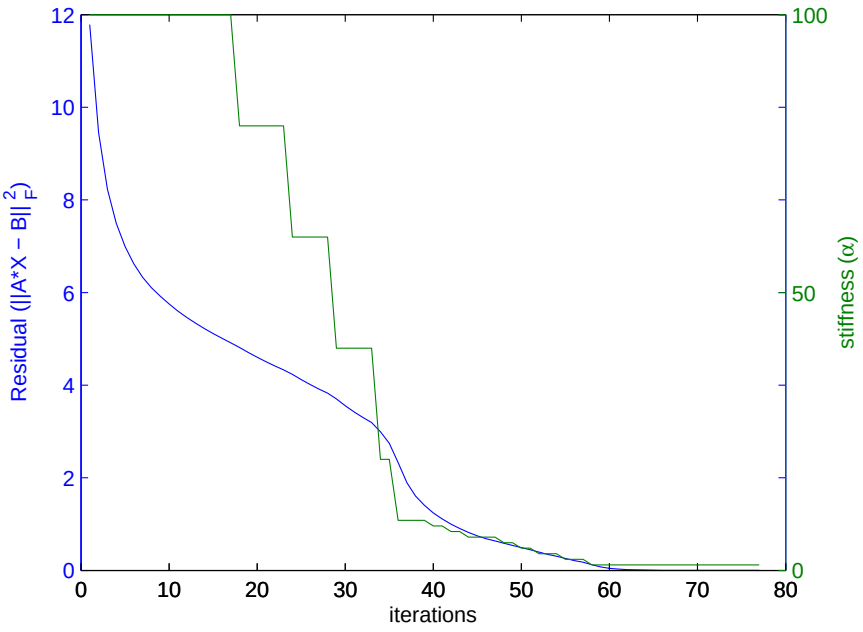


Figure 5.2: Residual and stiffness for the registration of one head scan.

struction error. Every face is approximated by forming a linear combination of all the other registered scans. The mean distance between corresponding vertices of the reconstructed faces and the original is at about 9e-5. By itself this reconstruction error is not very meaningful, but on one hand it can be compared with the minimum distance 1.6e-3 between two faces. On the other hand, if we observe the histogram of the reconstruction error (see Fig. 5.3), it can be seen that there are some clear outliers. A visual inspection of those faces for which a linear combination fails to model them, shows that the algorithm did not succeed to correctly register them with the template face. In the following step of constructing the 3DMM using PCA those faces are not included.

On the 507 scans that were successfully brought into dense 3D correspondence during the previous step, a Principal Components Analysis is performed to build the final 3DMM. For both of shape and texture the computation time of the principal components in Matlab is about 21s on the previously mentioned system.

PCA is used to reduce the dimensionality of data by discarding the least important components. The decision of how many components to discard is a tradeoff between loosing important information and lower dimensionality. In the case of our head scans, half of the minimal difference between two faces in this set serves as a hint on how much

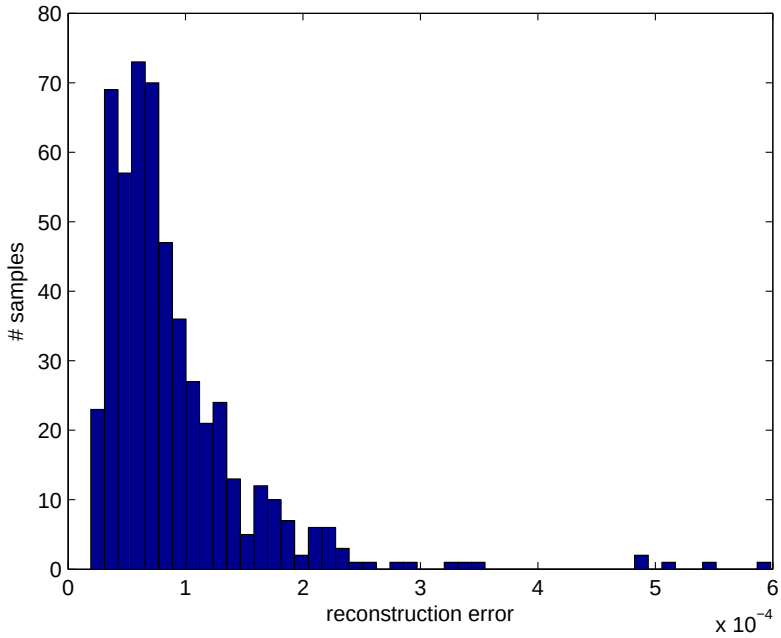


Figure 5.3: Histogram of the reconstruction error for all OSNRICP registered faces.

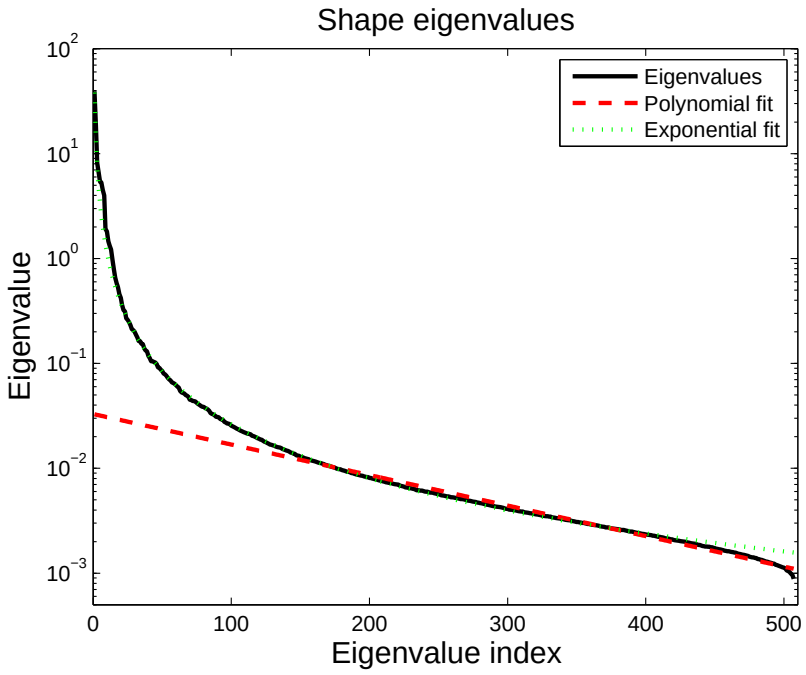


Figure 5.4: Plot of the logarithmic eigenvalues of the shape, with exponential and polynomial fitting.

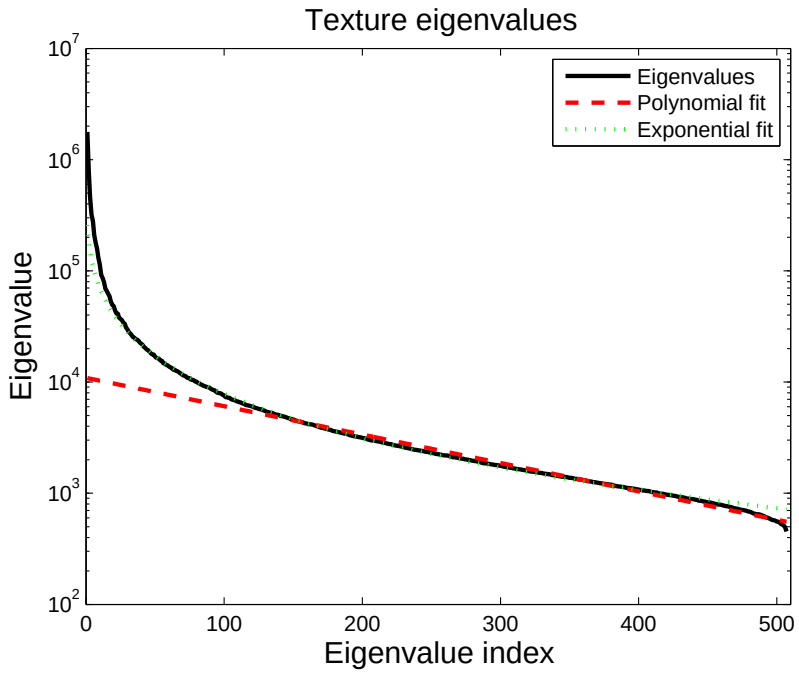


Figure 5.5: Plot of the logarithmic eigenvalues of the texture, with exponential and polynomial fitting.

information is needed to distinguish between two persons in this set. It can be compared to the estimated reconstruction error made when leaving out $N - n$ components.

$$\min_{\substack{i,j \in \text{faces} \\ i \neq j}} \frac{\| S_i - S_j \|^2}{2} \geq \sum_{j=n}^{N} \sigma_j^2 \tag{5.1}$$

$$\min_{\substack{i,j \in \text{faces} \\ i \neq j}} \frac{\| T_i - T_j \|^2}{2} \geq \sum_{j=n}^{N} \sigma_j^2 \tag{5.2}$$

The squared minimum distance between two faces in our set of 507 faces is 16.41. The expected reconstruction error for 27 Principal Shape Components lies with 8.14 just under half of this difference. So at least 27 PCs are needed to reconstruct a face sufficiently from our set of 507 persons and distinguish it from the others. For the texture modes, the minimal difference is 2043484, and the reconstruction error of 91 modes lies a bit under half of this value.

But as the main goal of our model is not discrimination but the reconstruction of a persons face, this minimal distance can serve only for orientation.

Another method is to analyze the LEV-diagram (logarithmic eigenvalues) (see Fig. 5.4,

Fig. 5.5). The cutoff usually is made at the point where the decrease of the eigenvalues changes from exponential to polynomial (i.e. the point where the LEV graph goes over in a more or less straight line) - see section 3.5. For the shape eigenvalues this happens after the 172^{th} eigenvalue, for the texture, the decrease is getting polynomial from the 161^{th} component on.

Nevertheless neither the minimal face difference compared to the PCA reconstruction error nor the LEV analysis consider the specific details and needs of our model. When fitting the model to an 2D input image for pose estimation, the number of PCA components used can be quite low - 10 shape and texture dimensions of the model are enough to approximate the face sufficiently for this application. For reconstruction of a visually plausible model, 100 shape and texture coefficients have shown to be adequate by visual inspection and comparison. With 100 shape coefficients, 97.8% of the total variance is kept, while the 100 texture components represent 87.29% of the texture eigenvalue energy spectrum.

A closer inspection of the first few PCA components shines light on some interesting details. The first texture component is related to the complexion of the face, while astonishingly the second mode is clearly related to the gender. On the other hand, the results of the shape components are easier to expect. They model the non-uniform scaling and shearing of the face geometry. With the predominance of those shearing and scaling components, probably also the small step in the LEV diagram of the shape components around the modes 5 to 10 can be explained.

The optimal step non-rigid ICP algorithm does a good job for establishing dense correspondence between the raw 3D scans. A disadvantage is the high computational effort, that furthermore does not scale with respect to the number of vertices of the template face. Although the possibility exists that the algorithm fails on a few scans, those exceptions can be easily figured out by inspection of the reconstruction error. Advantages of this approach are the smooth interpolation of registration for regions with ill-posed correspondence and the intrinsic capability to resample the surface and fill in missing areas.

Although not all prerequisites for the Principal Component Analysis are fully satisfied (linearity, etc.), it nevertheless generates a good 3D Morphable Model from the provided data. And furthermore, PCA extracted some interesting particularities from the dataset such as the “gender texture component” (2^{nd} texture component).

This successfully constructed model is used in the next two sections and evaluated along with the fitting algorithm.

5.2 Pose estimation

In the interaction between humans and computers, a very important task is the determination of the head pose. For the successful identification or verification of a face the position and orientation of a persons head in a captured image is a prerequisite. Using the Analysis-by-Synthesis algorithm detailed in chapter 4, in this section the 3D Morphable Model is used to estimate the orientation of faces in various databases.

5.2.1 Databases

CUbiC FacePix(30) [11], [48] is a freely available database that consists of 30 individuals. Each individual is represented by three sets of photographs. The first set shows the person with varying yaw angle from -90° to +90° (in incremental steps of 1°). Those images are taken under the illumination of two stationary diffuse light sources to simulate ambient lighting. The second and the third set are images with different lighting. For both, a spotlight is rotated around the person in one degree steps from the left to the right (-90° to +90°). While for the second set, the diffuse lights from set one are also added to the scene, the third set is only illuminated by the spotlight and therefore the face images in this set contain harsh shadowing.

USF Human ID 3D face database [13] includes recordings of 136 individuals captured using a Cyberware^{TM}laser scanner. The 3D laser scans consist of more than 90,000 vertices and 180,000 triangles. To evaluate the pose estimation capabilities of the framework, the 3D scans from this database are rendered with arbitrary pose and random textured background to 2D images. From the 50 first individuals and the combination of [-16° 0° 16°] yaw and [-8° 0° 8°] pitch angle, 450 test images are generated.

5.2.2 Error measures

During the evaluation of an algorithm for pose estimation, a measure to judge the quality of the estimate is needed. The mean absolute error is used for this task. It is calculated for both yaw and pitch angles comparing the ground truth GT with the estimated angle Φ. $E = |GT - \Phi|$. The ground truth for images often is not easy to establish - it is for example influenced by different head geometries. Easier is the determination of the relative pose between two images. By capturing a subject multiple times (with a moving camera or multiple cameras) from predefined angles, the difference of the pose of consecutive images

absolute yaw error				
mean	std	median	Q.25	Q.75
4.89	3.15	4.48	2.15	7.06

Table 5.1: Evaluations for the FacePix database. Mean, standard deviation, median, upper- and lower quartile of the absolute yaw angular error

is known exactly. The other possibility to establish a well-known ground-truth is to render images from 3D models. Here the parameters can be adjusted and are known exactly.

5.2.3 Evaluation

The evaluation of the framework for pose estimation has been done on two sets of images. First from the FacePix database the images with varying yaw angle are used as input. The input angles in range from -16° to +16° were covered in steps of 4° - so all in all the pose of 270 images has been calculated. No initialization was applied (except for setting the size of the faces once for all images in the database) - the optimization started at the middle of the input image with a rough light adaption. Then preliminary translation parameters were optimized, and in the next step also the three rotation angles are adjusted. In the two final parts of the optimization, also 10 shape and texture parameters are added to the set of the fitted parameters. Empirical observation has shown that in the stochastic optimization, a set of 500 triangles newly selected every 30 iterations provides a good tradeoff between computational effort and added random noise. The average runtime sums up to 33.5 seconds per face fitted.

For the cases with bad performance of the pose estimate, an inspection shows that either the estimate of the face size is wrong (Fig. 5.7(a), (b)) or that hair occludes a significant part of the face (Fig. 5.7(c)). But for some other cases visual observation shows that the fitted faces resemble the pose quite well (Fig. 5.7(d)) - here the problem of determining an “absolute” pose becomes obvious.

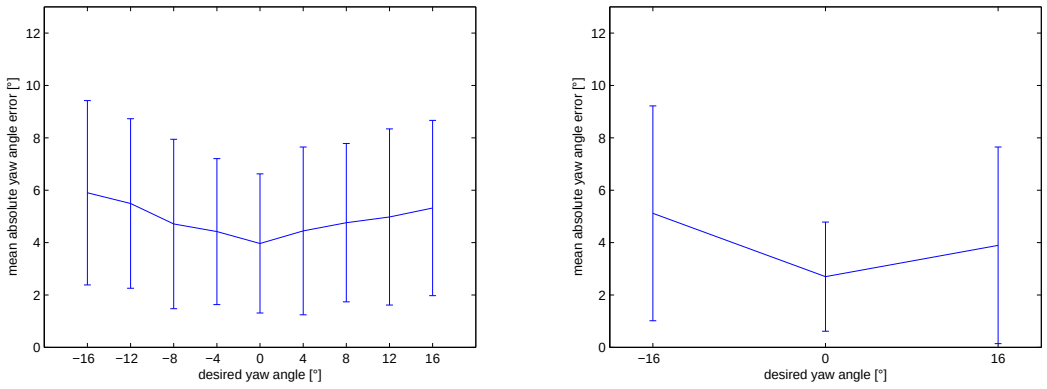
One example of the FacePix database and the fitted 3D heads are shown in Figure 5.6.

The same procedure has been applied to 450 image renderings of the USF Human ID database with combinations of [-16° 0° 16°] yaw and [-8° 0° 8°] pitch angle. The fitting followed the same steps as for the FacePix database: rough light adaption - preliminary translation - pose - 10 shape and texture parameters. The diagrams in Figure 5.9 show the mean absolute error for both the yaw and the pitch angle for the different poses.

In Figure 5.8 it can be observed that for both datasets the absolute yaw angle error does not increase a lot for non-frontal images. This supports the assumption that the

Figure 5.6: Pose estimation for one subject captured with yaw angles from $-16°$ to $+16°$

Figure 5.7: Examples from the FacePix database with bad pose estimates. Errors yaw angle estimate: (a) -13.4°, (b) -13.5°, (c) 14.1°, (d) 12.7°

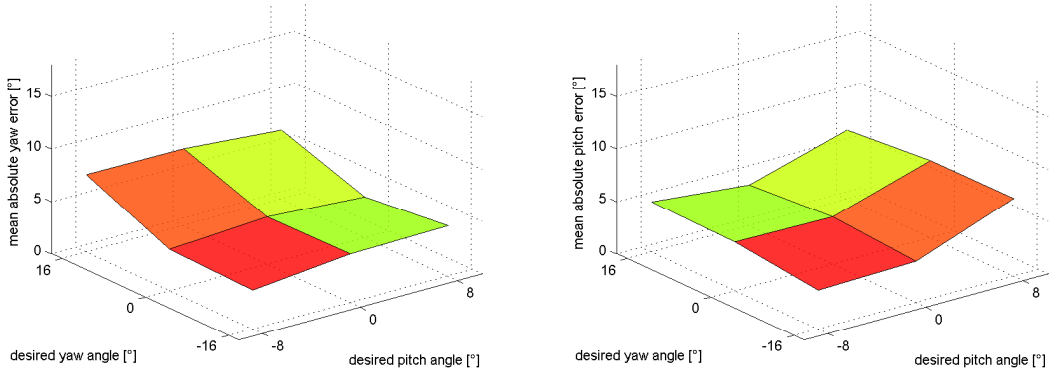


(a) (b)

Figure 5.8: Mean and standard deviation of the absolute yaw angle error for the (a) FacePix and the (b) USF Human ID 3D face database

absolute yaw error					absolute pitch error				
mean	std	median	Q.25	Q.75	mean	std	median	Q.25	Q.75
3.90	3.31	2.81	1.55	5.21	5.14	3.66	4.08	2.11	7.55

Table 5.2: Evaluations for the USF Human ID 3D face database. Mean, standard deviation, median, upper- and lower quartile of the absolute yaw and pitch angular error



(a) (b)

Figure 5.9: Mean absolute error for (a) yaw and (b) pitch angle for the USF Human ID 3D database

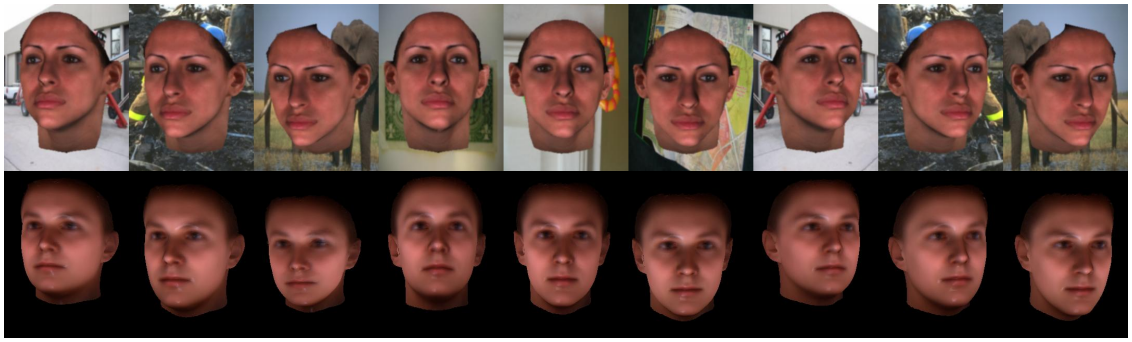


Figure 5.10: Pose estimation for one subject of the USF Human ID 3D face database captured with combination of yaw angles [-16° 0° 16°] and pitch angles [-8° 0° 8°]

quality of pose estimation with 3DMM, unlike some other pose estimation methods, does not decrease for images that deviate from the frontal pose.

For both datasets, a very rough initialization for the location of the face in the image is given. This is realistic as face detection algorithms provide a similar initialization. Only if the face initialization and the target face have enough overlap, the fitting process succeeds to estimate the pose correctly.

Difficulties emerge in combination with illumination, as pose and illumination parameters have a strong interdependence. For the two datasets this is tackled by starting with a rough light adaption that is followed by the determination of the translation parameters. Only then the correct rotation angles are searched. This sequential strategy - first starting

relative yaw error				
mean	std	median	Q.25	Q.75
2.27	2.97	1.27	0.56	2.64

Table 5.3: Evaluations for the FacePix database. Mean, standard deviation, median, upper- and lower quartile of the relative yaw angular error

with only a few parameters and then including more and more - proves to work well both for pose estimation as for shape and texture estimation in the next section.

Another problem becomes obvious within the search for a database with ground truth. The exact pose of the head is hard to determine and depending on the subject. Furthermore there are differences between the person turning her head and the camera being moved. For the pose estimations of the FacePix database, the relative error has also been determined. In the FacePix database, the exact pose change between two images of one subject is known. The absolute difference between this pose change and the pose change of the corresponding fittings is named relative error. Table 5.3, showing the absolute relative error for the yaw angles, exhibits that those errors are much smaller than the absolute errors.

To summarize, pose estimation using a 3D Morphable Model works quite satisfying. One drawback of the method is its relatively slow speed of computation. But in contrast to that, the accuracy of the pose estimate is quite high.

5.3 Shape and texture estimation

The fitting of a 3D Morphable Model to an input image is the main scope of this framework. To test its capabilities, faces of several datasets are used as input to the algorithm.

5.3.1 Databases

Model-generated examples The first test set consists of 250 model-generated images with a resolution of 640x480 pixels. Random shape and texture coefficients are chosen within $\sigma_S = \pm 1$, $\sigma_T = \pm 1$. Using those coefficients, the 3D shape and its texture are calculated. Under a predefined light and fixed pose, two-dimensional image renderings are created from those 3D heads. The advantage of this dataset is that the ground truth (3D shape and texture) of the images is known.

Rendered 3D head scans 250 scans from the original database (see section 3.1) that are not used to build the 3DMM form the basis for another artificial data set. The

full-resolution scans are rendered under predefined pose and illumination to 2D images with a size of 640x480 pixels.

CVL database The CVL database [60] contains 2D images (640x480 pixels) of 114 persons that are mostly of male gender and of younger age. Some subjects wear glasses. Of every person 7 images are taken - full profile, half profile and one frontal image are with neutral expression, while on two further frontal captures the persons are smiling (not used for the experiments). To provide a uniform illumination, the photographs are taken without flash. But additionally to the non-directional ambient also a soft light source illuminates the persons face from the left side of the images.

5.3.2 Error measures

To assess the quality of a fitted 3D model of the face is much more difficult than the quality of a pose estimation. Visual inspection by a human is not very satisfying as it is not easily reproducible and needs a lot of time and resources.

If only one image of the fitted person is available, the residual in the image space (difference of input image and rendered 3D model) gives a hint how well the model fitted the image.

$$e_c(x) = I^{in}(x) - T^{ill}(P^{-1}(x, \alpha, \delta), \alpha, \beta, \gamma) \qquad \forall x \in face \tag{5.3}$$

$$E = \sum_{x \in H} \parallel e_c(x) \parallel^2 = e_c^T(H) e_c(H) \tag{5.4}$$

The disadvantage of this measure is that it can not describe how well the model fitted the underlying 3D shape of the head shown in the input image. Therefore it provides no measure whether over-fitting has occurred or if the entanglement of light and texture has been solved correctly.

So for artificial test examples (model-generated heads or actual 3D scans rendered to an image), the reconstructed 3D model and the ground truth 3D data can be compared directly. A very common measure for this comparison is the Euclidean distance of the respective vertices.

$$E = \parallel x - y \parallel^2 \tag{5.5}$$

For the second dataset (rendered scans from the full database) no vertex-wise cor-

respondence between the fitted model and the original scan exists. Therefore a nearest-neighbor search is performed for every vertex in the fitted model to provide the calculation basis for the Euclidean distance.

The Euclidean distance can be calculated for both of the shape and texture. For shape it is obviously the distance between the corresponding vertices. For the the texture the error measure is the Euclidean distance between the color of the corresponding vertices in the RGB space.

5.3.3 Evaluation

The first evaluation is performed on the model-generated images. The fitting is initialized with known pose and illumination. As the underlying faces are within the model range, it is expected that those images are fitted well and without problems. The fitting of the 3DMM is performed in three steps. First only the 10 primary shape and texture components are calculated, then 50 and finally all 100 modes of the model are determined for the given input image. One fit takes just under 2 minutes of computation time on a system with an Intel Core2 Duo CPU @ 2.6GHz (T9550) and 4GB of main memory on Microsoft Windows Vista x64 SP2.

The details of one exemplary input image (Fig. 5.11(a)) are inspected. Figure 5.12 shows the image space residual over all iterations for the randomly selected triangles. On the course of fitting, new triangles are selected every 50 iterations. The spikes in Fig. 5.12 result from those triangle changes. This shown residual is used by the framework to calculate the derivatives and therefore the parameter updates. The same figure also displays the residual calculated over all image pixels of the rendered head for some specific iteration points during the fitting. Interesting for this dataset is that the model coefficients α and β of the input faces are not recovered in the fitting, although their shape and texture are accurately matched. The final fitted model of the input image can be observed in Fig. 5.11(b).

The next test set consists of images that are artificially rendered from full 3D scans. Thus as in the previous dataset, pose and illumination are known. But in contrast to the previous heads those faces have not been used to build the model, and therefore the generalization capabilities of the model can be judged. The number of modes fitted starts with 10, and is, after every 200 iterations, increased to 50 and then the full 100 components respectively. The fitting takes about the same time per image as it does for the previous dataset (just under 2 minutes). The yaw angle of the input images is 20°, and the pitch

(a) (b)

Figure 5.11: (a) One model-generated example from the first dataset and (b) the head fitted to the input image (a)

rotation is 10°.

One specific example of this second database is shown in Figure 5.14(a). Both the residual that is used for the optimization (only calculated at randomly selected triangles) and the residual calculated over all pixels of the fitted region are compared in Figure 5.15. The fitted head model that is the result of the process after the 600 iterations can be observed in Figure 5.14(b).

Figure 5.13 shows the Euclidean distance for the fittings of both datasets. Interestingly, the median error for the shape is slightly lower for the real head scans, whilst for the texture the median Euclidean distance is significantly smaller for the model-generated examples. This can be explained by observing that 100 shape coefficients cover more of the total variance than the 100 texture components do (97.8% versus 87.29%).

Real-world images from the CVL database are fitted in the third row of experiments. Here the three-dimensional shapes of the faces are not known, and the only error-measure

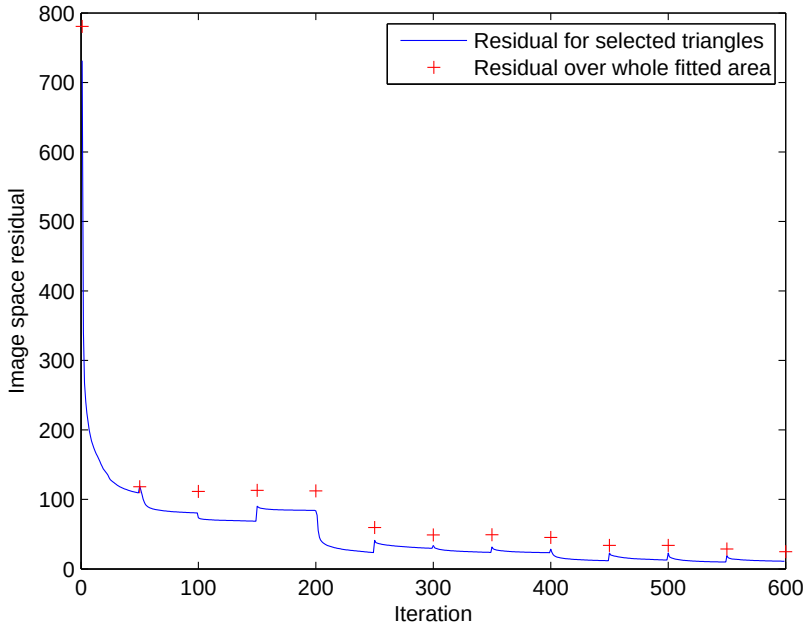
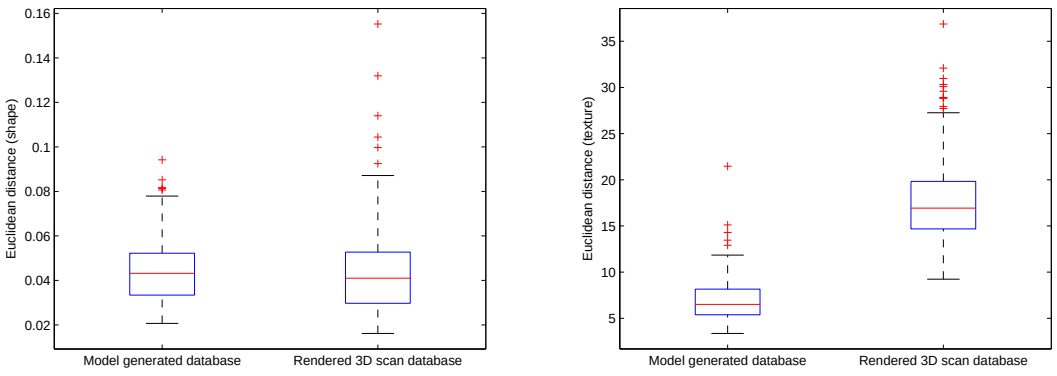
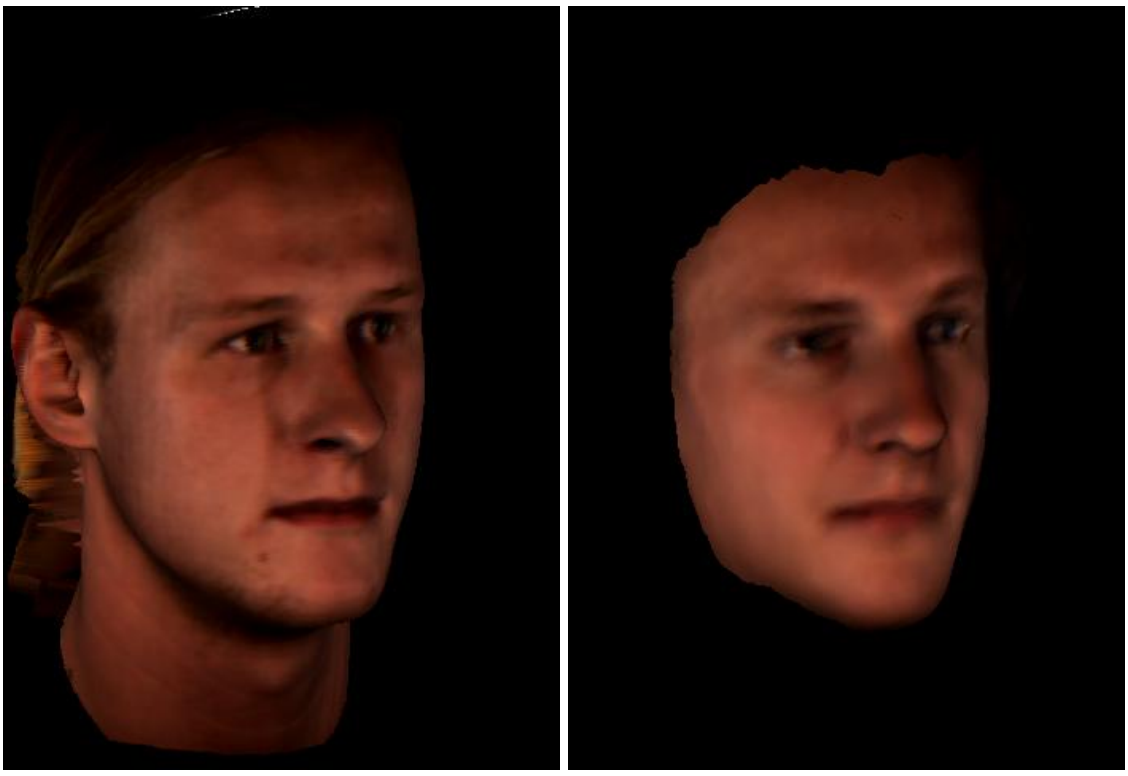


Figure 5.12: Image space residuals calculated over the triangles selected at the respective iterations and the full image space residual for the input image in Figure 5.11(a)



(a) (b)

Figure 5.13: Euclidean distance between the ground truth and the fitted result for (a) shape and (b) texture



(a) (b)

Figure 5.14: (a) Rendering of a head scan from the second dataset and (b) the head fitted to the input image (a)

that can be evaluated are the image-space residuals. For this database, only a very rough initial pose is given. The initial illumination has been determined once using a few exemplary images from the database. As for the previous experiments, the number of modes is increased after every 200 iterations from 10 to 50 to 100.

The boxplot of the image space residual for the five rough poses in the CVL database (Fig. 5.16) shows that the error for images with negative yaw angle is slightly higher than for the frontal and right looking poses. In theory the error should be the same for similar absolute yaw angles. But the illumination source in this dataset shines from the left side of the image. Therefore this asymmetry of the residuals can have two roots, both connected with the light. One can be the illumination initialization that is calculated only once for the whole dataset, and by coincidence is more accurate for images with increased yaw angle. The second source of error can be that the illumination model (see section 4.1.2) is not able to exactly represent the real light source used in the CVL database.

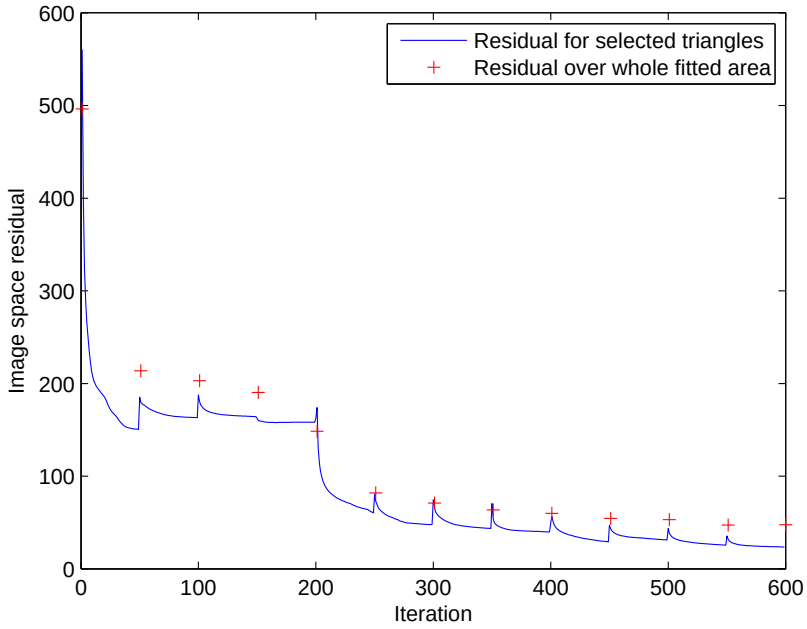


Figure 5.15: Image space residuals calculated over the triangles selected at the respective iterations and the full image space residual for the input image in Figure 5.14(a)

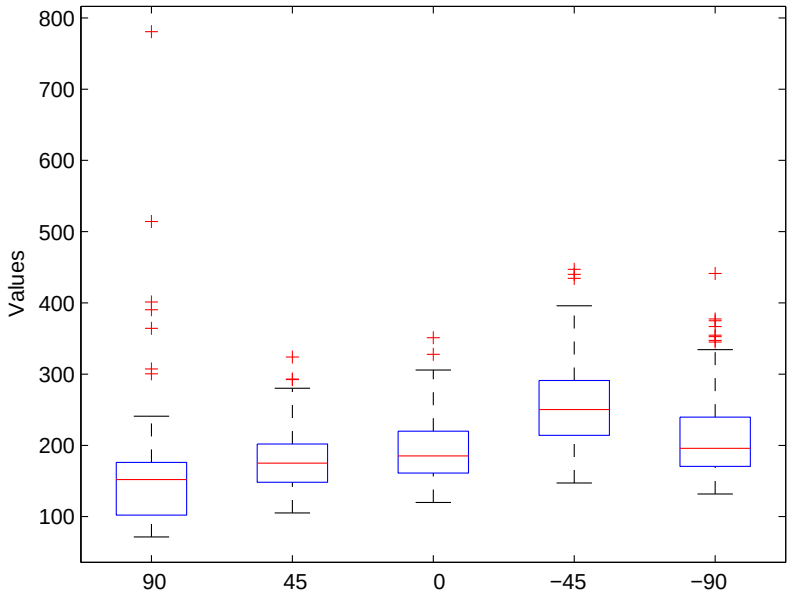


Figure 5.16: Image space residuals for the CVL database for different yaw angles

This asymmetry of the image space residual also shows the close entanglement of illumination, pose, shape and texture during the generation of an image, which is hard to separate during the fitting process. If one parameter is initialized substantially wrong, it is not possible to estimate the other parameters correctly.

As for the two foregoing examples, one specific input example is shown in Figure 5.17. The image space residual (selected triangles and all pixels) can be observed in Figure 5.18, and the resulting face has been rendered under the same pose in Figure 5.17(b).

(a) (b)

Figure 5.17: (a) One photo of the CVL database and (b) the face fitted to it

Apart from the initial illumination, it is also necessary to adjust the step size parameter λ for the gradient descent optimization. This parameter is fixed at a value where the descent is fast enough, but where still no oscillation occurs. Figure 5.19 shows the search for an optimal λ during the fitting of only the first shape parameter. In the first test (Fig. 5.19(a)), it is too low, the last one (Fig. 5.19(c)) clearly is too high, and the middle graph (Fig. 5.19(b)) shows a satisfying result.

Figure 5.20 shows some examples where the framework failed to fit a face with sufficient quality. For those images with a high error measure, it can be observed that most of them have hair occluding a larger part of the fitted face area. As the modeled residual is not robust with respect to outliers, those areas misguide the fitting resulting in a badly deformed model.

On the other hand, if the occlusions do not cover too many pixels of the input image, the holistic approach manages to recover a well-fitted model and, as a side effect, eliminates the occlusion. Figure 5.21(a) shows a subject wearing glasses. The result after the fitting

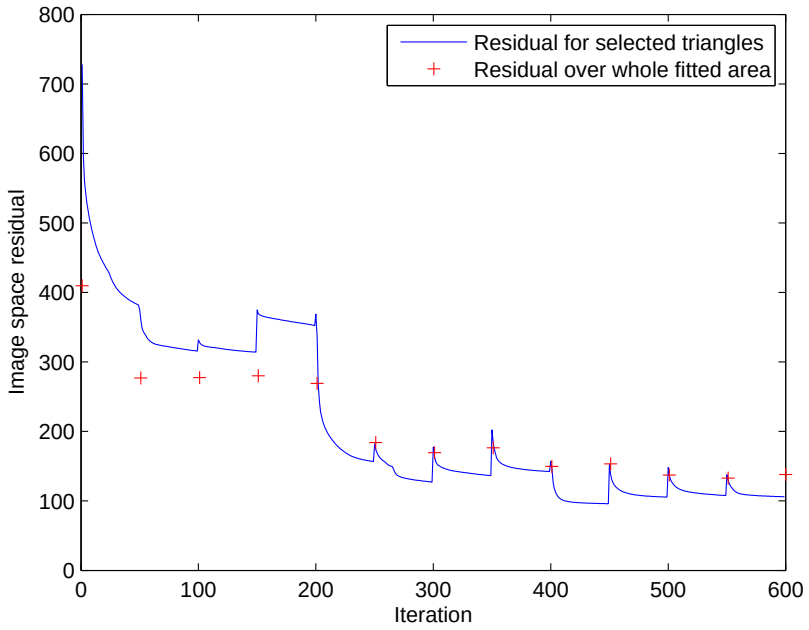
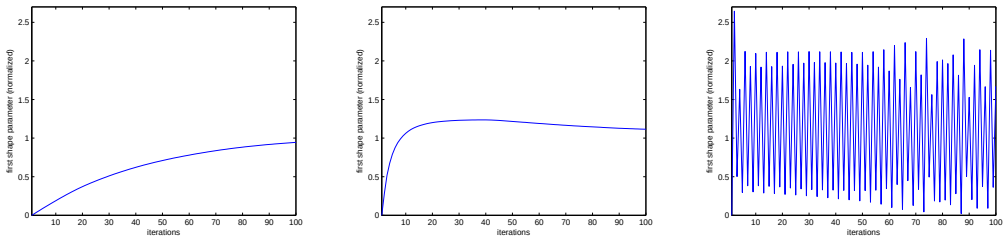


Figure 5.18: Image space residuals calculated over the triangles selected at the respective iterations and the full image space residual for the input image in Figure 5.17(a)



(a) (b) (c)

Figure 5.19: Adjustment of the parameter λ for the gradient descent: (a) value too low $\rightarrow$ slow convergence (b) well-chosen λ (c) value too high $\rightarrow$ oscillation occurs

is shown in Figure 5.21(b) - the "occluding" glasses are eliminated.

As several persons in the head scan database (section 3.1) that forms the basis of our 3DMM wear a beard, the framework also succeeds to model facial hair. Of course the successful beard fitting depends on the random triangle selection. If the beard-covered area is very small, there is only a minor probability that the elected triangles lie in there and the fitting process considers it. So for example in Figure 5.22(a) the person wears a

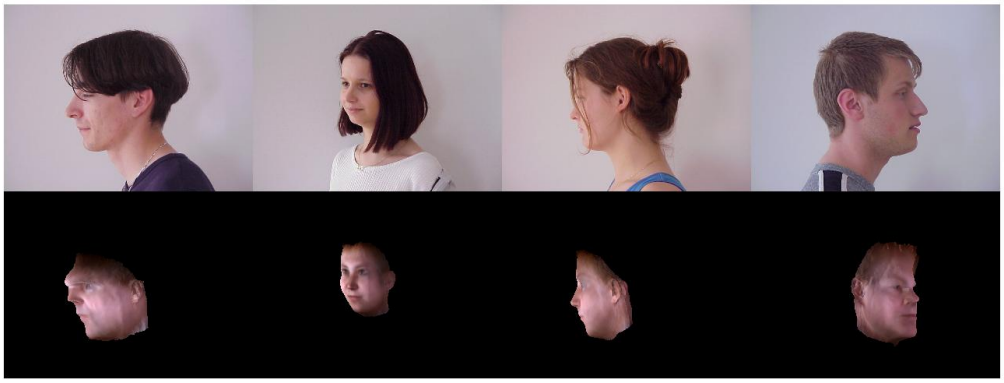


Figure 5.20: Input images from the CVL database (upper row) where the fitting failed (lower row)

(a) (b)

Figure 5.21: (a) One subject of the CVL database wearing glasses and (b) the fitted face with automatically removed glasses

very narrow beard on the chin. This beard is not captured during the adoption of the model to the input image and the final rendering shows the same person without facial hair (Figure 5.22(b)). The same happens with other small details such as birthmarks (see Figure 5.23(a) and the rendered fitting in Figure 5.23(b)). For the framework, those cases are the same as the minor occlusions in the foregoing example of the glasses. But if the area covered by facial hair is large enough, the beard is well reconstructed. Some successful examples (mustache, partial and full beard) are shown in Figure 5.24.

Another interesting detail is the mouth of the subjects. Only very few training samples

(a) (b)

Figure 5.22: (a) One subject of the rendered 3D scans database with a narrow "soul patch" beard and (b) the fitted face without facial hair

(a) (b)

Figure 5.23: (a) One subject of the rendered 3D scans database with a birthmark and (b) the fitted face with the mark removed

have an opened mouth, and therefore, the fitted 3D head has a closed mouth even if a person poses with open mouth (see Figure 5.25(a) and Figure 5.25(c)). On the other hand, many happily smiling subjects are in the head scan database and are used for the generation of the 3DMM. So the smiling expression in a subjects face (Fig. 5.25(b)) is successfully reproduced in its three-dimensional model (Fig. 5.25(d)).

The final Figure 5.26 compares the evaluation of all three datasets. Of course the examples from the first dataset are best reconstructed, as they are generated by the model. The reconstruction error from the second dataset is nearly as low as the reconstruction error from the first one. This result shows that the model can generalize well for unseen faces. Generalization capabilities with respect to new faces under different recording con-

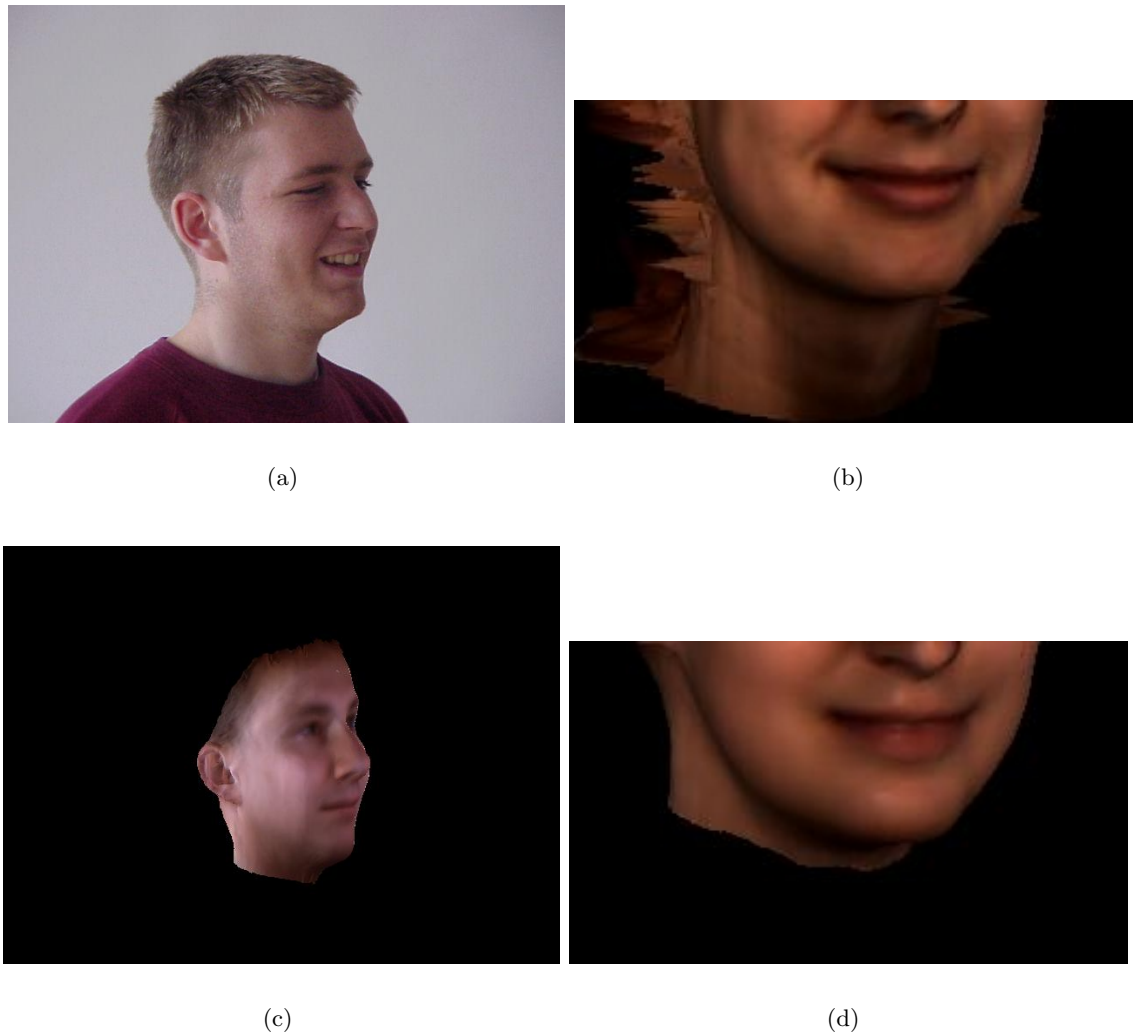
(a) (b) (c)

(d) (e) (f)

Figure 5.24: Three examples from the rendered 3D scan database (upper row) where the facial hair is successfully fitted (lower row)

ditions are shown only by the third dataset - real input images from the CVL dataset. Here the image space residuals are not as low, because illumination and pose errors contribute a proportional big share to the residual.

From the evaluation of shape and texture estimation using our 3D Morphable Model, it can be seen that the framework is able to successfully model unseen faces. Also the adoption to varying pose and illumination conditions works well. Interestingly, for the first dataset (model-generated images), the original coefficients could not be recovered. Difficulties arise for the fitting process because of the close entanglement of illumination, pose, shape and texture. If the adoption starts with either a significantly wrong pose or light, the search for the parameters is more likely to be trapped in a wrong minimum. But on the other hand, the framework has surprising properties with respect to some details in the human face. Small or fine structures (birthmarks, glasses, ...) are eliminated in the fitted model, while larger specialties such as a beard or a smile that are present in the training database are recreated.



(a) (b)

(c) (d)

Figure 5.25: Subjects smiling / with an open mouth and their fittings (lower row)

5.4 Implementation

Most of the framework for building and fitting a 3D Morphable Model has been implemented in Matlab. The first part of the framework covers the construction of the 3DMM from the raw laser scans. The second part is concerned with fitting the model to input images.

Building the model begins with importing the raw laser scans, stored in the echo file format, into Matlab by the I/O-function readecho. One scan is chosen as template face for the correspondence establishment. The shape of this face is cropped in the 3D modeling

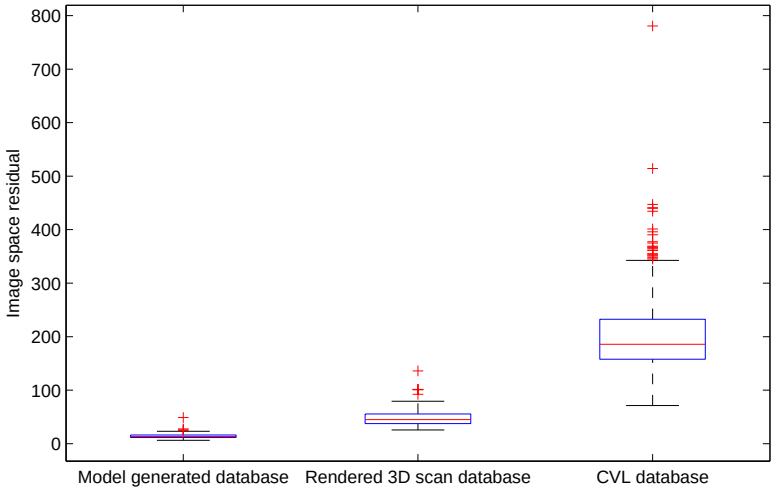


Figure 5.26: Box plot of image space residuals for fitted models of all three databases.

software "Blender"* (import/export through the the 3D file format OBJ† using readobj and writeobj). The following mesh simplification of the template face is done by "qslim". The data exchange with this program happens through the SMF file format‡, for which the I/O-functions readsmf and writesmf were implemented.

The rough pose estimation for the laser scans is done by an external face detection program (see section 3.4.1). It provides the base ground for a finer alignment that is done by the ICP algorithm (section 3.4.1.1). Crucial to this algorithm is the nearest neighbor search. Unfortunately this search can not be easily optimized in Matlab. Therefore a C++ implementation of a kd-tree provided at the Matlab central file exchange§ is used in Matlab through the mex interface. This interface is also used to speed up some routines such as multi-dimensional matrix multiplication (e.g. equation 4.27) in C.

The OSNRICP algorithm (section 3.4.4) for establishing dense 3D correspondence between the template face and a rigidly aligned laser scan uses huge matrices. They are implemented as sparse matrices, as it would not be feasible otherwise to carry out computations on them.

*Blender `http://www.blender.org/`. Accessed on January 28, 2010

†OBJ File Format `http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/`. Accessed on January 28, 2010

‡SMF file format. `http://people.sc.fsu.edu/~burkardt/data/smf/smf.txt`. Accessed on January 28, 2010

§KDTree Nearest Neighbor and Range Search - MATLAB Central. `http://www.mathworks.com/matlabcentral/fileexchange/7030-kd-tree-nearest-neighbor-and-range-search`. Accessed on January 28, 2010

convert head scan to JPEG	0.8s
face detection	1.4s
rough pose detection	0.2s
ICP	8s
OSNRICP	83s
PCA	0.1s

Table 5.4: Computation times per face

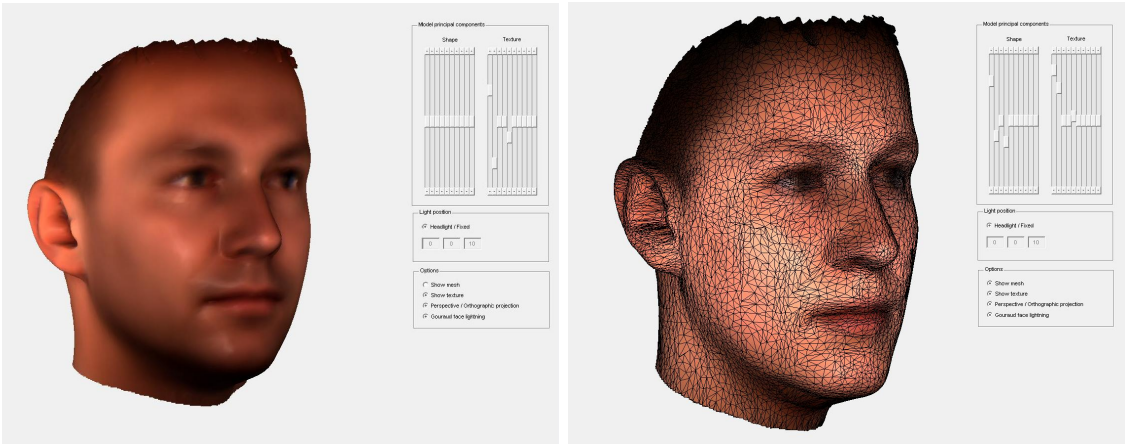
The final step in building the model is a Principal Component Analysis on the registered data. The Matlab internal function princomp is used with the 'econ' switch to calculate the important dimensions in the face space.

At this step, the "Morphable Model Explorer" (Figure 5.27) can visualize a 3DMM up to a certain degree and allows the user to inspect and judge the model. The rendered face can be adjusted through 10 slider controls for the shape components and another 10 controls for the texture components. Interactively the face is adapted as the sliders are moved. For close inspection the face can be rotated, dragged and zoomed. In the implementation of the Explorer, the Matlab-integrated routine "patch" has been used extensively. It uses the OpenGL capabilities of the graphics card and therefore provides fast rendering of the calculated face.

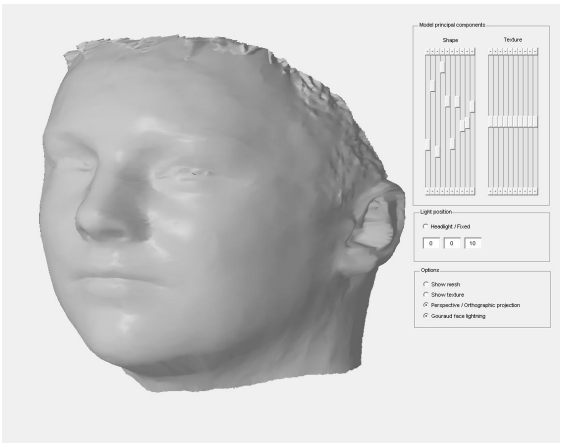
The second main part of the framework consists of fitting a 3DMM to an input image. The basis for the parameter updates of the model is provided by the derivatives of the residual function. They are calculated by different functions ("cost*") that only compute the relevant parts (w.r.t. shape, texture, pose etc.) of the derivatives. The optimization routine utilizes a standardized interface so that in various stages of fitting the derivative residuals can be easily exchanged.

As the residual is only calculated on the visible triangles of the face, the set of non-occluded areas is determined by the "visibleTriangles" function. Its decision on which triangles are not occluded from the viewer rests upon the preliminary parameters that have been adjusted so far. From this set, the function "selectTriangles" elects the triangles that are actually used in the randomized optimization. The probability that a triangle is used in the fitting is on one hand proportional to the area it occupies in the rendered image. On the other hand the probability increases for important regions such as nose, eyes and mouth or the silhouette.

The parameters of the model are structured in four objects. "alpha" contains the shape coefficients α, while "beta" stores the texture components β of the current face. The illumination parameters are grouped in the structure *ill*. "specular_material" sets the



(a) (b)



(c)

Figure 5.27: Morphable Model Explorer

Matlab patch property SpecularStrength, which is $L^d k_s$ from the formula 4.14 (intensity of the specular component). The exponent ν that defines the size of the specular reflection is stored in "shininess" (Matlab patch property SpecularExponent). Finally the pose and projection parameters are aggregated within *rho*.

color_contrast	1	c	(formula 4.20)
color_gain	[1 1 1]	$[g_r\ g_g\ g_b]$	(formula 4.20)
color_offset	[0 0 0]	o	(formula 4.19)
ambient_light	[0.6 0.6 0.6]	L^a	(formula 4.10)
directional_light	[1 1 1]	L^d	(formula 4.11)
light_direction	[0 0 -1]	d	(formula 4.11)
specular_material	0.1	$L^d k_s$	(formula 4.14)
shininess	70	ν	(formula 4.14)
view_direction	[0 0 -1]	v	(formula 4.14)

Table 5.5: Variables in the illumination structure *ill*

rx	0	ϕ_x	(formula 4.2)
ry	0	ϕ_y	(formula 4.3)
rz	0	ϕ_z	(formula 4.4)
s	0	s	(formula 4.1)
tx	0	t_x	(formula 4.5)
ty	0	t_y	(formula 4.5)
tz	-20	t_z	(formula 4.5)
viewport_x	640	input image size	(formula 4.16)
viewport_y	480	input image size	(formula 4.16)
fovy	10	field of view y	(formula 4.15)
zNear	30	first clipping plane	(formula 4.15)
zFar	2	second clipping plane	(formula 4.15)

Table 5.6: Variables in the pose structure *rho*

Chapter 6

Conclusion and Outlook

Contents

6.1 Summary and Contributions

In this master's thesis a framework for building and fitting a 3D Morphable Model has been developed. In section 2.1 first some views on the historical development of statistical shape and appearance models is given, and then we present some details on the popular Active Appearance Models. Section 2.3 introduces the concept of 3D Morphable Models, and then an overview over the many areas where they can be employed is given.

From a large database of raw three-dimensional head scans we construct a full 3D Morphable Face Model. The full correspondence between the scans is obtained by adapting the Optimal-Step Non-Rigid ICP algorithm. Using all the registered scans, we explain the idea of the Face Space. The following Principal Component Analysis removes redundancy from the data and orders the dimensions with respect to their importance. Some alternatives for the determination of the number of Principal Components are shown.

In the fitting chapter 4 the steps of adapting a 3D Morphable Model so it resembles a given input image as close as possible are shown. We first explain in detail the rendering of a three-dimensional head to a two-dimensional image, and then establish a residual measure using a slight variation of this rendering process. The derivatives of this residual measure (section 4.3) are used as a basis for the optimization process that is at the core

of the Morphable Model Fitting. For a visually pleasing result, an illumination-corrected texture from the input image is merged with the fitted model.

An experimental evaluation is done in the last major chapter 5. The generated 3D Morphable Model is inspected closely, and the few cases where registration failed are identified through the reconstruction error. Then the 3DMM is evaluated in two types of experiments. First the framework is used to determine the pose of human heads in two-dimensional photographs. The results (section 5.2.3) show high accuracy for those experiments. Second, the frameworks performance for fitting input images under controlled and uncontrolled conditions is inspected. Three datasets are used for this performance evaluation: model-generated images, rendered 3D head scans and the CVL database. Finally some details of the implementation are given.

6.2 Conclusions

For the use of 3D Morphable Models in various areas we see both advantages and disadvantages: The effort for the construction of a 3DMM is quite high, although it has to be done only once. 3D scans of many examples of the underlying category need to be captured. Establishing the dense three-dimensional correspondence between the raw scans is timely and computationally consuming. The building of a model has to be done for every new category - the learned knowledge can not be transfered. But the big reward is a model with which a complete category can be represented in an efficient and very elegant way.

Also the fitting process shows some difficulties, but has also many positive sides. There are a number of constants that have to be determined manually for a type of input images (focal length, specular coefficient, etc.). The optimization of a parameter space with such a high dimensionality is tedious and it is not easy to find a globally optimal solution. This problem is further complicated, as several parameters such as pose, light and shape have a high inter-dependence. But on the other side, the 3D model gives us the possibility, in contrast to other statistic deformation models, to explicitly describe and model illumination, pose and shape. The stochastic approach to the optimization introduces robustness and as a nice side effect also speeds up the computation.

6.3 Directions for Future Work

This framework can be enhanced and developed further in several directions. The possible improvements can roughly be split belonging to the model building and to the fitting

process.

Depending on the field of application (e.g. identification, but not pose estimation), it may be useful to support the foundation of the model with more samples. Those can be faces with expressions (smiling, anger, ...), samples with greater diversity in race, and also just additional faces from the same spectrum. Only a few studies (e.g. Meytlis [54]) have been conducted on how many dimensions are needed in the face space. This consideration can be done using mathematical data (minimum distance between two faces, LEV diagram, ...) or on comparing it with the human visual system (which faces can be distinguished by a human observer). As high-resolution photographs become more and more common, it is also useful to have a model at hand that represents the underlying category with an equally high detail. In this work we observed that the computational load for the Optimal Step Non-Rigid ICP algorithm increases non-proportionally with a higher number of vertices. So a possible direction could be the combination of Optical Flow with this algorithm. If it can be assured, that the model fitted a new input image successfully, this input image gives probably also some new information that can be used to update the model. This would not be very difficult for the texture, but could be more involved for the shape direction.

The fitting of a 3D Morphable Model to an input image also provides room for improvement. Several examples (see section 5.3.3) showed that occlusions like hair can distort the optimization towards a correct solution. Instead of using the sum of squared differences for the error measure, a more robust cost function (Huber norm, Lorentzian norm, ...) that tolerates outliers better could improve the quality of a parameter estimate significantly. But also an algorithm that excludes detected outliers from the ongoing fitting process is promising better results. Apart from the quality of the parameter estimation, the speed of the optimization could preferably be faster. The adoption of the Inverse Compositional Image Alignment by Baker [5] to 3D Morphable Models (Rhomdani and Vetter, [69]) that is quite fast could be expanded to include strong perspective and a more realistic illumination model. In our framework, the calculation of the Jacobian consumes most of the time, so another direction of improvement lies in the precalculation or approximation of the derivatives of the error measure. Another option is to reduce the high dimensionality of the face space. The illumination model restricts accurate fitting to photographs taken with only one main light source. The quality of the shape estimate could probably be increased by including more lights and expanding the calculations for a more sophisticated illumination model than the Blinn-Phong. Finally more features in addition to the simple

pixel color might both speed up the calculation and add accuracy to the fitting process. For example strong image gradients - edges - could provide a good guidance for the shape estimation.

Appendix A

Derivatives

Contents

A.1 Derivatives with respect to shape coefficients

The derivative of the error measure 4.24 with respect to the shape coefficients α is the following

$$\frac{\partial E}{\partial \alpha} = \frac{1}{2}\frac{\partial (e_c^T e_c)}{\partial \alpha} = \frac{1}{2}(\frac{\partial e_c^T}{\partial \alpha} e_c + e_c^T \frac{\partial e_c}{\partial \alpha}) = \frac{\partial e_c^T}{\partial \alpha} e_c \tag{A.1}$$

For the derivative of the cost function, it is necessary to use the chain rule

$$\frac{\partial e_c}{\partial \alpha} = \frac{\partial I^{in}(P(U,\alpha,\delta))}{\partial \alpha} - \frac{\partial T^{illc}(U,\alpha,\beta,\gamma)}{\partial \alpha} \tag{A.2}$$

$$\frac{\partial I^{in}(P(U,\alpha,\delta))}{\partial \alpha} = \frac{\partial I(\bar{x},\bar{y})}{\partial (\bar{x},\bar{y})}|_{P(U,\alpha,\delta)} \frac{\partial P(U,\alpha,\delta)}{\partial (U)}|_{S(\alpha)} \frac{\partial S(\alpha)}{\partial \alpha} \tag{A.3}$$

The first part of the derivation are the image gradients in both $\bar{x}$ and $\bar{y}$ direction of the input image, evaluated at the locations of the triangle centers of the head image coordinates rendered with the current parameter estimates α and δ. The next part derived using the chain rule is the derivative of the transformation P (equation 4.18) with respect

to the triangle centers in 3D. The last one is the derivative of the shape derived with respect to the shape parameters α.

$$\frac{\partial I(\bar{x},\bar{y})}{\partial(\bar{x},\bar{y})} = \begin{bmatrix} \frac{\partial I^R}{\partial \bar{x}} & \frac{\partial I^R}{\partial \bar{y}} \\ \frac{\partial I^G}{\partial \bar{x}} & \frac{\partial I^G}{\partial \bar{y}} \\ \frac{\partial I^B}{\partial \bar{x}} & \frac{\partial I^B}{\partial \bar{y}} \end{bmatrix} \tag{A.4}$$

$$\frac{\partial P(x,y,z)}{\partial(x,y,z)} = \begin{bmatrix} \frac{\partial P^{\bar{x}}}{\partial x} & \frac{\partial P^{\bar{x}}}{\partial y} & \frac{\partial P^{\bar{x}}}{\partial z} \\ \frac{\partial P^{\bar{y}}}{\partial x} & \frac{\partial P^{\bar{y}}}{\partial y} & \frac{\partial P^{\bar{y}}}{\partial z} \end{bmatrix} \tag{A.5}$$

$$\frac{\partial S(\alpha)}{\partial \alpha_i} = S_i \tag{A.6}$$

In the derivative of the illuminated texture the ambient term falls out completely, as it only depends on the parameter β. It is mainly based on the derivative of the normal vectors.

$$\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \alpha} = M\left(L^d \frac{\partial < n,d >}{\partial \alpha} T + L^d k_s \nu < h,n >^{(\nu-1)} < h, \frac{\partial n}{\partial \alpha} >\right) \tag{A.7}$$

$$\frac{\partial < n,d >}{\partial \alpha} = \frac{\partial n}{\partial \alpha} d \tag{A.8}$$

To derive the triangles normal vector the two vectors vec_1 and vec_2 from the definition of the normal product are used. Those two vectors point from one vertex v_1 of the triangle to the other two corners v_2 and v_3. The final normal n result from the transformation by the modelview transformation (equation 4.7 of the original normal $\tilde{n}4$.

$$n = MV\tilde{n} \tag{A.9}$$

$$\frac{\partial \tilde{n}}{\partial \alpha} = \frac{\frac{vec_1 \times vec_2}{\| vec_1 \times vec_2 \|}}{\partial \alpha} = \frac{\frac{vec_1 \times vec_2}{\partial \alpha} \parallel vec_1 \times vec_2 \parallel - vec_1 \times vec_2 \frac{\| vec_1 \times vec_2 \|}{\partial \alpha}}{\parallel vec_1 \times vec_2 \parallel^2} \tag{A.10}$$

$$\begin{aligned}\frac{\partial \parallel vec_1 \times vec_2 \parallel}{\partial \alpha} =& \frac{\partial \left((vec_1 \times vec_2)^T (vec_1 \times vec_2)\right)^{1/2}}{\partial \alpha} \\ =& \frac{\left(\frac{\partial (vec_1 \times vec_2)^T}{\partial \alpha}(vec_1 \times vec_2) + (vec_1 \times vec_2)^T \frac{\partial (vec_1 \times vec_2)}{\partial \alpha}\right)}{2\left((vec_1 \times vec_2)^T (vec_1 \times vec_2)\right)^{1/2}}\end{aligned} \tag{A.11}$$

The derivative of the cross-product is as follows

$$\frac{\partial (vec_1 \times vec_2)}{\partial \alpha} = \frac{\partial vec_1}{\partial \alpha} \times vec_2 + vec_1 \times \frac{\partial vec_2}{\partial \alpha} \tag{A.12}$$

$$vec_1 = v_2 - v_1 \tag{A.13}$$

$$vec_2 = v_3 - v_1 \tag{A.14}$$

$$\frac{\partial vec_1}{\partial \alpha} = \frac{\partial v_2}{\partial \alpha} - \frac{\partial v_1}{\partial \alpha} \tag{A.15}$$

$$v^i = \bar{S}^i + \sum_{k=0}^{m} \alpha_k S_k^i \tag{A.16}$$

$$\frac{\partial v^i}{\partial \alpha_j} = S_j^i \tag{A.17}$$

All the homogeneous coordinates of the vertices have to be normalized after the multiplication with all the transformation matrices united in P. The derivatives of this normalization are

$$\frac{\partial P^{\bar{x}}}{\partial x} = \frac{P_{\bar{x},1}}{N} - \frac{N_1}{N^2} P_{\bar{x}} \tag{A.18}$$

$$\frac{\partial P^{\bar{x}}}{\partial y} = \frac{P_{\bar{x},2}}{N} - \frac{N_2}{N^2} P_{\bar{x}} \tag{A.19}$$

$$\frac{\partial P^{\bar{x}}}{\partial z} = \frac{P_{\bar{x},3}}{N} - \frac{N_3}{N^2} P_{\bar{x}} \tag{A.20}$$

$$\frac{\partial P^{\bar{y}}}{\partial x} = \frac{P_{\bar{y},1}}{N} - \frac{N_1}{N^2} P_{\bar{y}} \tag{A.21}$$

$$\frac{\partial P^{\bar{y}}}{\partial y} = \frac{P_{\bar{y},2}}{N} - \frac{N_2}{N^2} P_{\bar{y}} \tag{A.22}$$

$$\frac{\partial P^{\bar{y}}}{\partial z} = \frac{P_{\bar{y},3}}{N} - \frac{N_3}{N^2} P_{\bar{y}} \tag{A.23}$$

A.2 Derivatives with respect to texture coefficients

The only terms that vary with the texture parameters are the ambient and the diffuse illumination. So the simple derivation of the cost function with respect to the parameters β is

$$\frac{\partial e_c}{\partial \beta} = -\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \beta} \tag{A.24}$$

$$\frac{\partial T^{illc}(u,\alpha,\beta,\gamma)}{\partial \beta} = M\left(L^a \frac{\partial T}{\partial \beta} + L^d < n,d > \frac{\partial T}{\partial \beta}\right) \tag{A.25}$$

The derivative of the texture with respect to the j-th texture parameter is the j-th Principal Component.

$$\frac{\partial T}{\partial \beta_j} = T_j \tag{A.26}$$

A.3 Derivatives with respect to pose parameters

Both the $I^{in}(P(u,\alpha,\delta))$ and the $T^{illc}(u,\alpha,\beta,\gamma)$ part of the residual function depend on the pose parameters ρ. For the first part it is obvious that it depends on the pose parameters, but also the illuminated texture varies with changes in pose. If the light direction is given in world coordinates, the diffuse reflection term is influenced by a pose change, as the angle between the normal vectors and the light direction changes. The specular term depends also on the viewers direction with respect to the face - also this relation changes as the pose is altered. Here the derivatives with respect to the rotation angles are shown. For the other pose parameters, they are similar.

$$\frac{\partial e_c}{\partial \rho} = \frac{\partial I(\bar{x},\bar{y})}{\partial(\bar{x},\bar{y})}|_{P(\rho,S(\alpha))} \frac{\partial P(\rho,x,y,z)}{\partial \rho}|_{S(\alpha)} - \frac{\partial T^{illc}}{\partial \rho} \tag{A.27}$$

$$\frac{\partial P(\rho,x,y,z)}{\partial \rho} = \begin{bmatrix} \frac{\partial P^{\bar{x}}}{\partial \rho_x} & \frac{\partial P^{\bar{x}}}{\partial \rho_y} & \frac{\partial P^{\bar{x}}}{\partial \rho_z} \\ \frac{\partial P^{\bar{y}}}{\partial \rho_x} & \frac{\partial P^{\bar{y}}}{\partial \rho_y} & \frac{\partial P^{\bar{y}}}{\partial \rho_z} \end{bmatrix} \tag{A.28}$$

$$\frac{\partial T^{illc}}{\partial \rho} = M\left(L^d < \frac{\partial n}{\partial \rho}, d > T + L^d k_s \nu < h,n >^{(\nu-1)} < h, \frac{\partial n}{\partial \rho} >\right) \tag{A.29}$$

In the modelview matrix MV (equation 4.7) only the rotation matrix R depends on the rotation angles. The normal is transformed by this modelview matrix.

$$\frac{\partial n}{\partial \rho} = T \cdot \frac{\partial R}{\partial \rho} \cdot SC \cdot \tilde{n} \tag{A.30}$$

$$\frac{\partial R}{\partial \rho_x} = \begin{bmatrix} 0 & \cos(\rho_x)\sin(\rho_y)\cos(\rho_z)+\sin(\rho_x)\sin(\rho_z) & -\sin(\rho_x)\sin(\rho_y)\cos(\rho_z)+\cos(\rho_x)\sin(\rho_z) & 0 \\ 0 & \cos(\rho_x)\sin(\rho_y)\sin(\rho_z)-\sin(\rho_x)\cos(\rho_z) & -\sin(\rho_x)\sin(\rho_y)\sin(\rho_z)-\cos(\rho_x)\cos(\rho_z) & 0 \\ 0 & \cos(\rho_x)\cos(\rho_y) & -\sin(\rho_x)\cos(\rho_y) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.31}$$

$$\frac{\partial R}{\partial \rho_y} = \begin{bmatrix} -\sin(\rho_y)\cos(\rho_z) & \sin(\rho_x)\cos(\rho_y)\cos(\rho_z) & \cos(\rho_x)\cos(\rho_y)\cos(\rho_z) & 0 \\ -\sin(\rho_y)\sin(\rho_z) & \sin(\rho_x)\cos(\rho_y)\sin(\rho_z) & \cos(\rho_x)\cos(\rho_y)\sin(\rho_z) & 0 \\ -\cos(\rho_y) & -\sin(\rho_x)\sin(\rho_y) & -\cos(\rho_x)\sin(\rho_y) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.32}$$

$$\frac{\partial R}{\partial \rho_z} = \begin{bmatrix} -\cos(\rho_y)\sin(\rho_z) & -\sin(\rho_x)\sin(\rho_y)\sin(\rho_z)-\cos(\rho_x)\cos(\rho_z) & -\cos(\rho_x)\sin(\rho_y)\sin(\rho_z)+\sin(\rho_x)\cos(\rho_z) & 0 \\ \cos(\rho_y)\cos(\rho_z) & \sin(\rho_x)\sin(\rho_y)\cos(\rho_z)-\cos(\rho_x)\sin(\rho_z) & \cos(\rho_x)\sin(\rho_y)\cos(\rho_z)+\sin(\rho_x)\sin(\rho_z) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.33}$$

A.4 Derivatives with respect to illumination parameters

Just the illuminated corrected texture term T^{illc} has to be concerned for changing lighting conditions. Here is its derivation with respect to the direction of light d. The derivatives for the other illumination parameters are similar.

$$\frac{\partial T^{illc}}{\partial d} = M\left(L^d \frac{\partial < n, d >}{\partial d} T + L^d k_s \nu < \frac{(d+v)}{\parallel d+v \parallel}, n >^{(\nu-1)} < \frac{\frac{(d+v)}{\|d+v\|}}{\partial d}, n > 1_{3\times 1}\right) \tag{A.34}$$

$$\frac{\frac{(d+v)}{\|d+v\|}}{\partial d} = \frac{d \parallel d+v \parallel -(d+v)\frac{\partial \|d+v\|}{\partial d}}{(\parallel d+v \parallel)^2} \tag{A.35}$$

$$\begin{aligned} \frac{\partial \parallel d+v \parallel}{\partial d} = \frac{1}{2}\left[(d+v)^T(d+v)\right]^{-\frac{1}{2}} \left[\frac{\partial (d+v)^T}{\partial d}(d+v) + (d+v)^T \frac{\partial (d+v)}{\partial d}\right] = \\ \frac{1}{2}\left[(d+v)^T(d+v)\right]^{-\frac{1}{2}} \left[\frac{\partial d^T}{\partial d}(d+v) + (d+v)^T \frac{\partial d}{\partial d}\right] \end{aligned} \tag{A.36}$$

Bibliography

[1] Abate, A., Nappi, M., Riccio, D., and Sabatino, G. (2007). 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906.

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

[3] Amberg, B., Romdhani, S., and Vetter, T. (2007). Optimal step nonrigid ICP algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8.

[4] Ansari, A. and Abdel-Mottaleb, M. (2003). 3D face modeling using two views and a generic face model with application to 3D face recognition. In *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, pages 37–44.

[5] Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1.

[6] Barnard, S. T. and Fischler, M. A. (1982). Computational stereo. *ACM Computing Surveys (CSUR)*, 14(4):572.

[7] Beichel, R., Bischof, H., Leberl, F., and Sonka, M. (2005). Robust active appearance models and their application to medical image analysis. *IEEE Transactions on Medical Imaging*, 24(9):1151–1169.

[8] Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720.

[9] Bergen, J., Anandan, P., Hanna, K., and Hingorani, R. (1992). Hierarchical model-based motion estimation. In *Computer Vision—ECCV'92*, pages 237–252. Springer.

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

[11] Black, J., Gargesha, M., Kahol, K., Kuchi, P., and Panchanathan, S. (2002). A framework for performance evaluation of face recognition algorithms. *ITCOM, Internet Multimedia Systems II, Boston*, pages 478–482.

[12] Blanz, V., Mehl, A., Vetter, T., and Seidel, H. P. (2004). A statistical method for robust 3D surface reconstruction from sparse data. *2nd International Symposium on 3D Data Processing, Visualization, and Transmission, 3DPVT*, pages 293–300.

[13] Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co.

[14] Blanz, V. and Vetter, T. (2003). Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1063–1074.

[15] Bledsoe, W. W. (1964). The model method in facial recognition. *Panoramic Research Inc., Palo Alto, CA, Rep. PR1*, 15.

[16] Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, San Jose, California. ACM.

[17] Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585.

[18] Chua, C. S. and Jarvis, R. (1997). Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85.

[19] Chui, H. and Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141.

[20] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.

[21] Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59.

[22] Daum, H. (2005). Influences of image disturbances on 2D face recognition. In *Audio- and Video-Based Biometric Person Authentication*, pages 900–908. Springer Berlin / Heidelberg.

[23] Dhond, U. R. and Aggarwal, J. K. (1989). Structure from stereo-a review. *IEEE Transactions on Systems Man and Cybernetics*, 19(6):1489–1510.

[24] Durou, J. D., Falcone, M., and Sagona, M. (2008). Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer vision and image understanding*, 109(1):22–43.

[25] Edwards, G., Taylor, C., and Cootes, T. (1998). Interpreting face images using active appearance models. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 300–305.

[26] Ezzat, T. and Poggio, T. (1997). Facial analysis and synthesis using image-based models. In *Algorithms for robotic motion and manipulation: 1996 Workshop on the Algorithmic Foundations of Robotics*, page 449. AK Peters, Ltd.

[27] Faggian, N., Paplinski, A. P., and Sherrah, J. (2006). Active appearance models for automatic fitting of 3D morphable models. In *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*. IEEE Computer Society Washington, DC, USA.

[28] Fleishman, S., Drori, I., and Cohen-Or, D. (2003). Bilateral mesh denoising. *ACM Transactions on Graphics (TOG)*, 22(3):950–953.

[29] Foley, J. D., Dam, A. V., Feiner, S. K., and Hughes, J. F. (1995). *Computer graphics: principles and practice*. Addison-Wesley Professional.

[30] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[31] Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660.

[32] Gibson, J. J. (1986). *The ecological approach to visual perception*. Lawrence Erlbaum.

[33] Hansen, P., Jaumard, B., and Lu, S. H. (1992). Global optimization of univariate lipschitz functions: I. survey and properties. *Mathematical Programming*, 55(1):251–272.

[34] Horn, B. K. P. (1970). Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. *AITR-232*.

[35] Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203.

[36] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417–441.

[37] Hu, Y., Jiang, D., Yan, S., and Zhang, L. (2004). Automatic 3D reconstruction for face recognition. *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 843–848.

[38] Ikeuchi, K. and Horn, B. K. P. (1981). Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17(1-3):141–184.

[39] ISO/IEC 19794-5:2005 (2005). *Information technology - Biometric data interchange formats - Part 5: Face image data*. ISO, Geneva, Switzerland.

[40] Jarvis, R. A. (1983). A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139.

[41] Jolliffe, I. T. (2002). *Principal component analysis*. Springer New York.

[42] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331.

[43] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern analysis and Machine intelligence*, 12(1):103–108.

[44] Krotkov, E. (1988). Focusing. *International Journal of Computer Vision*, 1(3):223–237.

[45] Lanitis, A., Taylor, C. J., and Cootes, T. F. (1995). Automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401.

[46] Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164–168.

[47] Li, H. and Hartley, R. (2007). The 3D-3D registration problem revisited. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

[48] Little, G., Krishna, S., Black, J., and Panchanathan, S. (2005). A methodology for evaluating robustness of face recognition algorithms with respect to variations in pose angle and illumination angle. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia.

[49] Loéve, M. (1963). *Probability Theory*. Van Nostrand Company, Princeton, NJ, 3rd edition.

[50] Lu, X. and Jain, A. K. (2005). Deformation analysis for 3D face matching. In *Applications of Computer Vision and the IEEE Workshop on Motion and Video Computing, IEEE Workshop on*, volume 1, pages 99–104, Los Alamitos, CA, USA. IEEE Computer Society.

[51] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, volume 3.

[52] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441.

[53] Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc.

[54] Meytlis, M., Sirovich, L., and Place, G. L. L. (2007). On the dimensionality of face space. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 29:2007.

[55] Mitchell, S. C., Bosch, J. G., Lelieveldt, B. P. F., van der Geest, R. J., Reiber, J. H. C., and Sonka, M. (2002). 3-D active appearance models: segmentation of cardiac MR and ultrasound images. *IEEE Transactions on Medical Imaging*, 21(9):1167.

[56] Murphy-Chutorian, E. and Trivedi, M. (2008). Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1).

[57] Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

[58] Patel, A. and Smith, W. (2009). 3D morphable face models revisited. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 0, pages 1327–1334, Los Alamitos, CA, USA. IEEE Computer Society.

[59] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572.

[60] Peer, P. (2010). CVL face database, ŠCV, PTERŠ, Velenje. `http://www.lrv.fri.uni-lj.si/facedb.html`. Accessed on July 6, 2010.

[61] Pentland, A. (2000). Looking at people: sensing for ubiquitous and wearable computing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):107–119.

[62] Pentland, A. P. (1984). Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 661–674.

[63] Phillips, P. J., Grother, P., Micheals, R., Blackburn, D. M., Tabassi, E., Bone, M., and DARPA, A. (2003). Face recognition vendor test 2002. *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*.

[64] Phillips, P. J., Scruggs, W. T., O'Toole, A. J., Flynn, P. J., Bowyer, K. W., Schott, C. L., and Sharpe, M. (2007). FRVT 2006 and ICE 2006 large-scale results. *NIST, Tech. Rep. NISTIR*, 7408.

[65] Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317.

[66] Posdamer, J. L. and Altschuler, M. D. (1982). Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1–17.

[67] Rohr, K. (2001). *Landmark-Based Image Analysis: Using Geometric and Intensity Models*. Kluwer Academic Publishers.

[68] Romdhani, S. (2005). *Face Image Analysis using a Multiple Feature Fitting Strategy*. PhD thesis, PhD thesis, University of Basel, Computer Science Department, Basel.

[69] Romdhani, S. and Vetter, T. (2003). Efficient, robust and accurate fitting of a 3D morphable model. *Proceedings of the International Conference on Computer Vision*, 1:59–66.

[70] Scheenstra, A., Ruifrok, A., and Veltkamp, R. (2005). A survey of 3D face recognition methods. In *Audio- and Video-Based Biometric Person Authentication*, number 3546 in Lecture Notes in Computer Science, pages 891–899. Springer Berlin / Heidelberg.

[71] Storer, M., Urschler, M., Bischof, H., and Birchbauer, J. A. (2008). Face image normalization and Expression/Pose validation for the analysis of machine readable travel documents. In Kuijper, A., Heise, B., and Muresan, L., editors, *Proceedings 32nd OAGM/AAPR Conference*, volume 232, pages 29–39.

[72] Suikerbuik, R., Daanen, H., and Tangelder, H. (2004). Automatic feature detection in 3 d human body scans. In *Proceedings of the conference "SAE Digital Human Modelling for Design and Engineering"*.

[73] Tanaka, J. W. and Farah, M. J. (1993). Parts and wholes in face recognition. *The Quarterly Journal of Experimental Psychology Section A*, 46(2):225–245.

[74] Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

[75] Vetter, T. and Poggio, T. (1997). Linear object classes and image synthesis from a single example image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):733–742.

[76] Viola, P. A. (1995). *Alignment by Maximization of Mutual Information*. PhD thesis, Massachusetts Institute of Technology.

[77] Wang, C., Yin, B., Shi, Q., and Sun, Y. (2005). 3D face correspondence based on uniform mesh resampling combined with mesh simplification. *Journal of Computational Information Systems*, 1(2):343–350.

[78] Weyrauch, B., Heisele, B., Huang, J., and Blanz, V. (2004). Component-Based face recognition with 3D morphable models. *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, pages 85–85.

[79] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., and Shum, H. (2004). Mesh editing with poisson-based gradient field manipulation. In *ACM SIGGRAPH 2004 Papers*, pages 644–651, Los Angeles, California. ACM.

[80] Zhang, R., Tsai, P. S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706.

[81] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Computing Surveys (CSUR)*, 35(4):399–458.