THREE-DIMENSIONAL FACE MODELLING & ANALYSIS

Karl Skoglund

IMM-THESIS-2003-43

IMM

Printed by IMM, Technical University of Denmark

Preface

This thesis has been prepared at the department of Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU). The project started February 3rd, 2003, and finished August 1st, 2003. The amount of work answers to 35 ECTS academic points.

The project is part of the author's M.Sc. studies in Computer Science and Engineering at the Lund Institute of Technology (LTH). LTH is part of Lund University, Lund, Sweden. The completed and accepted thesis gives 20 Swedish academic credits.

While working on the project, the author applied for and was granted a Ph.D. scholarship. Studies will begin November 1st 2003.

The author would like to thank his supervisor, Assistant Professor Rasmus Larsen, his Swedish examiner, Professor Kalle Åström (LTH), the people at 3D-Lab - Professor Sven Kreiborg and Assistant Research Professor Jon Sporring, Ph.D. students Brian Lading, Mikkel B. Stegmann and Ramus R. Paulsen, and Ph.D. Klaus Baggesen Hilger. All except Kalle Åström, Sven Kreiborg and Jon Sporring is at IMM, DTU.

Karl Skoglund, IMM, September 3, 2003.

Abstract

This thesis describes methods for modelling the shape, texture and appearance of human faces in three dimensions. These models provide a foundation for applications such as image segmentation and object recognition.

The models use a data set of 24 faces acquired using a 3-D scanner. To make the data useful in a statistical context, the representation of the objects must be unified, a process called registration. The registration method described here uses a set of nine manually placed points of correspondence on each face. These are used to form a dense correspondence of thousands of points. This makes sure the model is able to capture the subtle details of a human face.

The thesis presents the mathematical methods, with in-depth descriptions of key concepts, as well as the outlines of the implementation. The capabilities of the models are demonstrated in applications such as 2-D face segmentation and fully automated face registration.

The results are promising, but calls for higher quality and quantity of face data. The models were successfully used to automatically register new face scans, however the image segmentation, implemented using a simple optimization algorithm, fails in most cases. A number of suggestions for future work are given, for example the implementation of a 3-D active appearance model for face recognition.

Keywords: Statistical Image Analysis, Shape Analysis, Shape Modelling, Appearance Modelling, 3-D Registration, Face Model, 3-D Modelling, Face Recognition, VTK.

Contents

List of Figures

Chapter 1

Introduction

The human face is one of the most popular and best understood objects in research areas such as statistical shape analysis, object recognition, feature extraction, synthesis and tracking. The reason for this is the importance of the face in our daily life. The appearance of faces helps us determine characteristics such as gender, age and race, along with more intricate properties such as mood and health status. Because the appearance of faces is so important, the human has developed into an expert in face interpretation and recognition. Our "database" spans thousands of faces learned through our whole lives, yet any face can instantly be recognized, even though it might have changed over time. This appearance expertise puts high demands on an area such as face synthesis.

There are several difficulties involved in creating a model of a human face. One problem is that the shape of a face is very complex. Building a purely mathematical model, for example with a set of parameterized curves, will produce unnatural and unrealistic faces. Another problem is the great variability found in faces. Age, gender and race are sources of great diversity, but the variation is also *specific*. Even a minor change in shape or texture can make a face unacceptable to a human observer. Several attempts have been made to address these problems. So far, the most popular and successful method has been the *appearance model*.

An appearance model is a deformable model built from a database of examples. The model can be used in many different applications, the most

important being face synthesis, segmentation and recognition. The matching algorithm used for interpreting images is called the *active* appearance model.

This thesis presents the design of a three-dimensional appearance model. Models in three dimensions are still fairly uncommon because of the large amount of overhead and computational difficulty involved in working with a three-dimensional data set. However, as computer power increases and equipment for acquiring three-dimensional data becomes reasonably priced, this type of model will be more widely used.

Even though this material deals with human faces exclusively, the methods described herein are applicable to most three-dimensional surface data.

The next chapter will present the relevant precursors to the (active) appearance model. These methods deal with image data in two dimensions. The most important attempts at extending the appearance model to three dimensions will also be described. The "preliminaries" chapter will give a brief explanation of the basic concepts in shape analysis. This should make the rest of the report understandable to people outside the field of computer vision. "Methods and Materials" presents the ideas and algorithms of the model, as well as giving a presentation of the imaging equipment used. The "Implementation" chapter explains how the presented algorithms can be realized on a computer. This is followed by a presentation of the results, a summary, and ideas for future work. The appendices cover some of the mathematical methods more in detail, and shows images from the database of faces.

Chapter 2

Background

This thesis presents a method of creating a statistical model of human faces. Model-based interpretation of images is one of the main disciplines in computer vision. Several types of models exist although the application and useability of these varies. This chapter will list a few models that can be seen as precursors to the (active) appearance model. It will also present attempts at extending the model to three dimensions.

2.1 Precursors of Appearance Models

2.1.1 The Golden Image

The simplest way of creating a model of an object is to use a single image. If the variability of the object class is low, this model can successfully be matched to a new image. According to [8], this method has successfully been used to locate brain structures in magnetic resonance images.

To precisely match and segment structures of objects with varying appearance, the model needs to be deformed. Since the golden image used here is static, this is not possible. Therefore, a more advanced model is needed.

2.1.2 Eigenfaces

The concept of eigenfaces was introduced by Turk and Pentland in 1991 [27]. The idea is to create a 2-D model of the human face using a database of example images, also known as a *training set*. Each image of $k \times n$ pixels is represented by a single vector in kn-dimensional space. The model is built by finding a compact representation of the subspace defined by the training set. This is done using principal component analysis, which yields a parameterized, efficient model. This is then used to interpret images.

The main difference between eigenfaces and the modelling of texture variation in appearance models, is the lack of landmarks. No points of correspondence are defined in the eigenface approach, and therefore it is impossible to use methods such as Procrustes analysis to filter out differences in location, orientation and scale in the images of the training set. This makes the eigenface method less accurate.

2.1.3 Contour Models

The concept of active contour models (ACM), or *snakes*, was introduced by Kass *et al.* [16]. A snake is a physically based model, defined by an energy minimizing spline. The model can be used to find structures with high gradients, contours, in an image. Since the model is not based on statistical information on a specific class of objects, it is very general. Therefore, it can be matched to any type of object.

To find a certain structure in an image, a set of control points are manually defined to construct an initial spline. The snake is then iteratively deformed to fit to an appropriate solution around the desired contour. The energy constraints guiding the snake are divided into internal and external forces. The internal forces are elasticity and rigidity of which the former makes the control points stay regularly inter-spaced, while the latter keeps the curve smooth. The image gives rise to the external forces; the control points are attracted to edges and corners - high gradients.

2.1.4 Shape Models

In 1995 Cootes *et al.* introduced the concept of shape models [9] where shapes are defined by a set of landmarks. From a database of shapes, a

statistical point distribution model (PDM), or shape model, can be built, describing the main modes of shape variation. New shapes can be constructed by changing a set of parameters of the model. As long as the parameters are within certain constraints, the model will produce plausible shapes.

The shape model can be used for matching to information in images. The method is called *active* shape models. The shape model is given an initial position somewhere in the image, and is then iteratively translated, rotated, scaled and deformed until a solution is found. Since the model is defined by its parameters and since these only generate plausible shapes, the algorithm will not match to image structures which cannot be described by the model. This makes the matching procedure robust and accurate. The obvious drawback of the method is that it only handles object of a certain class.

2.2 Appearance Models

In 1998 the shape model was extended to include texture, the intensity values contained by the shape boundaries. Such models are called *appearance models* [8]. The parameters of the appearance model control the shape and texture simultaneously. This makes it possible to synthesize photo-realistic new examples.

The most common application for an appearance model is the *active* appearance model [23] which can be used to match, segment and interpret information in images. The basic idea of the algorithm is to provide a-priori knowledge on how to correct the parameters of the model according to the current residuals between the model and the image. This is done using principal component regression.

2.2.1 Three-dimensional Appearance Models

The original appearance model formulation was for two-dimensional images. The majority of work on appearance models since, has been done using two-dimensional data. The reason for this is the ease of gathering and annotating data, and the low demands for computational power.

To be able to synthesize and match to images where the pose of the object varies significantly, there are two basic approaches. One is to construct a

two-dimensional model that includes different viewing angles. These are called view-based active appearance models [24]. The other is to keep a three-dimensional model, which naturally can be used to synthesize images of objects from any viewpoint. This section will list relevant work using this method.

Mitchell *et al.* describe the building of a three-dimensional appearance model from volumetric cardiac magnetic resonance (MR) images [22]. This paper also describes the implementation of an *active* appearance model for image segmentation and recognition.

Blanz and Vetter show how a three-dimensional morphable model of human faces can be built [4]. Although the model is similar to the appearance model, separate models for shape and texture are used. The dense point-to-point correspondence between shapes are formed using an optical flow algorithm along with a bootstrapping method for automatic registration. To fit the model to two-dimensional images, a gradient descent optimization function is used.

Hutton *et al.* build a dense correspondence model of the human face using a semi-automatic algorithm [26]. Each face is manually annotated with a sparse set of landmarks which are used to form the dense correspondence. This is the method used in this thesis, with a minor modification suggested by Paulsen [18].

Chapter 3

Preliminaries

In this chapter, a brief explanation of the basic concepts in statistical shape analysis will be given. This knowledge is a prerequisite for the rest of this thesis. For those already familiar with these terms, it might serve as useful remainder.

3.1 Shape Definition

Statistical shape analysis is concerned with the shapes of objects of a specific class. The shape class can be anything from the shape of cars to the shape of the human brain. Since man-made objects seldom contain any interesting or uncharted variation, shape analysis most often involves data from disciplines such as medicine, biology, image analysis, geography and archaeology.

There is no strict definition of shape, but the most intuitive and common definition is given by D.G. Kendall (1977):

> **Shape** is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.

In other words, shape is invariant under what is called *similarity transforms*. These transforms rotates, scales and translates an object. A transformation that only rotates and translates an object is called a *rigid-body transform*. This type of transform leads to the following definition

> **Size-and-shape** is all the geometrical information that remains when location and rotational effects are filtered out from an object.

Two objects have equal shape if one can be transformed using a similarity transform so that it perfectly matches the other. This is called shape alignment. If a rigid-body transform is sufficient, the objects have equal size-and-shape. Statistical shape analysis often involves working with a set of similar, but not equal shapes. It is therefore necessary to obtain the best possible alignment, and then measure the distance between pairs of shapes. This can be done using Procrustes analysis, a topic covered in appendix D.

3.2 Landmarks

It is intuitive to define a shape by its outline, contour or surface. The question is how to represent this in mathematically useful manner. One way is to define implicit or parametric curves or surfaces as functions [3]. A simpler way is to define a set of points along the outline or surface. These points are called *landmarks*.

> A **landmark** is a point of correspondence on each object that matches between and within populations.

Three basic types of landmarks exist

Anatomical landmarks are points of correspondence assigned by an expert in positions motivated by the object type. Such positions include the tip of the nose or the corner of the mouth on a human face, the base of a monkey's scull, or the start of the line forming a handwritten digit.

Mathematical landmarks are points connected to some mathematical or geometrical property of an object. These might be in positions of high curvature, at an extreme point or similar.

Pseudo-landmarks are made-up points dependent on other landmarks. A common example of pseudo-landmarks is a set of equidistant points between two anatomical landmarks.

Shape analysis is preformed using the landmarks. To keep track of the correspondence of landmarks between objects, they must be labelled in some way. The easiest way of labelling landmarks is to keep the landmarks of an object in a list, and make sure that the landmark order is the same for all objects. Assume the shape of k objects $\mathbf{x}_i$, $1 \leq i \leq k$, in m dimensions are each marked with n landmarks. One possible shape representation is then

$$\mathbf{x}_i = [x_0^0 \ \ldots \ x_n^0 \ \ldots \ x_0^m \ \ldots \ x_n^m], \quad i = 1 \ldots k \tag{3.1}$$

where x_i^j is the ith point of the jth dimension. To clearify, consider k objects with three points in two dimensions. The representation then becomes

$$\mathbf{x}_i = [x_0 \ x_1 \ x_2 \ y_0 \ y_1 \ y_2], \quad i = 1 \ldots k \tag{3.2}$$

Of course, any other representation is just as good, however this is the one used in this thesis along with many other applications.

3.2.1 Placing Landmarks

The process of defining a shape by placing landmarks is called *annotation* or *registration*. This can be done manually, semi-automatically or automatically. Shapes in two dimensions may require hundreds of landmarks to capture the shape, and three-dimensional shapes may require magnitudes more. This has sparked the development of semi or fully automated registration algorithms. In this thesis, a semi-automatic algorithm is used [26, 18]. Fully automated registration algorithms exist for general shapes. These have proved to be fairly successful for data in two dimensions. One such method is Minimum Description Length (MDL) [25]. This method generalizes to three dimensions, but very few implementations yet exist.

3.3 Shape Analysis Data

Data used for shape analysis can be in any form. The majority of two-dimensional data comes in the form of digital raster images, but vectorial images are also possible. As long as the images reside in some form of coordinate system, landmarks can be defined and shape analysis preformed. Three-dimensional data is more complicated to represent. The most common representation is polygonal data, where surfaces are defined as the area

enclosed by a set of 3-D points. See section 4.1.2 for a detailed description of one way to represent 3-D polygonal data.

3.4 Terminology

In this thesis, a number of synonyms are used which are listed here.

- Object, face, scan, example, sample, shape (the shape of an object, as opposed to the texture)
- Bringing objects into correspondence, registration, forming a (sparse or dense) correspondence
- Unregistered object, novel object, new object
- Training set, examples, object database

Chapter 4

Methods and Materials

This chapter will cover procedures for data acquisition and the mathematical methods for building models and applications. Chapter 5 will give the outlines of an implementation of these methods and chapter 6 renders the results.

4.1 Data Acquisition

An important step towards building a 3-D appearance model is the data gathering. Building a data set of two-dimensional images is rather straightforward, requiring only a digital camera, an appropriate setting, suitable lighting equipment and people to photograph. Three-dimensional data, on the other hand, requires rare and expensive equipment, more commitment from the people being registered and more time. As the models become rather large in size, data storage and transfer can also pose a problem.

4.1.1 Hardware

The data was acquired using a *Minolta Vivid 900* laser scanner provided by the 3D-Laboratory at the School of Dentistry, University of Copenhagen.

The camera has a single CCD[1] which registers both the reflected laser beam and the digital image used for texturing. The scanner performs the following steps for a single data registration:

1. The scanner probes the object in front of it using the laser beam. This is done to set the target area for the laser to scan.
2. The scanner forms a horizontal laser plane which scans the object top down. The reflections are measured by the CCD and stored as 3-D points.
3. Finally, the CCD is used to acquire a digital 2-D image. The CCD is monochrome, so to register a color image, three images are registered, each with a different filter put in front (red, green, blue).

The laser beam, as any directed light, casts shadows. When a human face is scanned, protruding parts, such as the nose and the chin, cause problems with shadows, leaving parts of the face unregistered. The amount of shadows is dependent on the angle from which the scanning is performed, however, no single angle can capture the whole face area. This is solved by scanning a face from multiple angles, and then merging the data. Extensive testing was done to find the optimal angles and number of scans. Since the scanner weighs roughly 20 kilos with the tripod, it proved to be easier to rotate the person being scanned than moving the camera. To facilitate this, a dentist's chair was used which can be raised, lowered and turned in an exact fashion.

It is difficult for the person being scanned to keep absolutely still, and to establish the exact same pose before each scan. For this reason, as few scans as possible should be used, but too few scans result in an incomplete face representation. With careful positioning of the scanner and the object, as little as three scans suffice. By putting the scanner in a slightly lower position than the person being scanned, a decent representation of the chin and nostrils can be achieved. To register the whole face, including the cheeks and the sides of the nose, the three scans were performed from $0°$ and $\pm 30°$.

The drawback of moving the person instead of the camera is the change in lighting conditions. As the person rotates, the face will be differently illuminated. This is the case when the ambient light of the room is directed instead of diffuse. The light should therefore be as diffuse as possible. To

[1] CCD is short for "Charged Coupled Device" and is the component capturing an image in digital cameras, similarly to the photographic film of classic cameras.

achieve this, professional lighting equipment for common photography was used. This consisted of two 1000 watt lamps mounted on tripods. The light from these lamps were bounced of parabolic reflectors, resulting in diffuse lighting conditions. However, perfect diffuse light is very hard to achieve, and the placement of the lighting equipment was crucial for high quality results. The optimal setting proved to be one light on each side of the camera, approximately 0.6 meters perpendicular to the direction of photography. One light was placed at face height and the other slightly higher. Both lights were directed towards the face.

Figure 4.1 shows the camera flanked by the lighting equipment.

4.1.2 Scanner Software

The Minolta scanner comes with a 3-D data processing software called *Polygon Editing Tool*. Using this, the camera can be controlled and data can be imported, processed and exported.

After scanning and importing the three views that make up a complete face scan, the program shows the three scans represented as polygon meshes with texture in the same frame. The meshes are placed as they were scanned, i.e. the side views are rotated and therefore out of place. To merge the separate views into one, the software uses an unknown registration algorithm, possibly ICP (see appendix C), to align the meshes. This works surprisingly well as long as the overlap is significant. In almost all cases, the software was able to align the surfaces without manual guidance. When the surfaces are aligned, they can be merged. The merged representation is then saved, and the individual views are discarded.

Not only the 3-D data is merged. The three texture maps are also merged, i.e. for each 3-D point, a decision must be made to which texture map the point should be linked. Because of the imperfect lighting conditions, the merged texture has many neighboring pixels with large differences in illumination. This effect can be reduced with a built-in function for texture blending, which makes sure transitions between the three texture maps are smooth.

Despite merging three scans, there might still exist small unregistered areas. These show as holes in the polygon mesh. The software aids in finding these holes and eliminates them by inserting new points and polygons. This is done so that the local curvature of the mesh is preserved.

Figure 4.1: The scanning setup. The lens can be seen on the upper front part of the camera. The opening below holds the laser.

When the post-processing of the data is finished, the result is saved using a suitable format. Minolta's own format produces binary files, and no documentation on how the format works exist. It is therefore not useable outside the Polygon Editing Tool. Luckily, the program is able to export the file to a few other formats. The only non-binary (ASCII) format that saves all information, including texture and texture coordinates (see below), is VRML. VRML is an abbreviation for *Virtual Reality Modelling Language* and is a format for creating 3-D graphics for the web. A typical VRML file contains an object description consisting of 3-D points, polygons, coloring and texture. It can also hold animation specifications, lighting parameters etc. Refer to `www.web3d.org/VRML2.0/FINAL/` for the full specification. Since the file format is easy to understand and read, and because the models can be investigated using a web browser, this format was chosen as the most suitable.

The finished models consist of around 30 000 3-D points, saved as triplets of floating point numbers. Roughly the same amount of vertex references make up the polygons. The texture map is included in the file as hexadecimal numbers. Six hex numbers represent a pixel. The first two denote the amount of red, from 0 (0 hex) to 255 (FF hex). The middle two numbers represents green and the last two blue. 320 000 such numbers make up the whole texture map, which results in an 800×400, 24 bit color image. To be able to represent the texture of a 3-D surface by a 2-D image, the textural data must somehow be projected onto a flat surface. The projection used here is cylindric, which is suitable for faces, since a face can (very crudely) be approximated by a vertical cylinder. To map the 3-D points to the texture image, *texture coordinates* are used. For each 3-D point, there is a texture coordinate telling where in the texture map this point has its color information. The texture coordinates are normally denoted (s, t) and range from $0 \leq s, t \leq 1$. This mapping actually defines a bivariate function $(s, t) = (f(x), g(x))$ where $x \in \mathbf{Z}$ is a point index and s and t are the resulting coordinates. The model's polygons are textured using interpolation of the texture-coordinates of each vertex.

Below is a simple example VRML file, which defines a triangle with texture coordinates. Note that only the first three pixels of the texture map are represented.


```
#VRML V1.0 ascii
...
    Texture2 {
        image 800 400 3
        0x1a0010 0x1b0109 0x1a0008
...
    }
    Coordinate3 {
        point [
        0.00  0.00 0.00,
        10.00 0.00 0.00,
        5.00  5.00 0.00
        ]
    }
    IndexedFaceSet {
        coordIndex [
        0, 1, 2, -1
        ]
    }
    TextureCoordinate2 {
        point[
        0.120453 0.114399,
        0.003498 0.001515,
        0.357738 0.424964
        ]
    }
...
```

4.1.3 The Face Database

24 faces were scanned during two major and a few minor sessions. Ages ranged from 20 to 40, plus one child. Most of the people were students and staff from the Technical University of Denmark, hence, many of the subjects were male and of Scandinavian origin. The age, gender and race distribution is therefore limited. When constructing a database of faces used for modelling, the usual objective is to make the model as general as possible. If the model is going to be used for a specific purpose, it might make sense to create a more specific model. The limited variation and number of scans puts high constraints on the model.

The scanner is not able to register hair, so a full head representation is not possible to acquire. Eyes are also hard to register since the reflected laser beam is too dispersed to capture. Therefore, all scans are performed with the eyes closed.

A scan from one angle takes approximately 5 seconds. The total scanning process lasts approximately two minutes, and including post-processing, the total time is around 15 minutes per person.

The resulting shape and texture data is partially of poor quality.

- The shape is well represented but has a rough surface. This is a result of the difficulty in maintaining the exact same pose throughout the whole scanning process.
- The texture projection from surface to cylinder have resulting artifacts. Areas at (almost) right angles to the cylinder have insufficient mappings.
- The color balance is incorrect which might be a result of the type of lighting used. The camera is, according to the user's manual, made to operate in "office lighting". The color temperature of the equipment used is similar to that of daylight. The problem shows as low intensity of green colors, or equivalently, an excess of red and blue tones.
- The texture-coordinate mapping has missing entries. This shows as the mapping $(s, t) = (0, 0)$. Although this mapping is valid, it is incorrect, something which is easily seen when the scans are examined.

Despite this, the database should be useful for anyone interested in three-dimensional modelling. As will be shown, the data quality is sufficient for creating a useful model. Appendix E shows 2-D images of the faces in the database.

The work with acquiring the data and finding the optimal scanning setup was performed together with Ph.D. student Brian Lading, IMM, Technical University of Denmark.

4.2 Model Creation

This section describes a method for building a three-dimensional appearance model from a training set of human faces. Only the mathematical methods and the outlines of the algorithms are presented here. How this is implemented in code is described in chapter 5.

4.2.1 Shape Registration

When constructing a 2-D shape- or appearance model, each object in the data set is annotated with corresponding landmarks. Around 60 landmarks suffice for a 2-D facial image. In three dimensions, thousands of landmarks are required to capture the complex surface of a human face. Obviously, it is not feasible to annotate these by hand. Instead, some sort of automated process is necessary. A semi-automatic algorithm is used here, which constructs a dense distribution of corresponding points from a sparse set of manually placed landmarks [26].

The unregistered face data is unordered. This means that a certain 3-D point can be placed anywhere on a face. On one face the point can be part of the ear, while on another it is part of the nose. If each point represented the same position on every face, they would be in correspondence. The idea of the registration algorithm is to change the order of the points to give all objects the same representation. If all objects have an equal amount of points and the same point ordering, every point will act as a landmark. Since a face consists of tens of thousands of points, there will be enough landmarks to perform statistically satisfying shape analysis.

The Template Shape

The algorithm uses one of the objects as a *template shape*. The idea is to pick a suitable face as the template, and then change the extent and point ordering of the other faces to match the template.

The template should be well represented and have an "average" shape.

Since the registration algorithm will give all the faces in the database the same number of points as the template, it is essential that the area represented by each point is present in all the other objects. To ensure this, the objects are carefully examined. The template is then pruned so that the resulting face area is present in all examples.

Defining a Sparse Set of Landmarks

Each object is manually annotated using a 3-D landmarking program, *ISA*, developed by Rasmus Paulsen, IMM, DTU. This software was originally used to annotate 3-D models of human ear canals, and had to be improved

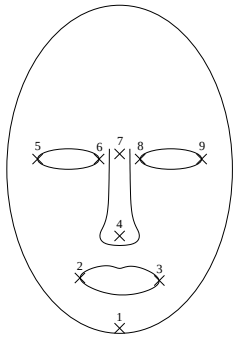


Figure 4.2: A schematic face with enumerated landmarks.

Number	Description
1	The tip of the chin
2	The left corner of the mouth
3	The right corner of the mouth
4	The tip of the nose
5	The left corner of the left eye
6	The right corner of the left eye
7	The minimum of the curve defined by the adjoining of the nose and the forehead
8	The left corner of the right eye
9	The right corner of the right eye

Table 4.1: The sparse set of manually defined landmarks

to be able to work with face data. The most important improvement was the addition of the ability to work with textured objects. Placing a landmark at the corner of an eye is much easier using the texture as guidance than just the shape.

Nine landmarks were used. These are shown in figure 4.2 and listed in table 4.1. Note that all landmarks are anatomical, except landmark seven which is of mathematical type.

Forming the Dense Correspondence

The sparse set of landmarks can now be used to form a dense set of thousands of landmarks.

The template face is deformed [18] to roughly fit the shape to be registered using a thin-plate spline warp [6]. The thin-plate spline warp consists of a non-linear transform that stretches and bends an object to fit to certain points. The transformation is defined by the sparse landmarks. The landmarks of the template act as *source* points, and the landmarks of the object to be registered are the *target* points. The transform warps the template so that the source points coincide with the target points. The rest of the template's shape is altered as little as possible. This is an effect of the thin-plate spline warp being defined by an energy-minimizing function of a thin steel plate. See appendix B for more information on thin-plate splines.

The shapes are now very similar in shape. This can be used to easily form a dense correspondence of points between the template and the unregistered object. As an example of how this is done, take any point in the template mesh. Find the closest point on the *surface* of the other object. This new point is part of the novel object and corresponds to the point of the template. Doing this for all points results in the following registration algorithm:

1. Warp the template shape using the sparse set of landmarks of each object. This makes sure the shape of the template is similar to the shape of the new object.
2. For each template point, find the closest point on the neighboring surface
3. Discard the old points of the object, and replace them with the new registered points

Note that the correspondence is found *point-to-surface* instead of point-to-point. If the point distribution is very dense, the methods are almost equal, but if the distance between points can be noticeable, the stated algorithm gives better results. Figure 4.3 shows the registration process for 2-D curves. Note that it generalizes to three dimensions.

Each point of the new object now corresponds to the same point of the template, while the shape of the object is preserved. The registered object will also contain less points than before. This means that the registered object is automatically pruned according to the template.

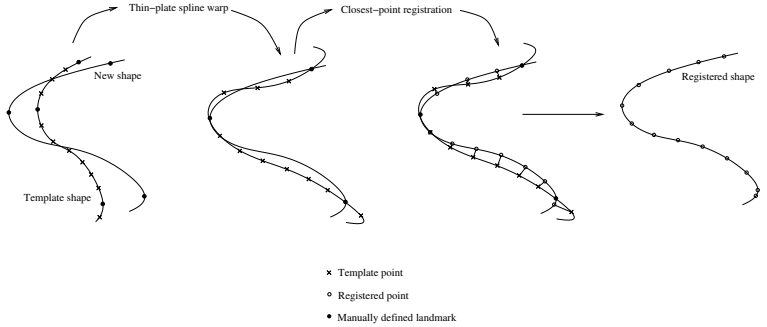


Figure 4.3: The registration process shown with 2-D curves.

Since the point ordering and extent of the registered object have changed, the old polygonal vertex references are now invalid. However, since the representation of the registered object is approximately the same as for the template, the polygon references of the template can be used. These are therefore simply copied into the new object. The same goes for the texture coordinates, although this requires that the texture representation of the template and the registered objects are similar. This topic is covered in 4.2.3.

4.2.2 Shape Alignment

The faces now have a uniform shape representation, but location and orientation still differs somewhat. Before statistical shape analysis can be performed, the shapes must be rotated and translated so that only the shape and size, as defined in chapter 3, remains. To filter out the pure shapes from a set of objects, scale needs to be removed as well. Since the data used here was acquired using a laser scanner, which measures the actual size of an object, the scale of the objects is invariant. Differences in size derives from the actual inconsistencies in face size among the people scanned. This is an interesting feature of the model so these differences are left unaltered, resulting in a size-and-shape model (see chapter 3).

The shape alignment is performed using Procrustes analysis. This technique was originally introduced for applications in psychology as early as 1939. The algorithms used here were mainly developed by Gower (1971,

1975) and Ten Berge (1977). A thorough description of Procrustes methods can be found in appendix D, but the essential methods for three-dimensional data will be rendered here.

Metrics and Representations

Shapes in three dimensions consisting of n_s points $\mathbf{s}_i = (x_i, y_i, z_i)^T$ can be represented by a single vector in $\mathbf{R}^{3n_s}$

$$\begin{aligned} \mathbf{s} &= (\mathbf{x}^T, \mathbf{y}^T, \mathbf{z}^T)^T = \\ &= (x_1, \ldots, x_{n_s}, y_1, \ldots, y_{n_s}, z_1, \ldots, z_{n_s})^T \end{aligned} \tag{4.1}$$

The set of k shapes now consists of k points in $3n_s$-dimensional space.

Another way to represent an object is in matrix form. The columns are $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ respectively and one row equals one point. The matrix is therefore of size $(n_s \times 3)$.

$$S = [\mathbf{x}\ \ \mathbf{y}\ \ \mathbf{z}] \tag{4.2}$$

Both representations will be used here.

The *centroid* of a shape is defined as the center-of-mass of all landmarks, where a landmark is considered to be of unit mass.

$$\mathbf{s}_c = (\bar{x}, \bar{y}, \bar{z})^T = \frac{1}{n_s}\left(\sum_{i=1}^{n_s} x_i, \sum_{i=1}^{n_s} y_i, \sum_{i=1}^{n_s} z_i\right)^T \tag{4.3}$$

The *Procrustes mean* of all shapes is simply the vector built from the mean of each corresponding point.

$$\bar{\mathbf{s}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{s}_i \tag{4.4}$$

For completeness, although scale is not filtered out from the objects here, the *shape size metric* commonly used is the *centroid size*.

$$T(\mathbf{s}) = \sqrt{\sum_{i=1}^{n_s} (x_i - \bar{x})^2 + (y_i - \bar{y})^2 + (z_i - \bar{z})^2} \tag{4.5}$$

Removing location

To filter out location from the set of objects, they are translated so that their centroids match. The origin is a suitable matching location. Objects with centroid at the origin are said to be *centered*.

Removing orientation

To filter out differences in rotation, all shapes are aligned so that their orientation is equal to the orientation of the mean shape. Once this is done, calculating the mean anew will result in a different mean shape. The objects are therefore rotated again. This process is iterated until the mean is stable.

The rotation that optimally rotates one object to fit another can be found using *singular value decomposition* (SVD) [7]. This is done using the following method:

- Using the matrix representation of an object, denote the object to be rotated S_1 and the target object S_2.
- Construct a matrix M:

$$M = \frac{S_2^T S_1}{\|S_1\|\|S_2\|} \tag{4.6}$$

  where the norm $\|S\| = \sqrt{\text{trace}(S^T S)}$.
- Calculate the SVD of M:

$$M = V \Lambda U^T \tag{4.7}$$

- The optimal rotation is $R = UV^T$

This gives the follwing algorithm for removing orientation:

1. Calculate the Procrustes mean from all objects using equation 4.4
2. For each object, calulate the optimal rotation R that aligns the object to the mean and rotate the object using $S' = SR$
3. Iterate step 1 and 2 until convergence.

Convergence is declared when the RMS (Root Mean Square) value of the difference between the old and new mean vector falls below a certain threshold. Denote the mean of the last iteration $\bar{\mathbf{s}}_0$ and the mean of the previous

iteration $\bar{\mathbf{s}}_1$. The RMS value is then calculated as

$$\Delta RMS(\bar{\mathbf{s}}_0, \bar{\mathbf{s}}_1) = \sqrt{\frac{1}{n_s}\sum_{i=1}^{n_s}(\bar{s}_{0(i)} - \bar{s}_{1(i)})^2} \tag{4.8}$$

Another option is to use the squared norm of the difference vector

$$\|(\bar{\mathbf{s}}_0 - \bar{\mathbf{s}}_1)\|^2 \tag{4.9}$$

4.2.3 Texture Alignment

The template shape not only defines the polygon references of the final model, but also the texture coordinates. Therefore, the texture maps must be aligned so that each pixel represents the same part of all objects. All images already have the same size (the same number of pixels), but the location, size and extent of the content differs significantly. The texture maps can be given a uniform representation using a two-dimensional thin-plate spline warp, see appendix B. This requires that suitable source and target landmarks are defined. Since the 3-D points of all the shapes are now in correspondence, any set of 3-D points can define a set of 2-D landmarks by use of the texture coordinates.

For example, assume the shapes each have 20 000 3-D points, and a set of 20 points were chosen by selecting every thousand point starting with point number 1000. Each of these 20 points link to the texture map via the texture coordinates (s, t). If this is done for all shapes, each texture map will be annotated with 20 landmarks that can be used to define the warp. The 2-D landmarks of the template are the target points. The remaining question is how to select an appropriate set of 3-D points. Too few will result in an inexact warp and too many will cause too much bending of the area outside the landmarks. Also, any random choice of points may result in an inhomogeneous distribution of 2-D points. However, a suitable amount and distribution is defined by the set of 3-D landmarks listed in table 4.2. For each landmark position, the corresponding 3-D point of the shape is found and a 2-D landmark is registered.

The warp defined by these landmarks causes no distortion of the area outside the landmarks and is accurate enough. Because of the distribution, it is most accurate around the central part of a face which is important. The

areas around the cheeks and chin is not as important since the textures tend to be rather invariant there.

Just as the registration algorithm described in section 4.2.1 pruned all the shapes according to the template, the textures also need to be pruned. The only textural information needed is defined within the convex hull of the 2-D landmarks off *all* 3-D points. There is, however, a more efficient way of calculating this area. The outline of the template shape forms a three-dimensional closed loop. By using the texture coordinates, the outline can be mapped to 2-D. The area enclosed by this loop defines the extent of the texture maps needed by the model. Once this is defined, all the information outside the loop can be removed in all the texture maps.

Intensity Alignment

Even though the images have been aligned with respect to location, rotation and scale, they might still contain variation in overall intensity. The differences in individual pixel intensities is of course what defines the appearance of the different texture maps, but global intensity variation such as illumination and color balance can be filtered out to improve the specificity of the texture model.

By placing the intensity value of each pixel on a single axis it is understood that the intensities can be aligned using a one-dimensional Procrustes analysis. Since each pixel has three intensity values in color images, one each for red, green and blue intensity, three separate alignments are performed. This is preferred over a single three-dimensional Procrustes analysis, since it better handles differences in color balance. The discussion below concerns only one color component. The same method is used for the other two.

Let the intensities of a texture map of n_t pixels form a vector.

$$\mathbf{t} = [t_1, \ldots, t_{n_t}]^T \tag{4.10}$$

The analytical solution for planar ordinary Procrustes analysis is given in appendix D. The one-dimensional case is similar, except that real numbers are used instead of complex and rotations does not exist.

Assume that the mean texture $\bar{\mathbf{t}}$ is known. Assume also that all textures have been centered, i.e. the sum of each texture vector $\mathbf{t}^T\mathbf{1}_{n_t}$ is zero. To

find the optimal parameters that scales and translates the intensity of a texture according the the mean, the mean is expressed in terms of the unaligned texture.

$$\bar{\mathbf{t}} = \gamma\mathbf{1}_{n_t} + \beta\mathbf{t} + \epsilon \tag{4.11}$$

γ represents translation, β scaling and ϵ is an error vector describing the remaining difference between the two textures after alignment. To find the optimal parameters $\hat{\gamma}$ and $\hat{\beta}$, the error sum-of-squares $\epsilon^T\epsilon$ is minimized.

$$\epsilon^T\epsilon = (\bar{\mathbf{t}} - \gamma\mathbf{1}_{n_t} - \beta\mathbf{t})^T(\bar{\mathbf{t}} - \gamma\mathbf{1}_{n_t} - \beta\mathbf{t}) \tag{4.12}$$

The optimal parameters are easily found to be

$$\hat{\gamma} = 0 \tag{4.13}$$

$$\hat{\beta} = \frac{\mathbf{t}^T\bar{\mathbf{t}}}{\mathbf{t}^T\mathbf{t}} \tag{4.14}$$

The mean texture can be found using singular value decomposition. The approach used here is the iterative methods of generalized Procrustes analysis for higher dimensions.

1. Center the intensities of all texture vectors using

$$\mathbf{t}_{centered} = \mathbf{t} - (\mathbf{t}^T\mathbf{1}_{n_t}/n_t)\mathbf{1}_{n_t} \tag{4.15}$$

2. Calculate the mean of all k texture vectors

$$\bar{\mathbf{t}} = \frac{1}{n_t}\sum_{i=0}^{k}\mathbf{t}_i \tag{4.16}$$

3. Align the textures using

$$\mathbf{t}_{fit} = \frac{\mathbf{t}^T\bar{\mathbf{t}}}{\mathbf{t}^T\mathbf{t}}\mathbf{t} \tag{4.17}$$

4. Iterate step 2 and 3 until the mean is stable.

4.2.4 Building a Shape Model

Now that the shapes are registered and aligned, they reside in a common *coordinate frame*. This means that the differences in point locations are

solely due to differences in shape. The aligned shapes that are used as input to the model are called *training shapes* or *example shapes*.

A shape model produces linear combinations of the training shapes. This makes it possible to produce shapes not present in the training set. The shape space created by the model spans all the training shapes, all in-between shapes as well as extrapolations, depending on the coefficients. As long as the choice of coefficients are reasonable, plausible, i.e face-like, shapes are generated.

Since there will be as many coefficients of the model as there are examples, it is desirable to change the representation to something more manageable. One effective approach is *Principal Component Analysis* (PCA). Using the vector representation of equation 4.1, the m examples of n_s points will form a point cloud in $3n_s$-dimensional space. With enough shapes in the training set, this point cloud will have a shape depending on the characteristics of the examples. For example, male and female faces will form partitions in space, as well as different ages, face sizes and shapes. The distribution of points will therefore not be gaussian. PCA rotates the current coordinate system, so that the axes point in directions of maximum variation of the point cloud. The new axes are called the *main axes*. The result is that as much of the training set variation as possible is explained by the first axis. The second axis will be orthogonal to the first and will contain as much of the remaining variation as possible. The other axes will have the same properties. If, for example, age cause most of the variation among the examples, the first axis will be along this direction, from young to old. In reality, each axis normally represent a collection of attributes, although the dominant one can easily be identified.

The variation drops rapidly with each axis. When a certain proportion of the total variation is reached, e.g 98%, the rest of the axes can be omitted. This reduces the dimensionality of the model and makes it more manageable. Appendix A describes PCA in detail, however, the results are given here.

Modelling Shape Variation

The main axes of the point cloud are given by the eigenvectors of the covariance matrix. The covariance matrix $\mathbf{\Sigma_s}$ is

$$\mathbf{\Sigma_s} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T \tag{4.18}$$

where $\bar{\mathbf{s}}$ is the mean shape of equation 4.4. The eigenvectors are collected in the $(n_s \times k)$ column matrix $\mathbf{\Phi}_s$ and the corresponding k eigenvalues are denoted $\lambda_{s(i)}$. The resulting shape model is

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{\Phi}_s\mathbf{b}_s \tag{4.19}$$

where $\mathbf{b}_s$ are the shape parameters. These replace the coefficients of the original linear combination model and determine the weights of the new set of axes. It is easily seen that $\mathbf{b}_s = \mathbf{0}$ returns the mean shape. The parameters of an example $\mathbf{s}$ can be found by solving equation 4.19 for $\mathbf{b}_s$.

$$\mathbf{b}_s = \mathbf{\Phi}_s^T(\mathbf{s} - \bar{\mathbf{s}}) \tag{4.20}$$

Suitable Parameter Values

Suitable parameters can be found by using the examples and equation 4.20 to find the standard deviation of each entry of $\mathbf{b}_s$. There is however an easier way. The eigenvalues $\lambda_{s(i)}$ represent the variation of each parameter. Hence, the standard deviation of parameter value i is $\sqrt{\lambda_{s(i)}}$. Suitable values of $\mathbf{b}_s$ are usually in the range of $2 - 3$ standard deviations.

Reducing Model Dimensionality

As mentioned above, with enough examples in the training set, not all the axes of the shape model are necessary to produce a satisfying model. To determine how many parameters that can be omitted, a suitable proportion p of the total variation to be included in the truncated model is chosen. As mentioned, the common choice is $p = 98\%$. Since the eigenvalues represent the variation of the corresponding eigenvalue or axis, they are sorted in descending order. Thus, the t parameters [21] that explain a proportion p of the total variation are found that satisfies

$$p \geq \frac{\sum_{i=1}^{t} \lambda_{s(i)}}{\sum_{i=1}^{k} \lambda_{s(i)}} \tag{4.21}$$

The unused eigenvalues and eigenvectors can be discarded. $\mathbf{\Phi}_s$ is now of size $(n_s \times t)$ which among other things reduces the memory requirements in an implementation.

4.2.5 Building a Texture Model

Building a model that describes the texture variation is done using the same techniques as for the shapes. The vector representation of a texture with color components red, green and blue (r, g, b) is

$$\mathbf{t} = [r_1, \ldots, r_{n_t}, g_1, \ldots, g_{n_t}, b_1, \ldots, b_{n_t}]^T \tag{4.22}$$

The covariance matrix is

$$\mathbf{\Sigma_t} = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{t}_i - \bar{\mathbf{t}})(\mathbf{t}_i - \bar{\mathbf{t}})^T \tag{4.23}$$

and the texture model formulation is

$$\mathbf{t} = \bar{\mathbf{t}} + \mathbf{\Phi}_t \mathbf{b}_t \tag{4.24}$$

Parameters for the examples can be found by

$$\mathbf{b}_t = \mathbf{\Phi}_t^T (\mathbf{t} - \bar{\mathbf{t}}) \tag{4.25}$$

Just as for shapes, the dimensionality of the texture model can be reduced. This is even more useful here, since textures can contain hundreds of thousands of points, ten times the amount of shapes.

Note that the texture model described here is essentially the same as the eigenface model introduced by Turk and Pentland in 1991 [27].

4.2.6 Building an Appearance Model

To generate a complete representation of an object, a set of shape parameters $\mathbf{b}_s$ and a set of texture parameters $\mathbf{b}_t$ could be chosen, and the results could be used to form a single textured object. However, it is desirable to control a single model of both shape and texture with a single set of parameters. Setting $\mathbf{b}_s = \mathbf{b}_t$ is not sufficient since the texture and shape variations may be correlated. To remove this correlation, a third PCA is applied to the concatenated parameter vectors of the textures and shapes as follows.

Using equations 4.20 and 4.25, the parameters of each example are found and concatenated into a single vector.

$$\mathbf{b} = \begin{bmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_t \end{bmatrix} \tag{4.26}$$

The use of the matrix $\mathbf{W}_s$ will be described below. The third PCA is now applied to these vectors giving a third model

$$\mathbf{b} = \mathbf{\Phi c} \tag{4.27}$$

Here, $\mathbf{c}$ denotes the *appearance* parameters controlling both shape and texture. Note that since both the shape and texture parameters have zero mean, $\bar{\mathbf{c}} = \mathbf{0}$.

A more straightforward way of constructing the appearance model is to directly concatenate the shape and texture vectors and perform a single PCA on the resulting correlation matrix to reach the same result as above. Using the method of constructing separate shape and texture models as described here, it is easier to understand the details of an appearance model. The shape and texture models are also interesting to investigate separately.

Balancing shape and texture variation

The concatenated parameter vector $\mathbf{b}$ include a shape with points measured in distance units, and a texture with pixels measured in intensity units. The difference between these can be eliminated by multiplying $\mathbf{b}_s$ by a diagonal matrix $\mathbf{W}_s$ of parameter weights.

In a two-dimensional appearance model, a change in shape also changes the texture vector. Therefore, a somewhat complicated method must be

used to estimate the impact on the texture from varying the shape. In a three-dimensional model, the use of a texture map means that the texture vector is independent of changes in shape. Therefore, it is only necessary to make sure the texture and shape parameters exhibits the same variation. This can be done using the eigenvalues of the separate shape and texture models. The total variation is the sum of all eigenvalues.

$$\sigma_s = \sum_{i=1}^{k} \lambda_{s(i)} \tag{4.28}$$

$$\sigma_t = \sum_{i=1}^{k} \lambda_{t(i)} \tag{4.29}$$

$\mathbf{W}_s$ is then chosen to be $\mathbf{W}_s = r\mathbf{I}$ where $r = \sigma_t/\sigma_s$.

4.3 Model Application

This section describes a few ways of using the 3-D appearance model for image analysis purposes. Of course, there are a myriad of ways to use a general model of human faces; these are the ones implemented here.

4.3.1 Face Synthesis

By altering the appearance parameters $\mathbf{c}$ of equation 4.27, new faces can be synthesized. $\mathbf{c} = \mathbf{0}$ results in the mean shape with the mean texture. To be able to create new faces, the shape $\mathbf{s}$ and the texture $\mathbf{t}$ must be expressed in terms of $\mathbf{c}$.

$$\mathbf{b} = \mathbf{\Phi c} \Longleftrightarrow \tag{4.30}$$

$$\begin{bmatrix} \mathbf{W}_s\mathbf{b}_s \\ \mathbf{b}_t \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_{(s)}\mathbf{c} \\ \mathbf{\Phi}_{(t)}\mathbf{c} \end{bmatrix} \Longleftrightarrow \tag{4.31}$$

$$\begin{bmatrix} \mathbf{W}_s\mathbf{\Phi}_s^T(\mathbf{s} - \bar{\mathbf{s}}) \\ \mathbf{\Phi}_t^T(\mathbf{t} - \bar{\mathbf{t}}) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_{(s)}\mathbf{c} \\ \mathbf{\Phi}_{(t)}\mathbf{c} \end{bmatrix} \Longleftrightarrow \tag{4.32}$$

$$\begin{bmatrix} \mathbf{s} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{s}} + \mathbf{\Phi}_s\mathbf{W}_s^{-1}\mathbf{\Phi}_{(s)}\mathbf{c} \\ \bar{\mathbf{t}} + \mathbf{\Phi}_t\mathbf{\Phi}_{(t)}\mathbf{c} \end{bmatrix} \tag{4.33}$$

Note that $\mathbf{\Phi}_{(s)}$ is the part of $\mathbf{\Phi}$ occupied by the shape parameters and $\mathbf{\Phi}_{(t)}$ is the corresponding part for the texture parameters.

As an alternative, new faces can be synthesized using all three models by first calculating $\mathbf{b}$ from $\mathbf{c}$ using equation 4.27. From $\mathbf{b}$, $\mathbf{b}_t$ is found directly and $\mathbf{b}_s = \mathbf{W}^{-1}\mathbf{b}_{(s)}$. Using equations 4.19 and 4.24, $\mathbf{s}$ and $\mathbf{t}$ can finally be calculated. This is the method used in the implementation.

As before, suitable parameter values are in the range $\pm 3\sqrt{\lambda_i}$, where λ_i are the eigenvalues of the combined model.

4.3.2 Face Segmentation in 2-D Images

The most obvious use of an appearance model of human faces is face detection and segmentation. By using a snapshot of the current view of the 3-D model, a 2-D image representation can be created. Segmentation algorithms using a 3-D model are based on a comparison between the image to be interpreted and the snapshot. Using information from this comparison, the model is updated, a new snapshot is created and the result is used to further improve the fit. Convergence is declared when the difference between the underlying image and the model snapshot has reached a (local) minimum. Because of the high complexity of a 3-D model, the algorithm requires careful initialization to converge to the global minimum.

The common method of deciding how to update the model, using the information from the image comparison, is the *active* appearance algorithm. As described in chapter 2, the method is based on prior knowledge on how to alter the model to improve the fit. The method used here does not make use of any prior knowledge. Instead it is based on a general minimization algorithm. Most high-dimensional optimization algorithms are *gradient* based, meaning that the minimization path is along descent directions. The method used here is called an *amoeba* minimizer. Following a certain scheme, the algorithm feels around in the parameter space looking for directions where a better fit can be found.

The objective function G is based on the norm of the difference between the image to be interpreted I and the 2-D representation of the model M_{2D}. The variables of G are the appearance parameters $\mathbf{c}$, the position of the model in 3-D space $\mathbf{t} = [t_x, t_y, t_z]^T$ and the uniform model scale β.

$$G(\mathbf{c}^T, \mathbf{t}^T, \beta) = \|I - M_{2D}\|^2 \tag{4.34}$$

Note that t_z and β has a similar function if a perspective projection of the model is used. The parameters are just affecting the model, the image I is static throughout the optimization process.

The 2-D target image often contain significant background information while the model image is empty outside the model itself. To make the differentiation work, only the area of the target image that is covered by the model is included. Differences in brightness and color balance are taken care of by normalizing the images before differentiation.

More parameters could be added to improve the specificity of the model such as lighting direction, projection parameters etc.

4.3.3 Automatic Registration of New Face Scans

The appearance model can synthesize new 3-D faces as described in section 4.3.1. If the model is general enough, it will be able to approximate any unseen 3-D face. This can be used to automatically register new face scans. These registrations can then be included in the model to further generalize it.

In section 4.2.1, the original semi-automatic registration algorithm is explained. A thin-plate spline warp is used to approximate the new shape using the template shape. The warp is defined by the manually defined sparse sets of source and target landmarks. Using the finished appearance model, the new face can be approximated by the model instead. This can be done automatically (possibly with manual initialization) by an algorithm proposed by Hutton [26].

The algorithm is an iterative procedure including two basic steps.

The ICP step: *Iterative Closest Point* (ICP) [3] is an iterative process that optimally aligns a shape to another using either a rigid-body transform or a similarity transform. ICP is described in detail in appendix C. Here, the unregistered shape is aligned with the model shape.

The model refinement step: The novel shape is registered using the point-to-surface technique described in 4.2.1. Using equation 4.20, the parameters that best approximate the registration are found. The parameters are clamped to the range ± 3 standard deviations. The result is used to update the model using equation 4.19.

Since the model changes in the model refinement step, the new shape needs to be aligned again with ICP. The new alignment is then further refined until convergence. When the algorithm finishes, the current registration is used as the final registration result.

Details

The ICP algorithm [3] is very robust, and usually it is only necessary to match the scans by their centroids to make ICP converge correctly. On some occasions manual initialization may be required.

Since ICP works by using the same point-to-surface method as the registration algorithm, the new face scan must have an extent greater or equal to the model. This is also a prerequisite for producing a satisfying registration.

ICP translates, rotates and scales the model according to the new face scan. Since the extent of the new shape is greater or equal to the extent of the model, ICP can only align the model to the new shape, not the other way around. If the larger of the two is used as the source, some points will not have a correctly corresponding point on the target surface. Transformations of the model should, however, be avoided, as this brings it out of shape-space. Outside shape-space, the model refinement step does not work. The transformation found above is therefore inverted, and applied to the unregistered shape instead. This aligns the new shape with the model.

To measure the converge of the algorithm, the squared *Procrustes distance* between the registrations of the current and former iterations is used. Equation D.2 states the expression of this. When the distance drops below a certain threshold, the registered shape is stable and cannot be improved any further.

Chapter 5

Implementation

This chapter presents how the methods of the previous chapter are implemented, as well as listing the software used.

5.1 Software

5.1.1 VTK

The majority of the project was implemented using the *Visualization Toolkit*, also known as VTK. VTK is an open-source project with the aim to create a powerful and versatile toolkit for visualization in two, three and four dimensions. Source code, binaries and documentation can be found at the VTK web site:

`www.vtk.org`

The VTK project is administered by Kitware inc., which also offers other computer vision packages. Visit `www.kitware.com` for more information.

VTK is implemented in C++. A binary distribution exist, but to get the full distribution, the source code must be compiled. All VTK classes comes with a set of wrapper classes so that VTK can be used through Tcl/Tk, Python and Java. In this project, as much as possible has been implemented using Tcl/Tk because of the rapid development and the ease

of creating graphical user interfaces. Complicated and high performance methods has been implemented using Microsoft Visual C++ 6.0.

5.1.2 Miscellaneous Software

Code-Genie is a small, simple yet powerful code editor. It is free to try and can be found at `www.code-genie.com`.

The Gimp is a free image-editing program similar to the popular commercial program *Photoshop* from Adobe. The program was originally written for UNIX computers, but the version used here is the port for Windows computers. The Gimp can be found at `www.gimp.org`.

WinEdt is a text editor for creating LaTeX documents, such as this thesis. WinEdt is free to try and can be found at `www.winedt.com`.

MiKTeX creates DVI, Postscript and PDF documents from TeX files on Windows computers. MiKTeX is required for WinEdt to run properly. MiKTeX can be downloaded from `www.miktex.org`.

Cygwin provides a suite of programs to give a Windows computer the command-line power of a UNIX computer. It also provides an implementation of XFree86 which is an X-server. This makes it possible to graphically connect to UNIX computers. Cygwin can be found at `www.cygwin.com`.

Xfig is a program for drawing vectorial images for documents such as this one. See `www.xfig.org` for more information.

Emacs is an advanced text editor with support for most programming languages. Emacs is free, exists both for UNIX and Windows computers, and can be downloaded from `www.gnu.org/software/emacs`.

5.2 Registration Implementation

Figure 5.1 shows the flow of programs used to create an appearance model from a set of unregistered VRML files. The following sections will describe these programs in detail.

5.2.1 File Format Conversion

The VRML files created by the camera software must be converted to a format which can be used by VTK. The toolkit contains functionality for

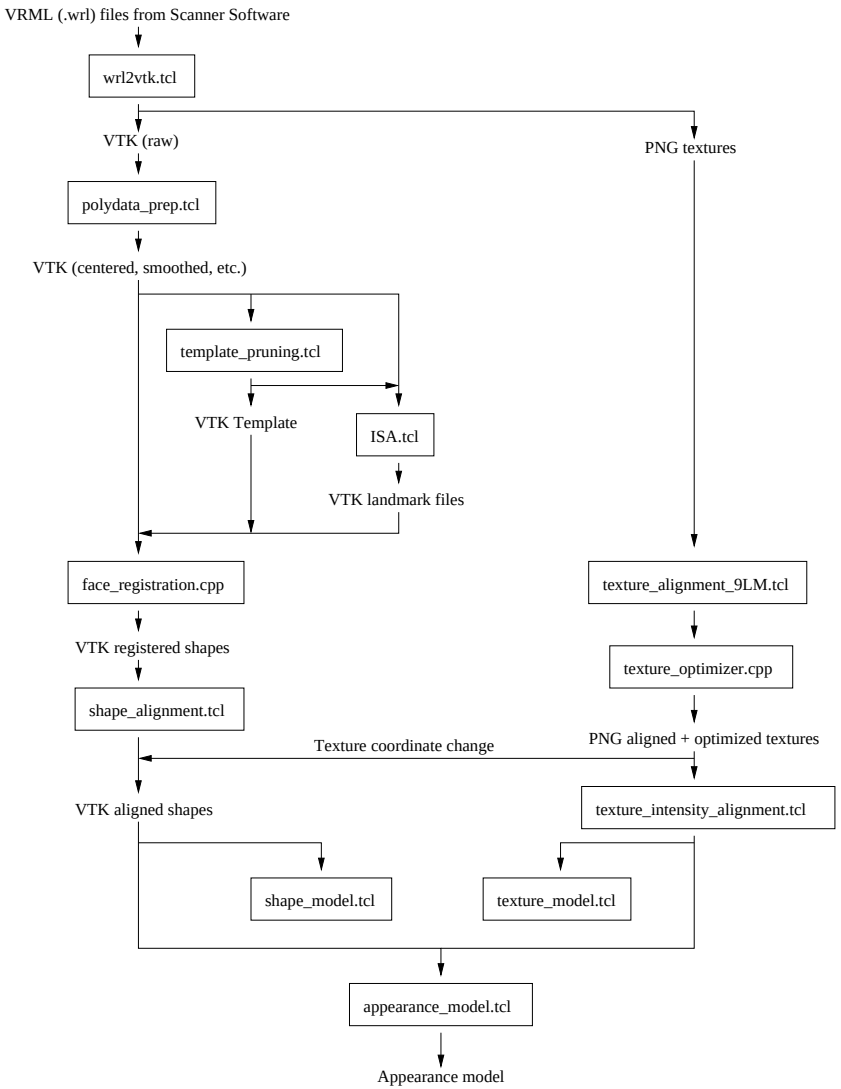


Figure 5.1: The range of implemented programs used to create an appearance model from a set of VRML files.

reading many different formats, but the native VTK format is similar to VRML and therefore suitable. Rendered below is an example file with data identical to the VRML file of section 4.1.2.

```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET POLYDATA
POINTS 3 float
0.0 0.0 0.0 10.0 0.0 0.0 5.0 5.0 0.0

POLYGONS 1 3
3 0 1 2

CELL_DATA 1
POINT_DATA 3
TEXTURE_COORDINATES textureCoords 2 float
0.120453 0.114399 0.003498 0.001515 0.357738 0.424964
```

The file contains 3 points defining a triangle in the xy-plane. "`POLYGONS 1 3`" means there is one triangle and the total number of vertex references is three. A polygon is defined by first specifying the number of references (3) and then specifying the references (0, 1, 2). The texture coordinates are defined as pairs of floating point numbers.

Since the VRML and the VTK formats are similar, the file conversion is straightforward to implement. Using the powerful regular expression functions of Tcl, a robust implementation is created. The program converts any VRML file created by the camera software to the VTK format, but it is not a general VRML to VTK converter, since there are many features of VRML that are not handled.

The VRML file contains the texture map as ASCII data while VTK lacks this feature. Instead, the texture map is read from the VRML file and saved as a regular image for use with VTK. The image format is PNG which is a non-degenerative format with high compression.

5.2.2 Preparing Files for Registration

The camera measures actual distances from the CCD. The faces of the training data is therefore placed far away from the origin. Since it is more

natural to work with objects centered at the origin the objects are translated accordingly. The true centroid is not calculated, instead a simpler approach is chosen where the center of the object's bounding box is used.

The scanning process sometimes registers hair and pieces of clothing which results in small "islands" of polygons around the model. These are removed by analyzing the *connectivity* of the model. The largest connected surface is found and saved, while all disconnected parts are discarded.

As explained in section 4.1.3, the surfaces resulting from the scanning process are rather rough. Therefore, they are smoothed using an algorithm that adjusts the point coordinates according to a windowed sinc function interpolation kernel [11]. The operation results in a relaxation of the mesh, making the polygons better shaped and the vertices more evenly distributed.

5.2.3 Template Preparation

The 3-D annotation software has functionality for defining and saving planes in three dimensions. This can be used to define a set of boundaries which in turn can be used for pruning the template. By examining all the objects of database using the annotation software, a suitable template area is found. Three planes are used to prune the model, one for removing parts of the neck, one for removing the top of the head and one for removing the back of the head, including the ears.

The template defines the polygon references and texture coordinates of all the other objects during the registration process. It is therefore crucial for the quality of the final model that these are of the best possible quality. The polygon references are acceptable and need no alteration. The texture coordinates, on the other hand, need to be improved. The top left corner of all texture maps is empty. The mapping $(s,t) = (0,0)$ is therefore incorrect, yet several entries are mapped this way. Since the neighboring texture coordinates with $(s,t) \neq (0,0)$ are assumed to be correct, the missing mappings can be filled in using linear interpolation. Let s_0 denote the s-value to interpolate from and let s_n be the target value. In between, there are $n-1$ faulty values to fill in. The expression for the value of slot k is

$$f(k) = s_0 + \frac{k}{n}(s_n - s_0), \quad 1 \leq k < n \tag{5.1}$$

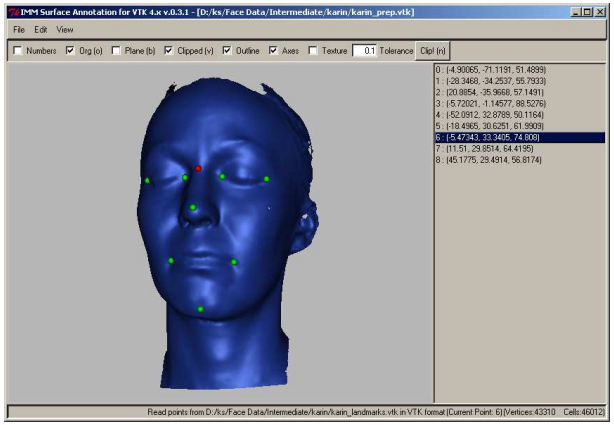


Figure 5.2: The landmarking software showing a completely annotated face.

The same formula applies for t-values. The result is a complete representation of texture coordinates.

5.2.4 Improving the Annotation Software

Even though a sparse set of only nine landmarks are manually defined on each object, this is a time-consuming process with a large database. The 3-D annotation software (see section 4.2.1) is a great aid. The software was originally written for annotating 3-D models of human ear canals, and therefore it lacks functionality for textured objects. Hence, the program has been improved to include this. A function for changing the tolerance when selecting vertices has also been added. This makes the program capable of annotating objects of different scale with greater accuracy.

Adding texture to 3-D objects can make the annotation process easier since the user is guided by both shape and texture when placing landmarks. Some landmarks, like the corner of an eye, is defined more in terms of the texture than of the shape. Other landmarks, like the tip of the chin, is easier to define using the shape.

The landmarks are saved in regular VTK files containing point data only.

Figure 5.2 shows the software with an annotated face.

5.2.5 Shape Registration

The registration algorithm is straightforward to implement since the complicated concepts of the thin-plate spline warp and finding the closest point of a surface are supported by VTK.

As mentioned earlier, the polygon definitions of the registered objects are discarded and replaced by the references of the template. In the final model, the texture coordinates of the template are also used for all models, but to be able to view individual registered faces with their original texture, new texture coordinates are calculated. When the closest point on a polygon is found, new texture coordinates for that point is calculated as follows.

1. Express the new position as a linear combination of the positions of all vertices included in the polygon. Denote the new position $\mathbf{x}'$ and the n vertices of the polygon $\mathbf{x}_i$. The expression is then

$$\mathbf{x}' = w_1\mathbf{x}_1 + \ldots + w_n\mathbf{x}_n \tag{5.2}$$

   This is an underdetermined system of equations, but unique values of w_i can be found using *barycentric coordinates* [1].
2. Use the weights w_i to find the new texture coordinates $\mathbf{t}'$ by

$$\mathbf{t}' = w_1\mathbf{t}_1 + \ldots + w_n\mathbf{t}_n \tag{5.3}$$

5.3 Alignment and Optimization

5.3.1 Shape Alignment

VTK also provides functionality for performing a Procrustes analysis on the vertices of polygonal objects. The alignment is very quick and works well with the extremely large amount of points used here. Internally the Procrustes analysis is performed using a quaternion-based landmark transform, see appendix C.

The aligned objects are saved so that the modelling implementations described below have easy access to prepared data.

5.3.2 Texture Alignment

The texture alignment implementation is a surprisingly complicated process. As described in section 4.2.3, the alignment is based on landmarks defined by the 3-D landmarks which are transformed into 2-D landmarks by means of the texture coordinates. Since the sparse landmarks are defined as a set of separate points, the corresponding points of the model must be found. Therefore, the implementation uses the 3-D data, the texture and the landmark points of each model.

The point correspondence landmark-to-model is found using a point-to-point locator supplied by VTK. The landmark transform is defined by a thin-plate spline warp just as for 3-D surfaces. The only difference is that the z-coordinate is kept constant.

The next step is to prune the textures to remove all the unused textural information and to make sure all textures have uniform extent. The outline of the 3-D model is transformed to a 3-D loop using the texture coordinates. Before this is done, the 3-D points are sorted so that the sequence of points describe the loop one-by-one until the loop is closed. When this is done the points are used to define a polygon so that the loop is transformed into a filled area. The 2-D coordinates of the points are then found and the polygon references are copied into the new 2-D polygon. This polygon is then used to build an image stencil which can be used to mask parts of an image. The stencil has to be reversed so that the correct parts of the images are masked out. Using this stencil all the textures are trimmed and saved.

After alignment, the textures have a uniform representation. The texture coordinates of the template will correctly match the polygons of any of the registered shapes to any of the textures. As a side effect, this makes it possible to combine any shape with any texture.

5.3.3 Optimizing Texture Size

The textures are 800 by 400 pixels large. Three bytes are used to encode the color information of each pixel, resulting in an image of $800 \cdot 400 \cdot 3 =$ 0.96MB. The actual image information of the aligned textures occupy an area of 371 by 279 pixels, just 11% of the total size. If the image size is changed to this, considerable savings in computational power can be

made. This is easily done, but the common texture coordinates have to be transformed since the image effectively has been translated and scaled. Let x_{min}, y_{min}, x_{max} and y_{max} denote the boundaries of the target area. Let ΔX and ΔY represent the width and height of the original image. The new texture coordinates can be expressed as a function of the old by

$$(s', t') = (\frac{s\Delta X - x_{min}}{x_{max} - x_{min}}, \frac{t\Delta Y - y_{min}}{y_{max} - y_{min}}) \tag{5.4}$$

5.3.4 Intensity Alignment

The intensity alignment algorithm described in section 4.2.3 is implemented in a VTK extension called `vtkImageIntensityAlignmentFilter`. This will eventually be submitted to possibly be included in the VTK distribution. Before this can be done, the iterative approach should be changed to the analytical alternative since this is magnitudes faster. Also, the filter works on a copy of the data. If the implementation would be able to work on the data directly, the memory requirements would be halved.

5.4 Model Implementation

The shape, texture and appearance models are implemented in Tcl/Tk. All three programs contain a graphical user interface with the current shape visible in the main window. The interface also contains scales for the first ten modes of variation which can be altered interactively. The shape and appearance models can be rotated, panned and scaled by the user. The programs essentially read the aligned data, performs the necessary decomposition (PCA) operations and displays the results.

5.4.1 The Shape Model

Using the aligned shapes, the shape model implementation is uncomplicated. VTK includes a filter called `vtkPCAAnalysisFilter` for performing PCA on 3-D points.

Figure 5.3 shows the shape model software in action.

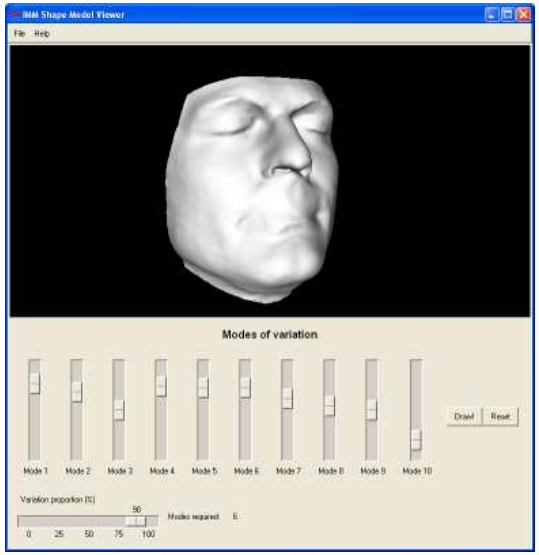


Figure 5.3: The shape model viewing software.

5.4.2 The Texture Model

The texture model requires more effort to implement since VTK lacks functionality for performing PCA on a set of images. This ability was added in a filter called `vtkImagePCAFilter`. Just as the intensity alignment filter described above, this will be contributed to the VTK distribution. With this available, the implementation is identical to that of the shape model.

As mentioned above, a similar viewing program as for the shape model has been implemented. This can be seen in figure 5.4.

5.4.3 The Appearance Model

The appearance model combines the functionality of the shape and the texture model, see section 4.2.6. To perform the final PCA on the resulting parameter vectors, neither `vtkPCAAnalysisFilter` nor `vtkImagePCAFilter` can be used. Instead, a third PCA variant, `vtkArrayPCAFilter` had to be implemented. The usefulness of this outside appearance modelling is

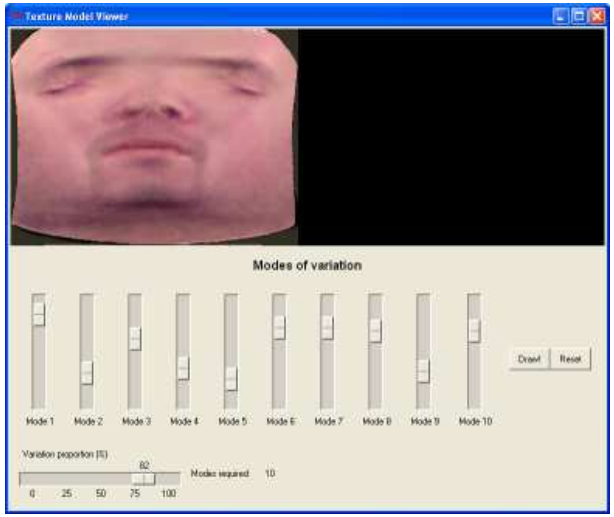


Figure 5.4: The texture model viewing software.

questionable, and this file will probably not be contributed to the VTK distribution.

Figure 5.5 shows a snapshot of the appearance model viewing program.

5.5 Implementation of Applications

5.5.1 Fitting to 2D Images

The implementation of the face image segmentation algorithm is very basic and does not provide a user friendly interface. The only user interaction that occurs is the manual initialization of the 3-D model according to the 2-D image. Preferably, the model and the image would be displayed in the same window to facilitate this. Unfortunately, the computer used for the development of this program is equipped with an ATI graphics board. Hardware from this manufacturer contains a bug that makes such overlay operations impossible. The program therefore exists in two versions, one for ATI and one for non-ATI graphics boards.

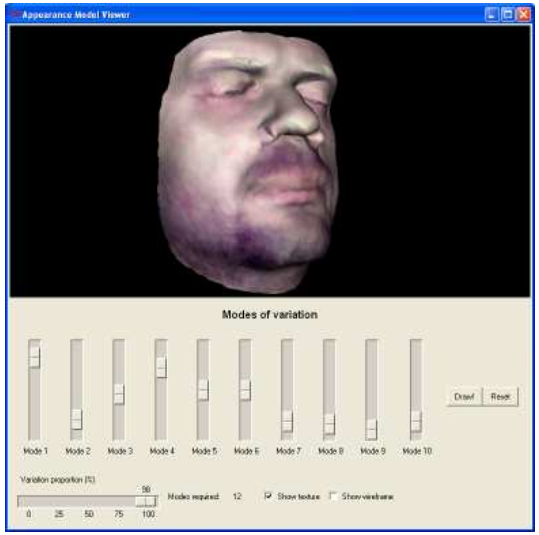


Figure 5.5: The appearance model software. The scales control the ten first modes of variation.

VTK supports methods for creating 2-D snapshots from 3-D models and the other necessary operations. A new class, `vtkImageStatistics`, was created which eventually will contain all sorts of methods for acquiring statistical information from images.

5.5.2 Automatic Face Registration

The automatic registration software has a menu-driven graphical user interface. The model used for the fitting process is visible in the main window together with the shape to register. Three sub windows show the original shape, the current registration and the best model approximation of the current registration. The program contains functions for loading an unregistered shape and for saving the registered shape. The registration process can either be followed one iteration at a time, or be performed all at once.

The registration algorithm itself cannot be implemented in Tcl/Tk. Therefore, a helper VTK extension class, `vtkAutoRegEngine`, had to be implemented.

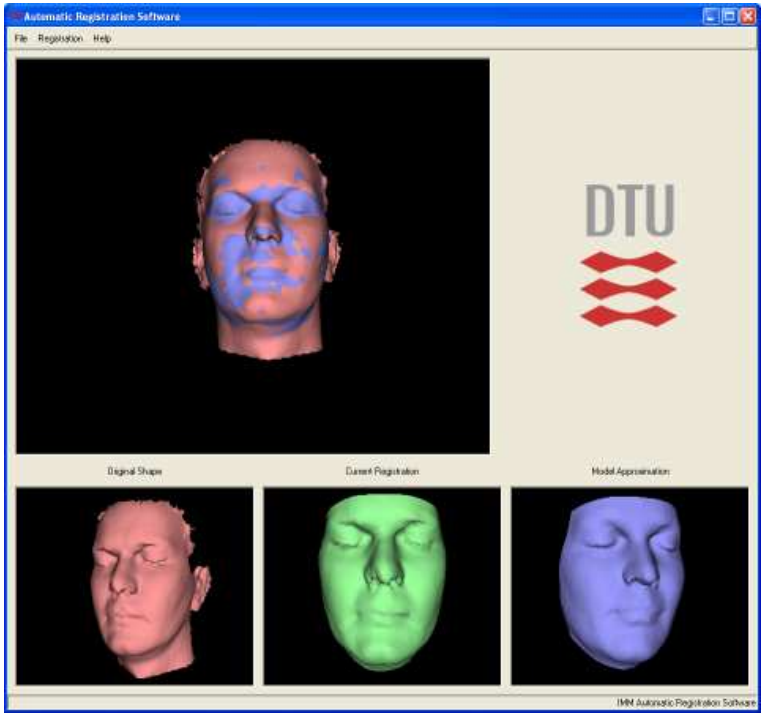


Figure 5.6: The automatic registration software. The top window shows how the new shape is fitted to the model.

Figure 5.6 shows the output of the automatic registration software.

Chapter 6

Results

This chapter presents the results of the methods and algorithms used to create models and applications.

6.1 Face Data

As described in section 4.1.3, the quality of the face scans is poor. The main imperfections are

- a rough shape surface resulting from the difficulty for the people being scanned to maintain the same pose during all three scans,
- no texture projection can give a perfect representation, the cylindrical projection used here gives poor results in areas perpendicular to the cylinder surface,
- incorrect texture color balance, possibly a result of improper lighting,
- missing texture-coordinate mappings showing as the mapping $(s, t) = (0, 0)$.

Appendix E shows the face database where these imperfections are more of less visible.

Figure 6.1 shows the result of a single scan of a smiling person with open eyes. This demonstrates the problems with registering eyes and expressions.

Figure 6.1: The results of a single scan of a smiling person. Notice the poor representation of eyes, teeth and hair.

6.2 Model Creation

The quality of the models are dependent on the quality of the data and the registration algorithm. The results of the latter are listed here.

6.2.1 Annotation

After modifying the landmarking software, the program is ideal for annotating most textured objects. The ability to set the tolerance when picking vertices in the mesh makes it possible to work with objects of any scale. When placing a landmark in a position defined in terms of the shape, the texture can be distracting. It is therefore possible to turn the texture on and off.

6.2.2 Registration

The semi-automatic registration algorithm works well, but requires that the template and surface to be registered are rather similar. Any scale and aspect ratio differences are handled by the thin-plate spline warp, but large shape variation results in uneven and incorrect registration. The problem becomes apparent when the template surface has high curvature and the

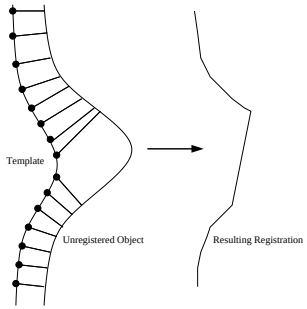


Figure 6.2: The registration of two surfaces with differing curvature. Notice the difference between the original and the registered shape.

novel surface curvature is low. The resulting surface will have an uneven point distribution and the high curvature parts will be cut off. This is depicted in figure 6.2. For the human face, problems occur mainly around the nose and eyebrows. A method for regularizing correspondences found through methods such as this one is described in [17].

6.2.3 Alignment

The shape alignment is fast, easy to use and accurate thanks to the Procrustes analysis methods of VTK.

The texture alignment works surprisingly well. Many other landmark configurations to drive the thin-plate spline warp were tested. These either gave an uneven or unsuitable distribution of landmarks, resulting in inexact alignment of important texture areas. Too many landmarks resulted in an overly bent and twisted texture.

Intensity Alignment

The intensity alignment of the geometrically aligned textures is immaculate, although a bit slow. By using the analytical approach to finding the mean intensity described in appendix D, the algorithm would be two to four times faster. Figure 6.3 shows three unaligned texture examples and figure 6.4 shows the corresponding aligned textures.

Figure 6.3: Cropped textures with unaligned global intensity and color balance.

Figure 6.4: The same textures as in figure 6.3, but with aligned global intensity and color balance.

6.2.4 The Shape Model

The implementation of the shape model and the accompanying face synthesis program work as expected.

Figure 6.5, 6.6 and 6.7 show the first three modes of shape variation.

6.2.5 The Texture Model

The texture model software was created principally for testing the implementation of the image PCA filter for VTK, and the impact on the texture model from the intensity alignment.

Figure 6.8, 6.9 and 6.10 show the first three modes of texture variation.

6.2.6 The Appearance Model

Figure 6.11, 6.12 and 6.13 show the first three modes of variation. All three modes has a face size component, since the model is of size-and-shape type.

Furthermore, the first mode seem to model gender, while the second and third mode model aspect ratio and amount of beard.

The six first modes each describe 10% of the total model variation. This is an effect of the low number of faces in the training set. No apparent clustering of the face vectors can occur for such a small number of examples, instead they form a roughly gaussian distribution. With more faces in the database, the PCA transform would construct a basis with clearly descending modes of variation.

6.3 Applications

The applications of the model were implemented during a short period of time, and the results might suffer somewhat from this.

6.3.1 Face Segmentation

The results of the face segmentation algorithm are disappointing. Even in the easiest case imaginable, fitting the model to an image of the model, the matching fails in most cases. Of course, segmenting a regular photo of an unseen face, or a person in the database, also fails. In figure 6.14, the model has converged to a solution some distance from the correct minimum. The model was initialized to a seemingly accurate position, but wandered off to an incorrect position during the optimization process.

6.3.2 Automatic Registration

The automatic registration software works satisfyingly. The algorithm initially converges quickly and monotonically, less than ten iterations are usually sufficient to obtain a satisfying registration. The convergence rate drops with more iterations, and the process is no longer monotonic. However, the trend is still towards a better fit.

Because of the robustness of the ICP algorithm, the initial position and orientation of the template seldom has to be changed manually for the algorithm to converge. The method is therefore fully automatic, as desired.

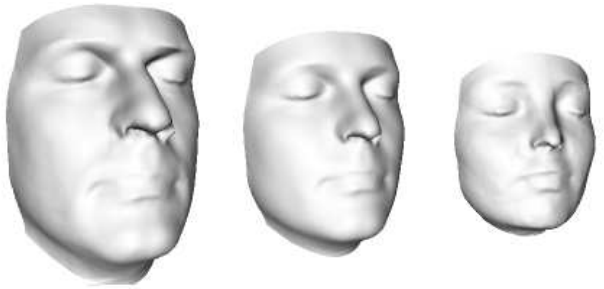


Figure 6.5: The first mode of shape variation.

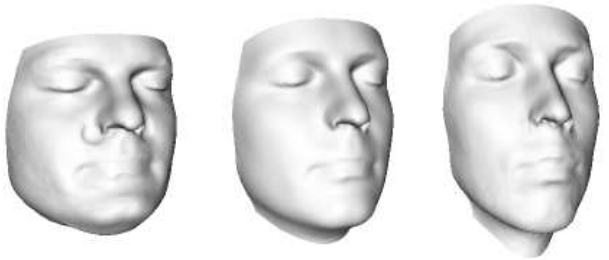


Figure 6.6: The second mode of shape variation.

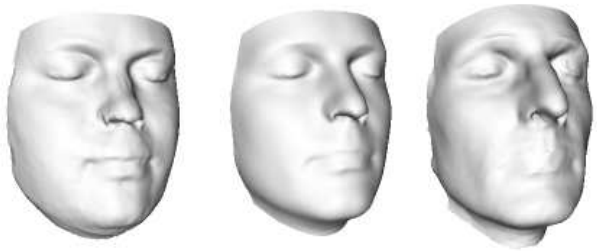


Figure 6.7: The third mode of shape variation.

Figure 6.8: The first mode of texture variation.

Figure 6.9: The second mode of texture variation.

Figure 6.10: The third mode of texture variation.

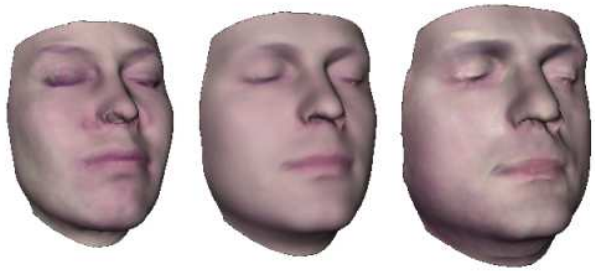


Figure 6.11: The first mode of combined appearance variation.

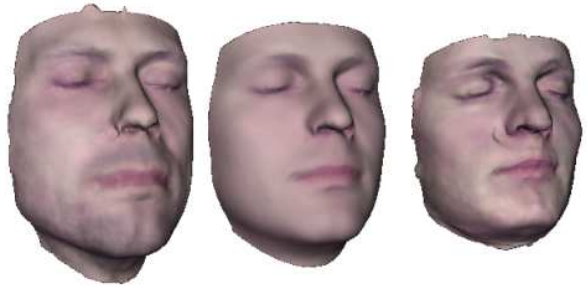


Figure 6.12: The second mode of combined appearance variation.

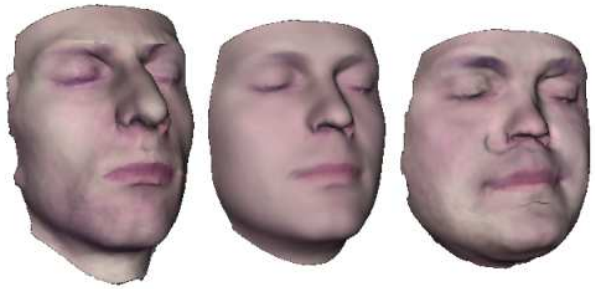


Figure 6.13: The third mode of combined appearance variation.

Figure 6.14: The model incorrectly fitted to a 2-D image.

Chapter 7

Summary and Conclusions

This thesis has described methods for building three-dimensional models of shape, texture and appearance of human faces.

A data set of 24 faces was used. These were acquired using a 3-D scanner at the School of Dentistry, University of Copenhagen. Each complete scan required three sub-scans that were merged using the software provided by scanner manufacturer. It proved to be hard to get a complete and well-defined representation of a whole face. This was due to the difficulty for the people being scanned to maintain the exact same pose during all three shots. The imperfections show as rough surface patches, missing texture mappings and incomplete polygon coverage. The color balance of the texture was also unsatisfying, possibly due to incorrect color temperature of the lighting used. Despite this, the resulting database is good enough for many image analysis applications.

Each face scan consists of a set of 3-D points, polygonal point references, a texture and texture coordinates. Every shape has a different point ordering and extent. To be able to analyze the data statistically, the representation of the shapes must be unified. This *registration* process, suggested by Hutton *et al.* [26], uses nine manually defined landmarks to automatically register thousands of points on each shape. The point ordering, polygonal representation, texture coordinates and extent is determined by one of the examples, called the *template shape*. Using a thin-plate spline warp, the template is transformed, using the nine manually defined landmarks, so

that it takes on a shape similar to the new shape. The extent and point ordering is then unified by finding the closest point on the new shape's surface from each template point. The results of the registration are satisfying, but some artifacts occur which calls for a replacement of the point-to-surface closest point operation.

The registered shapes were then *aligned* using a partial Procrustes analysis. This translates and rotates the objects optimally with respect to their mutual mean. The shapes are thereby brought into *shape-space*. The textures were also aligned so that size, location and shape of each texture became identical. This was done using the nine manual landmarks and a thin-plate spline warp.

Using principal component analysis (PCA), two separate models of shape and texture were built from the aligned data. These were combined into a model of appearance, using a third PCA. Programs for viewing the models and interactively change the modes of variation were implemented using the Visualization Toolkit (VTK), Tcl/Tk and C++. The quality of the synthesized faces is better than the input data, and produce near photo-realistic results. This shows that the methods used, despite drawbacks, are good enough for the purpose of face synthesis.

Two other applications of the models were implemented. As a first attempt of using the appearance model for 2-D image segmentation, the model was matched to information in images using a simple optimization method. The algorithm seldom converged to a correct solution. Using a gradient based method instead should improve the results. The shape model was used in an algorithm for automatic registration of new face scans. The quality of the resulting registrations were comparable to the ones created using the semi-automatic method described above.

In conclusion, building shape and appearance models in three-dimensions comes with a high amount of overhead. Everything from data acquisition to registration requires more work, complicated algorithms and powerful computers. 2-D models are easier to create, requires less computer power and can be almost as general as a 3-D model [24]. However, as computers get faster, equipment for acquiring 3-D data becomes reasonably priced and new research emerges, this type of model is expected to be more common. This project has provided the author, and possibly the reader, with a detailed introduction to this exciting field of research.

Chapter 8

Future Work

This chapter presents a few ideas for directions of future work. Some ideas have the purpose of improving the existing models while others are suggestions for new extensions and applications.

8.1 Increased Quality and Diversity of Face Scans

The face data is the foundation of the methods in this thesis. As described in section 4.1.3, the data is far from perfect. Although these imperfections are dealt with using smoothing, interpolation, intensity alignment etc., it would be preferable to have data of sufficiently high quality from the start. Even though a lot of time has been put into finding an optimal scanning process, there may still be changes that improves the quality.

A major limitation to the database is that all faces are scanned with the eyes closed. This makes it difficult to fit the model to most 2-D images, since these normally depict faces with open eyes. While the camera has trouble registering eyes, this improvement might be hard to accomplish.

The database only covers people with natural expressions. To make the model more general, faces with expressions ranging from a frown to a smile should be included. The difficulty here is for the people being scanned to

maintain the exact same expression during three scans. Experiments show that even after practicing a few times, it is hard to keep a static expression for several minutes. One solution to this problem could be to scan each expression only once, from straight ahead. The extent of the data will be insufficient, but the necessary area of the face will be covered. This can then be pasted onto the complete face with the natural expression to yield a complete face with another expression. Data for this exist in the database.

The database should also be extended, preferably including a greater diversity of people. The age range should be broadened and an even distribution of race and gender should be fulfilled.

The conclusion drawn from the discussion above is that the camera used is not ideal from scanning human faces. Scanners used for this purpose often use *structured light* instead of lasers to capture the shape of an object. These are also magnitudes faster and cover more of a face with a single scan. However, at the time of writing, this type of equipment is very expensive, around 50 000 USD.

8.2 Assessment of an Alternative Registration Algorithm

Several algorithms exist for registering 3-D surfaces. Of particular interest is a method composed by R. R. Paulsen and K. Hilger [17] which can be used to regularize registrations of the type described in this thesis. Using Markov random fields, a set of constraints are imposed on the landmarks which makes them more regularly interspaced and matches local curvature of the template and the new shape.

A suitable task would be to compare this registration algorithm to the one used here, and assess the differences of the results.

8.3 Improved Registration by Iterative Model Refinement

The registration method used in this thesis works satisfyingly and requires fairly little work, since only nine landmarks have to be defined manually.

However, if a database of hundreds of objects are to be registered, it would be desirable to automate the process. The proposed algorithm can be used to drive a fully automatic registration method by "bootstrapping". This means that a subset is registered semi-automatically and the result of these registrations are used to automatically register the rest.

The proposed algorithm is as follows

1. Annotate a small subset of objects with sparse sets of landmarks.
2. Register these examples using the method described in section 4.2.1.
3. Using the registered shapes, build a shape model described by equation 4.19.
4. The remaining unregistered shapes are registered using the automated registration process described in section 4.3.3.
5. All shapes are now registered, however the shape model used is incomplete because low amount of examples included. The shape model is therefore rebuilt using all examples.
6. The old registrations are now discarded and the new shape model is used to re-register all examples.
7. The improved registration are once again used to build an improved shape model which is used to register all shapes. This process is iterated until the registrations are stable.

Since the registration is dependent on quality of the model, and since the model will be increasingly accurate, the registrations are assumed to converge to a correct solution.

8.4 Improving the Image Matching Algorithm

The algorithm and implementation that matches the appearance model to 2-D images is only briefly tested on a small set of images, with disappointing results. The method includes many parameters, and better values of these could improve the results significantly. Also, the amoeba optimization algorithm used should be changed to one which is less prone to wander off from the closest minimum. Any gradient-based method should suffice. More parameters could also be included, such as camera projection and light parameters.

8.5 Implementing a 3-D Active Appearance Model

One of the most interesting opportunities for extending the work presented in this thesis is an implementation of the active appearance model for 3-D surface data.

The idea of the active appearance model is to learn how to correct the model parameters according to the image residuals between the model and the new image. This method should also be appropriate with a 3-D model. In each iteration, an image of the current view of the model is created and used for the comparison with the new image. The model is then moved and deformed to improve the fit.

The 2-D appearance model is positioned in the frame of the new image using a set of pose parameters. These include translation, an in-plane rotation and scaling. In three dimensions, more parameters are required, for example to describe a full 3-D rotation. This should, however, not be a problem to the method.

To learn to correct the model parameters two methods can be used, either a method based on principal component regression or another, gradient-based method. Both should be tested to determine which is most appropriate for the three-dimensional case.

The increased dimensionality of the model and the parameters are expected to make the model less robust, but the method should still be useful.

Appendix A

Principal Component Analysis

Principal Component Analysis (PCA) [15] involves a change of basis for data to a basis with the following properties:

- The first axis is chosen so that the variance of the projected data along this axis is maximized. The second axis is chosen so that the remaining variance is maximized along that axis, and so on for all axes.
- All axes are orthogonal.
- Each new variable (axis) is a linear combination of the original variables (axes).
- The transformation does not contain scaling.

To illustrate these ideas, a simple example with two-dimensional geometrical data will be given. This can then be generalized to higher dimensions. Figure A.1 shows a set of points on a plane, and table A.1 lists the coordinates.

It is immediately seen that there is strong correlation between the points, they seem to roughly fulfill the equation $x_1 = x_2$. Let's assume that the greatest variation indeed can be found along this line and choose $x_1 = x_2$ as the first new axis. The second and last axis then becomes $x_1 = -x_2$ as this is the only choice for orthogonal axes. Projecting the points onto these

new axes by

$$\tilde{x}_1 = cos\frac{\pi}{4}x_1 + sin\frac{\pi}{4}x_2 = \frac{1}{\sqrt{2}}x_1 + \frac{1}{\sqrt{2}}x_2 \quad (A.1)$$

$$\tilde{x}_2 = -sin\frac{\pi}{4}x_1 + cos\frac{\pi}{4}x_2 = -\frac{1}{\sqrt{2}}x_1 + \frac{1}{\sqrt{2}}x_2 \quad (A.2)$$

gives the the updated data in table A.2.

Almost all variance is now along $\tilde{x}_1$, just as expected. This leads to the suspicion that there is an optimal choice of $\tilde{x}_1$, which maximizes the variation. There is, and the choice is the eigenvector corresponding to the largest eigenvalue of the covariance matrix of $\mathbf{x}$. The eigenvector corresponding to the second largest eigenvalue gives the second axis, and so on. The proof given here follows the one given by S. Sharma [19].

Let $\mathbf{x}$ be a $p \times n$ matrix where p is the number of variables and n is the number of observations. In our case $p = 2$ and $n = 10$. The sample covariance matrix, $\mathbf{\Sigma_x}$, is given by

$$\mathbf{\Sigma_x} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \quad (A.3)$$

where $\bar{\mathbf{x}}$ is the $(p \times 1)$ vector describing the sample mean. The new decorrelated, or variance maximized, variables will be given by linear combinations of the old. Let $\tilde{\mathbf{x}} = \mathbf{C}\mathbf{x}$ denote the new set of variables, where $\mathbf{C} = (\mathbf{c}_1\mathbf{c}_2 \ldots \mathbf{c}_p)^T$ is the $(p \times p)$ matrix where the rows $\mathbf{c}_i^T$ contain the weights of the linear combination . In the example above, $\mathbf{C}$ equals

$$\mathbf{C} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}. \quad (A.4)$$

The covariance matrix of the new variable is given by

$$\begin{aligned} \mathbf{\Sigma}_{\tilde{\mathbf{x}}} &= \frac{1}{n}\sum_{i=1}^{n}(\tilde{\mathbf{x}}_i - \tilde{\bar{\mathbf{x}}})(\tilde{\mathbf{x}}_i - \tilde{\bar{\mathbf{x}}})^T = \quad (A.5) \\ &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{C}\mathbf{x}_i - \mathbf{C}\bar{\mathbf{x}})(\mathbf{C}\mathbf{x}_i - \mathbf{C}\bar{\mathbf{x}})^T = \ldots = \\ &= \mathbf{C}\mathbf{\Sigma}_{\mathbf{x}}\mathbf{C}^T \end{aligned}$$

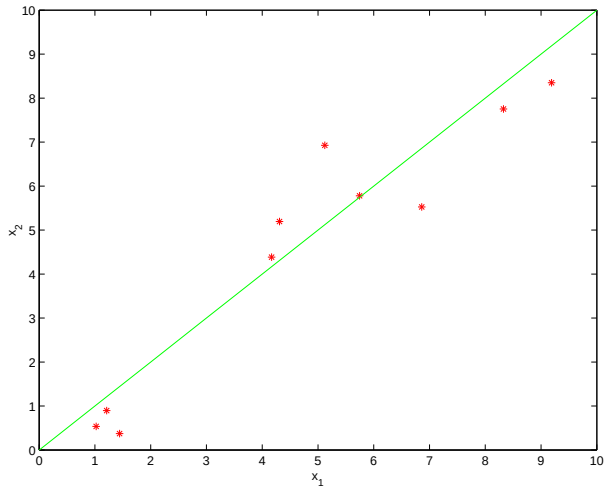


Figure A.1: Some geometrical data along with the line $x_1 = x_2$

Observation	x_1	x_2
1	1.02	0.546
2	1.44	0.372
3	1.21	0.897
4	4.31	5.19
5	4.17	4.39
6	5.75	5.78
7	5.12	6.93
8	6.86	5.53
9	8.33	7.75
10	9.19	8.35
Variance	48.7 %	51.3 %

Table A.1: Coordinates of the geometrical data

Observation	$\tilde{x}_1$	$\tilde{x}_2$
1	1.09	−0.372
2	1.26	−0.790
3	1.48	−0.260
4	6.73	0.456
5	6.05	0.0015
6	8.15	−0.183
7	8.55	1.06
8	8.73	−1.16
9	11.4	−0.692
10	12.4	−0.906
Variance	97.4 %	2.6 %

Table A.2: Coordinates of the projected data

This means that the variance of $\tilde{\mathbf{x}}_i = c_i^T \mathbf{x}$ equals $c_i^T \mathbf{\Sigma_x} c_i$. To maximize this, the optimal $\mathbf{C}$ must be determined subject to $c_i^T \perp c_j^T$, $i \neq j$.

The solution can be found by optimizing $\mathbf{C\Sigma_x C}^T$, subject to $\mathbf{CC}^T = \mathbf{I}$ (i.e. no scaling), using Lagrange multipliers.

Let

$$Z = \mathbf{C\Sigma_x C}^T - \lambda(\mathbf{CC}^T - \mathbf{I}). \tag{A.6}$$

The partial derivative is given by

$$\frac{\partial Z}{\partial \mathbf{C}} = 2\mathbf{\Sigma_x C}^T - 2\lambda \mathbf{C}^T. \tag{A.7}$$

Setting the above to zero yields

$$(\mathbf{\Sigma_x} - \lambda \mathbf{I})\mathbf{C}^T = \mathbf{0} \tag{A.8}$$

Rearranging we get,

$$\mathbf{\Sigma_x C}^T = \lambda \mathbf{C}^T. \tag{A.9}$$

This is recognized as the definition of eigenvectors and eigenvalues. That is, for the above to be true, the rows of $\mathbf{C}$ must contain the eigenvectors of $\mathbf{\Sigma_x}$ and λ holds the corresponding eigenvalues.

In conclusion, to find the new basis that maximizes the variance of the data, the rows of $\mathbf{C}$ should be chosen as the eigenvectors of $\mathbf{\Sigma_x}$. In the

example above we get eigenvalues 152.1 and 4.019 corresponding to eigenvectors $(0.6970, 0.7171)$ and $(-0.7171, 0.6970)$ respectively. The first principal component is therefore the line $x_1 = \frac{0.7171}{0.6970} x_2 = 1.029 x_2$, not far off the initial guess. This component accounts for 97.4% of the total variance. Therefore, if suitable, the second principal component could be omitted, thus reducing dimensionality; one of the main purposes of PCA.

Appendix B

Thin-Plate Spline Warping

Imagine an image printed on an (infinitely) elastic rubber sheet. Picture piercing the sheet with a set of pins and dragging these to new positions, thus transforming the image. The image represents visualization data and the pins represent landmarks. Dragging the pins to new positions translates to transforming the data to fit to new landmark positions, such as the mean landmarks. This type of transformation is an example of image warping.

Warping m-dimensional data $\mathbf{x}$ with landmarks $\mathbf{x}_i$ to m-dimensional data $\mathbf{y}$ with landmarks $\mathbf{y}_i$ is performed using a multivariate function $\mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ which preferably holds the following properties:

- continuous
- smooth
- bijective
- interpolating, i.e. $\mathbf{f}(\mathbf{x}_i) = \mathbf{y}_i$, $i = 1 \ldots n$

The rubber sheet warping mentioned above can be achieved using the bivariate function $(x', y') = \mathbf{f}(x, y) = (f_1(x, y), f_2(x, y))$. Since the equations for x' and y' are independent, the rest of the discussion will focus on a single scalar valued thin-plate spline function.

Warping is essentially the same as interpolation. With interpolation, the task is to find suitable values in-between known data while with warping,

the task is to find suitable positions for data in-between known positions. In one dimension interpolation can be performed using piecewise cubic polynomials called natural cubic splines. These lead to globally smooth functions, i.e. the second order derivatives are continuous throughout the spline. Physically, the cubic spline represents a thin metal rod which is somehow held in place at the points where data is known. The rod will take on a shape that minimizes its internal bending energy. The extension of cubic splines to $m \geq 2$ dimensions are thin-plate splines where, as the name suggests, the steel rod has been replaced by a thin steel plate.

The bending energy function of the steel plate is

$$\iint_{\mathbf{R}^2} (f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2)\, dx\, dy \tag{B.1}$$

not taking into account other physical factors such as gravity. The function $f(x, y)$ that minimizes this bending energy is

$$f(x, y) = \sum_{j=1}^{n} w_j U(r) + a_0 + a_1 x + a_2 y \tag{B.2}$$

where

$$U(r) = r^2 \log r^2 \tag{B.3}$$

and

$$r = \sqrt{(x^2 + y^2)} \tag{B.4}$$

The coefficients w_j, a_0, a_1 and a_2 are unknown, but the constraints imposed by $\mathbf{f}(\mathbf{x}_i) = \mathbf{y}_i$ and the wish to minimize the total bending energy makes it possible to determine these by solving linear systems of equations.

For n landmarks, the exact interpolation requirement gives the following n equations:

$$x'_i = \sum_{j=1}^{n} w_j U_{ij} + a_0 + a_1 x_i + a_2 y_i\,, \quad 1 \leq i \leq n \tag{B.5}$$

where $U_{ij} = U(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})$. This can be expressed in matrix form as

$$\mathbf{x}' = E\mathbf{w} + X\mathbf{a} \tag{B.6}$$

where $E = [U_{ij}]$ $(n \times n)$ and $X = [X_i] = [1 \;\; x_i \;\; y_i]$ $(n \times 3)$. The requirement for minimized bending energy gives

$$X^T \mathbf{w} = \mathbf{0} \tag{B.7}$$

Solving equation B.6 for $\mathbf{w}$ gives

$$\mathbf{w} = E^{-1}(\mathbf{x}' - X\mathbf{a}) \tag{B.8}$$

Inserting this into equation B.7 and solving for $\mathbf{a}$ gives

$$\mathbf{a} = (X^T E^{-1} X)^{-1} X^T E^{-1} \mathbf{x}' \tag{B.9}$$

Equations B.8 and B.9 are the analytical expressions for all unknown parameters collected in $\mathbf{w}$ and $\mathbf{a}$. Together these form $n+3$ equations for the same number of unknowns. The system can therefore be solved, and the coefficients are put into equation B.2 which then can be used for interpolation or warping.

For an exhaustive description of thin-plate spline warping, see Bookstein [6].

Appendix C

The Iterative Closest Point Algorithm

In 1992 General Motors scientists Paul Besl and Neil McKay presented the paper *A Method for Registration of 3-D Shapes* [3] where the popular Iterative Closest Point (ICP) algorithm is stated. The algorithm is concerned with rotating and translating a shape P according to a model shape X so that the inter-shape distance is minimized. Any type of geometric data can be used such as point sets, implicit and parametric curves, implicit and parametric surfaces, polygon sets etc., as long as it is possible to calculate the closest point on X from a given point in P.

The discussion here concerns polygonal surfaces defined from a set of points since that is what applies in this thesis. However, the methodology is the same for any other type of geometry, since a suitable set of points can be chosen from any representation.

The algorithm is sure to converge monotonically to the nearest local minimum, represented by the local best fit of P onto X. To find the global minimum, P must be given an appropriate initial position and orientation.

Let n_P denote the number of points in P. For each point $\mathbf{p}_i$ of P, find the corresponding closest point of X. The resulting set of closest points $\{\mathbf{y}_i\}$, $i = 1 \dots n_P$ is denoted Y. In other words, Y is an approximation of X with the same number of points and point ordering as P. Note that unless X

is a pure point set, the closest point operation is calculated as the shortest distance from a point in P to the continuous surface or curve of X.

To measure how well P has been fitted to X any distance metric can be used. The most common choice is the sum of mean squares metric,

$$d_{MS} = \frac{1}{n_P} \sum_{i=1}^{n_P} \|\mathbf{y}_i - \mathbf{p}_i\|^2 \tag{C.1}$$

Other choices are the RMS metric, $d_{RMS} = \sqrt{d_{MS}}$, and the mean absolute value.

The ICP algorithm proceeds as follows:

1. Compute the set Y of n_P closest points from P to X
2. Compute the rigid-body transformation Γ that optimally aligns the points of P with Y
3. Iterate until Y is stable, which means that the change in the error bound d has fallen below a certain threshold.

The idea is simple, but as Besl and McKay shows, it is magnitudes faster than any regular multivariate optimization technique such as Nelder-Mead [5] and conjugate gradient [5].

The remaining issue is how to find the optimal transformation Γ. This can be done using singular value decomposition of the cross-covariance matrix of P and Y. Besl and McKay have adopted a quaternion-based approach described by Horn [14].

Quaternions are complex number with one real and three imaginary parts, $q = q_0 + iq_1 + jq_2 + kq_3$. The *unit* quaternion, here represented as a vector, is $\mathbf{q}_R = [q_0 q_1 q_2 q_3]^T$ where $q_0 \geq 0$ and $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. Unit quaternions can be used to represent rotations in 3-D. The 3×3 rotation matrix $\mathbf{R}$ generated by a unit quaternion is

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{C.2}$$

Let $\mathbf{q}_T = [q_4 q_5 q_6]^T$ denote a translation vector. The transformation Γ can now be defined by the vector $\mathbf{q} = [\mathbf{q}_R^T | \mathbf{q}_T^T]^T$. For $\mathbf{q}$ to optimally align the

computed points Y to the points on P we wish to minimize the mean square objective function

$$f(\mathbf{q}) = \frac{1}{n_P} \sum_{i=1}^{n_P} \|\mathbf{y}_i - \mathbf{R}(\mathbf{q}_R)\mathbf{p}_i - \mathbf{q}_T\|^2 \tag{C.3}$$

Let

$$Q(\Sigma_{PY})_{4\times 4} = \begin{bmatrix} trace(\Sigma_{PY}) & \Delta^T \\ \Delta & \Sigma_{PY} + \Sigma_{PY}^T - trace(\Sigma_{PY})\mathbf{I}_3 \end{bmatrix} \tag{C.4}$$

where Σ_{PY} is the 3×3 cross-covariance matrix of P and Y and $\Delta = [A_{23}A_{31}A_{12}]$ where $A_{ij} = (\Sigma_{PY} - \Sigma_{PY}^T)_{ij}$. The unit eigenvector $[q_0 q_1 q_2 q_3]^T$ corresponding to the largest eigenvalue of matrix Q is then the optimal rotation quaternion q_R. The optimal translation is found by rotating the center of mass $\bar{\mathbf{p}}$ of P with $\mathbf{q}_R$ and subtracting the result from the center of mass $\bar{\mathbf{x}}$ of X, i.e. $q_T = \bar{\mathbf{x}} - \mathbf{R}(\mathbf{q}_R)\bar{\mathbf{p}}$.

The optimal transformation Γ is here defined as a rigid-body transformation, but can be any other transformation. To allow scaling in the fitting process, a similarity transform can be used.

Appendix D

Procrustes Analysis

The notation and majority of equations in this section are from [10].

Procrustes analysis [13, 2, 12] is an important method in shape analysis. Following D.G. Kendall, the following definition of shape is given.

Shape *is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.*

From this, it is understood that before shape analysis can be performed on a set of objects, they must first be transformed, giving them the same position, orientation and scale. This can be achieved using Procrustes analysis.

The following main variants on Procrustes analysis exist

- **Full Ordinary Procrustes Analysis (Full OPA)**. The matching of two configurations using the full set of similarity transforms, rotation, translation and scaling.
- **Partial Ordinary Procrustes Analysis (Partial OPA)**. The matching of two configurations using only rotation and translation resulting in a shape-and-size model.
- **Full Generalized Procrustes Analysis (Full GPA)**. The alignment of two or more configurations to their mutual mean using the full set of similarity transforms, rotation, translation and scaling.

- **Partial Generalized Procrustes Analysis (Partial GPA)**. The alignment of two or more configurations to their mutual mean using only rotation and translation resulting in a shape-and-size model.

Note that contrary to generalized Procrustes analysis, ordinary Procrustes analysis is not symmetric to the ordering of objects.

To be able to perform measurements on shapes, two important metrics are needed, *shape size* and *shape distance*. It is also important to be able to make an estimation of the *average shape*.

The shape size simply determines the size of a shape, so that it can be compared to other shapes of the same class. The measure most commonly used is the *centroid size*

$$S(X) = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{k}(X_i^j - \bar{X}^j)^2} \tag{D.1}$$

where n is the number of dimensions, k is the number of points, X_i^j is the ith point of the jth dimension and $\bar{X}^j$ is the mean of all points of the jth dimension.

The shape distance is the distance between two shapes. A low distance indicates that the shapes are similar. The measurement is called the (full or partial) *Procrustes distance*. The calculation is based on the eigenvalues of a matrix derived from the object data. The shapes need to be *pre-shapes*, i.e. translation and scaling are filtered out. However, if the shapes are fully aligned, a simpler estimate of the distance can be used

$$d(X_1, X_2) = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{k}(X_{1(i)}^j - X_{2(i)}^j)^2} \tag{D.2}$$

As can be seen, this is the root of the sum of all squared point distances.

The estimate of the average shape is based on an estimate of the mean shape. Notice the difference between the average shape, which is the true average of the object class in question, and the mean shape, which is the mean of the objects at hand. The question is now how to estimate the mean shape. Once the objects are aligned, the easiest and most commonly used method is to calculate the mean position of each landmark throughout the

set.

$$\bar{X} = \frac{1}{m}\sum_{i=1}^{m} X_i \tag{D.3}$$

For planar data, there exists a method where the data does not have to be aligned. This is described below, see equation D.23.

In the following sections, it is assumed that the objects are *centered*. This means that the center-of-mass of the landmarks is at the origin. Objects can easily be centered by subtracting the center-of-mass from each landmark. In matrix form this can be written as

$$X_{centered} = CX \tag{D.4}$$

where $C = I_k - \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^T$ and k is the number of landmarks.

D.1 Planar Ordinary Procrustes Analysis

For two-dimensional (planar) geometry it is advantageous to represent coordinates as complex numbers with the first and second dimension represented by the real and imaginary part. A complex number z can be written in either euclidian or polar form:

$$z = a + ib = re^{i\theta} \tag{D.5}$$

where $(a, b) \in \mathbf{R}$, $r = |z|$ and $\theta = \arg z$. For vectors and matrices of complex numbers, $\mathbf{y}^*$ denotes the complex conjugate of the transpose of $\mathbf{y}$. The complex conjugate of a complex number is $\overline{a + ib} = a - ib = \overline{re^{i\theta}} = re^{-i\theta}$.

Let $\mathbf{y} = [y_1, \ldots, y_k]^T$ and $\mathbf{w} = [w_1, \ldots, w_k]^T$ be landmarks in $\mathbf{C}^k$ of two objects. Assume that both configurations are centered, i.e. $\mathbf{y}^*\mathbf{1}_k = \mathbf{w}^*\mathbf{1}_k = 0$. To find the rotation, translation and scaling that optimally aligns $\mathbf{w}$ with $\mathbf{y}$, $\mathbf{y}$ is expressed in terms of $\mathbf{w}$ with a complex regression equation:

$$\mathbf{y} = (a + ib)\mathbf{1}_k + \beta e^{i\theta}\mathbf{w} + \epsilon \tag{D.6}$$

Here, $(a + ib)\mathbf{1}_k$ is a $k \times 1$ vector representing translation, β is uniform scaling, θ represents rotation and ϵ is an error vector. The optimal values of a, b, β and θ occur when the sum of squares of ϵ is minimal. The objective function to be minimized is therefore

$$D^2(\mathbf{y}, \mathbf{w}) = \epsilon^*\epsilon = \tag{D.7}$$
$$(\mathbf{y} - (a + ib)\mathbf{1}_k - \beta e^{i\theta}\mathbf{w})^*(\mathbf{y} - (a + ib)\mathbf{1}_k - \beta e^{i\theta}\mathbf{w})$$

The expansion of this product is:

$$\begin{aligned}&\mathbf{y}^*\mathbf{y} - (a + ib)\mathbf{y}^*\mathbf{1}_k - \beta e^{i\theta}\mathbf{y}^*\mathbf{w} - \\ &(a - ib)\mathbf{1}_k^*\mathbf{y} + (a - ib)(a + ib)\mathbf{1}_k^*\mathbf{1}_k + (a - ib)\beta e^{i\theta}\mathbf{1}_k^*\mathbf{w} - \\ &\beta e^{-i\theta}\mathbf{w}^*\mathbf{y} + (a + ib)\beta e^{-i\theta}\mathbf{w}^*\mathbf{1}_k + \beta^2 e^{-i\theta}e^{i\theta}\mathbf{w}^*\mathbf{w}\end{aligned} \tag{D.8}$$

Since $\mathbf{y}^*\mathbf{1}_k = \mathbf{w}^*\mathbf{1}_k = 0$, this can be simplified to:

$$\mathbf{y}^*\mathbf{y} - \beta e^{i\theta}\mathbf{y}^*\mathbf{w} + k(a^2 + b^2) - \beta e^{-i\theta}\mathbf{w}^*\mathbf{y} + \beta^2\mathbf{w}^*\mathbf{w} \tag{D.9}$$

It is immediately seen that for D^2 to be minimal, a and b must be zero. It is also seen that $\beta(\mathbf{y}^*\mathbf{w}e^{i\theta} + \mathbf{w}^*\mathbf{y}e^{-i\theta})$ should be maximal. Substituting $\mathbf{y}^*\mathbf{w}$ with $\gamma e^{i\phi}$ and noting that $\mathbf{w}^*\mathbf{y} = (\mathbf{y}^*\mathbf{w})^* = \gamma e^{-i\phi}$ yields

$$\beta(\gamma e^{i\theta}e^{i\phi} + \gamma e^{-i\theta}e^{-i\phi}) = \beta(\gamma e^{i(\theta+\phi)} + \gamma e^{-i(\phi+\theta)}) = 2\beta\gamma\cos(\theta + \phi) \tag{D.10}$$

This is maximal for $\theta + \phi = 0 + k2\pi$, $k \in \mathbf{Z}$. Since $\phi = \arg(\mathbf{y}^*\mathbf{w})$, one solution is $\hat{\theta} = -\arg(\mathbf{y}^*\mathbf{w})$.

To find the scaling, D^2 is differentiated with respect to β and set to zero. Using D.10 this yields

$$\frac{\partial D^2}{\partial \beta} = 2\beta\mathbf{w}^*\mathbf{w} - 2\gamma\cos(\theta + \phi) = 2\beta\mathbf{w}^*\mathbf{w} - 2\gamma = 0 \tag{D.11}$$

Hence,

$$\hat{\beta} = \frac{|\mathbf{y}^*\mathbf{w}|}{\mathbf{w}^*\mathbf{w}} = \frac{\sqrt{\mathbf{w}^*\mathbf{y}\mathbf{y}^*\mathbf{w}}}{\mathbf{w}^*\mathbf{w}} \tag{D.12}$$

To sum up, the optimal parameters for aligning a set of landmark $\mathbf{y}$ onto $\mathbf{w}$ using equation D.6 are

$$\hat{a} + i\hat{b} = 0 \tag{D.13}$$
$$\hat{\theta} = -\arg(\mathbf{y}^*\mathbf{w}) = \arg(\mathbf{w}^*\mathbf{y}) \tag{D.14}$$
$$\hat{\beta} = \frac{|\mathbf{y}^*\mathbf{w}|}{\mathbf{w}^*\mathbf{w}} = \frac{\sqrt{\mathbf{w}^*\mathbf{y}\mathbf{y}^*\mathbf{w}}}{\mathbf{w}^*\mathbf{w}} \tag{D.15}$$

This solution seems simple enough, but can be further simplified. Rewrite equation D.6 in matrix form:

$$\mathbf{y} = XP + \epsilon \tag{D.16}$$

where $X = [\mathbf{1}_k \ \mathbf{w}]$ and $P = [a+ib \ \beta e^{i\theta}]^T$. Let $\hat{P} = [0 \ \hat{\beta}e^{i\hat{\theta}}]$ be the optimal parameters. The solution can be written in the same form as the standard least squares solution, but with complex variables. Consult a statistics textbook for a comparison.

$$\hat{P} = (X^*X)^{-1}X^*\mathbf{y} \tag{D.17}$$

Expanding this equation gives

$$\hat{P} = \begin{bmatrix} 0 \\ \hat{\beta}e^{i\hat{\theta}} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{1}_k^* \\ \mathbf{w}^* \end{bmatrix} [\mathbf{1}_k \ \mathbf{w}] \right)^{-1} \begin{bmatrix} \mathbf{1}_k^* \\ \mathbf{w}^* \end{bmatrix} \mathbf{y} = \tag{D.18}$$

$$\begin{bmatrix} k & \sum \mathbf{w} \\ \sum \mathbf{w}^* & \mathbf{w}^*\mathbf{w} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1}_k^* \\ \mathbf{w}^* \end{bmatrix} \mathbf{y} = \tag{D.19}$$

$$\begin{bmatrix} \frac{1}{k} & 0 \\ 0 & \frac{1}{\mathbf{w}^*\mathbf{w}} \end{bmatrix} \begin{bmatrix} \mathbf{1}_k^* \\ \mathbf{w}^* \end{bmatrix} \mathbf{y} = \tag{D.20}$$

$$\begin{bmatrix} \frac{1}{k}\mathbf{1}_k \\ \frac{\mathbf{w}^*}{\mathbf{w}^*\mathbf{w}} \end{bmatrix} \mathbf{y} = \begin{bmatrix} 0 \\ \frac{\mathbf{w}^*\mathbf{y}}{\mathbf{w}^*\mathbf{w}} \end{bmatrix} \tag{D.21}$$

This results in the convenient expression $\hat{\beta}e^{i\hat{\theta}} = \mathbf{w}^*\mathbf{y}/\mathbf{w}^*\mathbf{w}$. The full Procrustes fit of $\mathbf{w}$ onto $\mathbf{y}$ can now be expressed as

$$\mathbf{w}_{fit} = \frac{\mathbf{w}^*\mathbf{y}\mathbf{w}}{\mathbf{w}^*\mathbf{w}} \tag{D.22}$$

To perform partial OPA, scaling can be left out of the process descibed above.

D.2 Planar General Procrustes Analysis

General Procrustes analysis in two dimensions is similar to its ordinary counterpart. The difference is that the objects are aligned with respect to their mutual mean. However, before the shapes are aligned, the mean estimate in equation D.3 cannot be used. Instead, the mean shape can be found by solving an eigenvalue problem.

The *full Procrustes mean shape* of n objects is the eigenvector corresponding to the largest eigenvalue of the matrix

$$S = \sum_{i=1}^{n} \frac{\mathbf{w}_i\mathbf{w}_i^*}{\mathbf{w}_i^*\mathbf{w}_i} \tag{D.23}$$

When an estimate of the mean shape exist, equation D.22 can be used to align the shapes.

D.3 Multidimensional Ordinary Procrustes Analysis

In n dimensions, ordinary Procrustes analysis is similar to the planar case. Objects are now represented by $(k \times n)$ matrices, where k is the number of landmarks. Just as above, configurations are assumed to be centered. The norm of a matrix X is calculated as $\|X\| = \sqrt{\text{trace}(X^TX)}$.

The full ordinary Procrustes fit of a configuration W onto another configuration Y is performed by minimizing the objective function

$$D^2(W,Y) = \|Y - \beta W\Gamma - \mathbf{1}_k\gamma^T\|^2 \tag{D.24}$$

where Γ is an $(n \times n)$ rotation matrix, β is a scaling parameter and γ is an $(n \times 1)$ translation vector.

The optimal parameters are

$$\hat{\gamma} = \mathbf{0} \tag{D.25}$$

$$\hat{\Gamma} = UV^T \tag{D.26}$$

$$\hat{\beta} = \frac{\text{trace}(Y^TW\hat{\Gamma})}{\text{trace}(W^TW)} \tag{D.27}$$

where U and V are the result of a singular value decomposition of the matrix

$$V\Lambda U^T = \frac{Y^TW}{\|W\|\|Y\|} \tag{D.28}$$

Like before, scaling can omitted to perform a partial OPA.

D.4 Multidimensional General Procrustes Analysis

When matching configurations to a mutual mean in $n \geq 3$ dimensions no linear expression for the mean exist. Instead, an iterative procedure [13, 2]

must be employed. Assume m objects X_i, $i = 1 \dots m$, each consisting of k points in n dimensions.

1. Center all objects to remove translation, denote the centered objects $X_{L(i)}$, $i = 1 \dots m$ (Location removed).
2. For each object $X_{L(i)}$, estimate the mean of all other objects

$$\bar{X}_i = \frac{1}{m-1} \sum_{j \neq i} X_{C(i)} \tag{D.29}$$

   and align $X_{L(i)}$ to this estimate of the mean using rotation-only OPA. Denote the rotated objects $X_{LR(i)}$. Repeat this process using the updated configurations until it converges.
3. Let O be the $(kn \times m)$ observation matrix with the objects represented by the columns as

$$X_{LR(i)} = [x_1^1 \ \dots \ x_k^1 \ \dots \ x_1^n \ \dots \ x_k^n]^T \tag{D.30}$$

   Let Φ be the $(m \times m)$ correlation matrix of this matrix and let $\phi = [\phi_1 \ \dots \ \phi_m]$ be the eigenvector corresponding to the largest eigenvalue of Φ. With this vector at hand, a scaling factor β_i can be calculated for each object,

$$\beta_i = \sqrt{\left(\frac{\sum_{k=1}^{m} \|X_{LR(i)}\|^2}{\|X_{LR(i)}\|^2} \right) \phi_i} \tag{D.31}$$

   The scaled objects are denoted $X_{LRS(i)}$.
4. Repeat step 2 and 3 until a stable solution is found.

This algorithm normally converges very quickly (3-5 iterations) but may appear cumbersome. A simpler solution [8] is to keep an estimate of the mean shape, and in each iteration carry out an OPA which aligns all the shapes to this mean. The new algorithm is a follows

1. Let any object be the first estimate of the mean shape
2. Align all objects to the mean using OPA
3. Make a new estimate of the mean shape using the Procrustes mean defined in equation D.3.
4. Iterate step 2 and 3 until convergence.

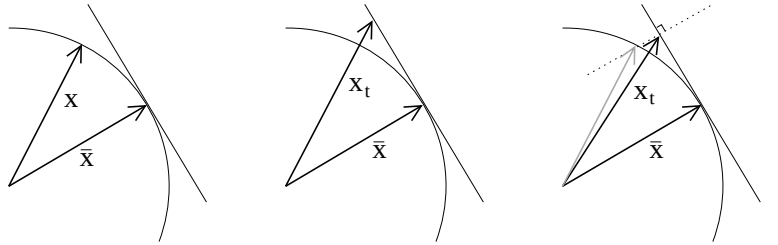


Figure D.1: Tangent space projections in 2-D. The left image shows the mean shape and the unprojected shape. The middle image depicts projection by scaling, and the right image shows projection by scaling and reshaping.

D.5 Tangent Space Projection

The resulting shape vectors of the Procrustes analysis reside in the non-linear vector space called *shape-space*. Most methods for shape analysis, such as principal component analysis (PCA), are linear, and may not work perfectly in conjunction with the Procrustes data. To improve performance, a *tangent space projection* can be carried out. This can be done by scaling the vectors, stretching them to the tangent hyper-plane defined by the mean shape. Another method is to scale and reshape the vectors in a direction perpendicular to the plane. Figure D.1 [20] shows these operations for two-dimensional vectors.

The equation stating how to transform the vectors from shape-space to tangent-space by scaling can be derived using simple linear algebra.

The projection of $\mathbf{x}_t$ onto $\bar{\mathbf{x}}$ is equal to $\bar{\mathbf{x}}$.

$$\frac{\mathbf{x}_t^T \bar{\mathbf{x}}}{\bar{\mathbf{x}}^T \bar{\mathbf{x}}} \bar{\mathbf{x}} = \bar{\mathbf{x}} \Longleftrightarrow \frac{\mathbf{x}_t^T \bar{\mathbf{x}}}{\bar{\mathbf{x}}^T \bar{\mathbf{x}}} = 1 \tag{D.32}$$

The projected vector $\mathbf{x}_t$ consists of a scaled version of the old vector $\mathbf{x}$, $\mathbf{x}_t = \alpha \mathbf{x}$. Using this in the equation above yields

$$\frac{\alpha \mathbf{x}^T \bar{\mathbf{x}}}{\bar{\mathbf{x}}^T \bar{\mathbf{x}}} = 1 \Longleftrightarrow \alpha = \frac{\bar{\mathbf{x}}^T \bar{\mathbf{x}}}{\bar{\mathbf{x}}^T \mathbf{x}} \Longleftrightarrow \mathbf{x}_t = \alpha \mathbf{x} = \frac{\bar{\mathbf{x}}^T \bar{\mathbf{x}}}{\bar{\mathbf{x}}^T \mathbf{x}} \mathbf{x} \tag{D.33}$$

In reality, the tangent space projection has little impact on methods such

as PCA. However, since the operation is easy to perform, and because it is nice from a theoretical viewpoint, it should not be omitted.

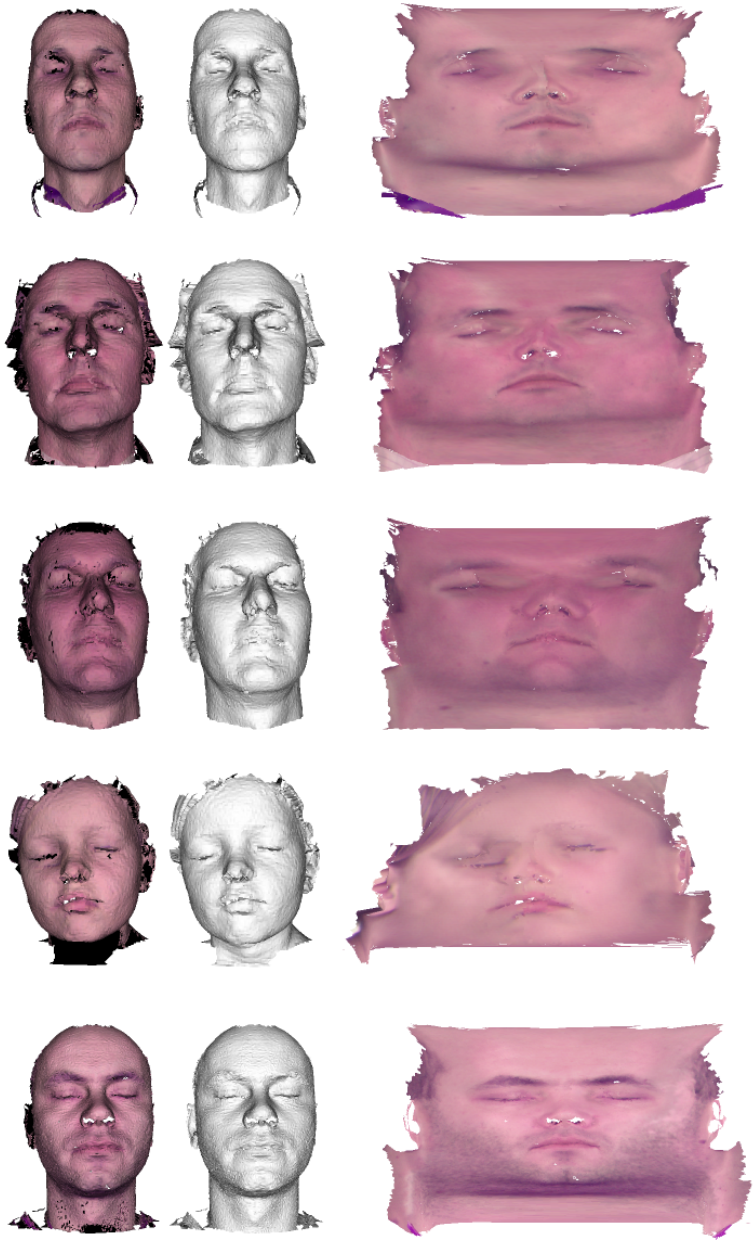
Appendix E

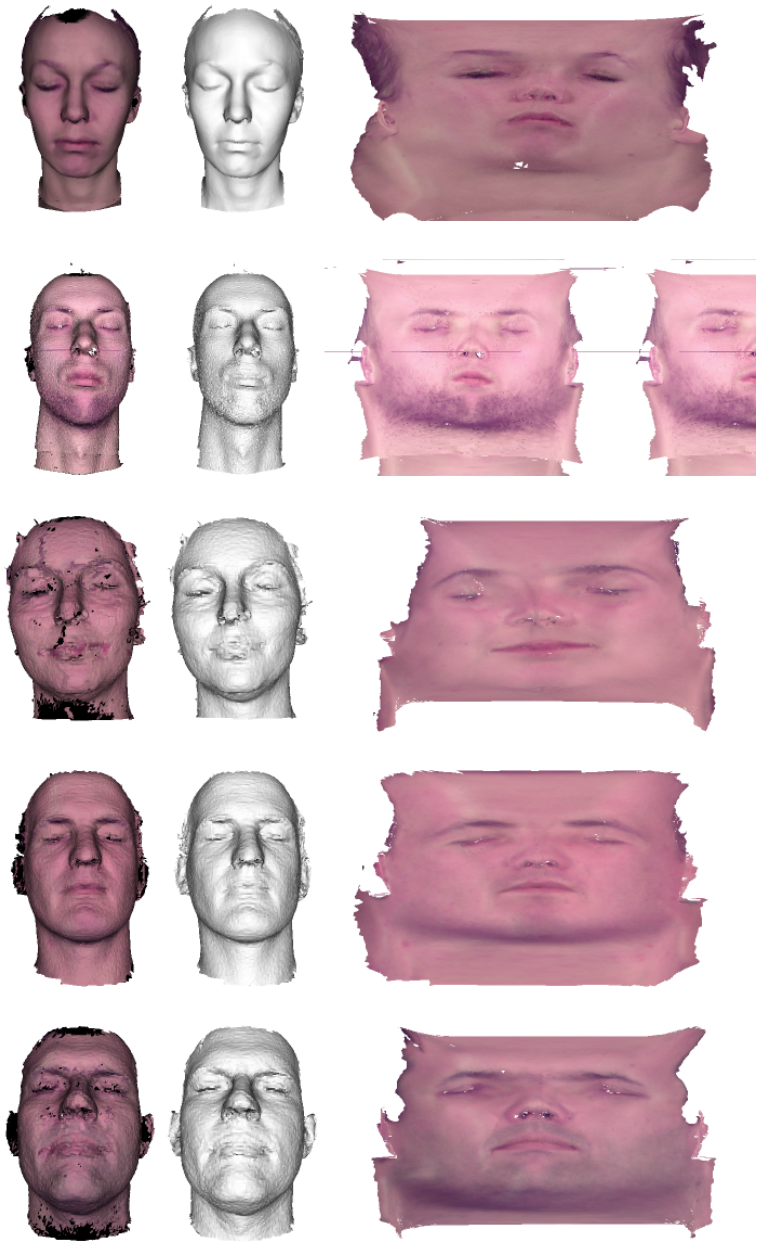
3-D Face Database

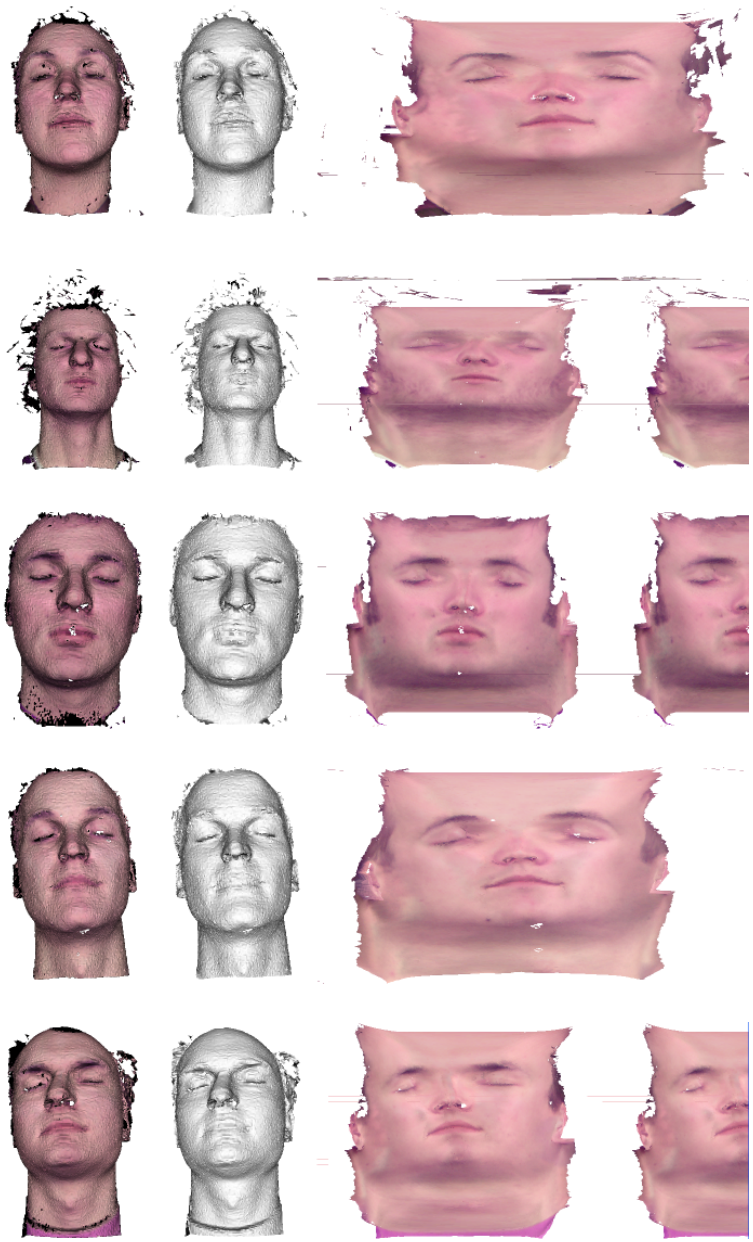
This chapter presents the 24 faces that make up the database used in this project. Each row present one face, with the textured shape to the left, the shape in the middle and the texture to the right.

E.1 Raw Data

The data shown in this section is the raw face data produced by the scanner software. No smoothing, connectivity analysis etc. has been performed.

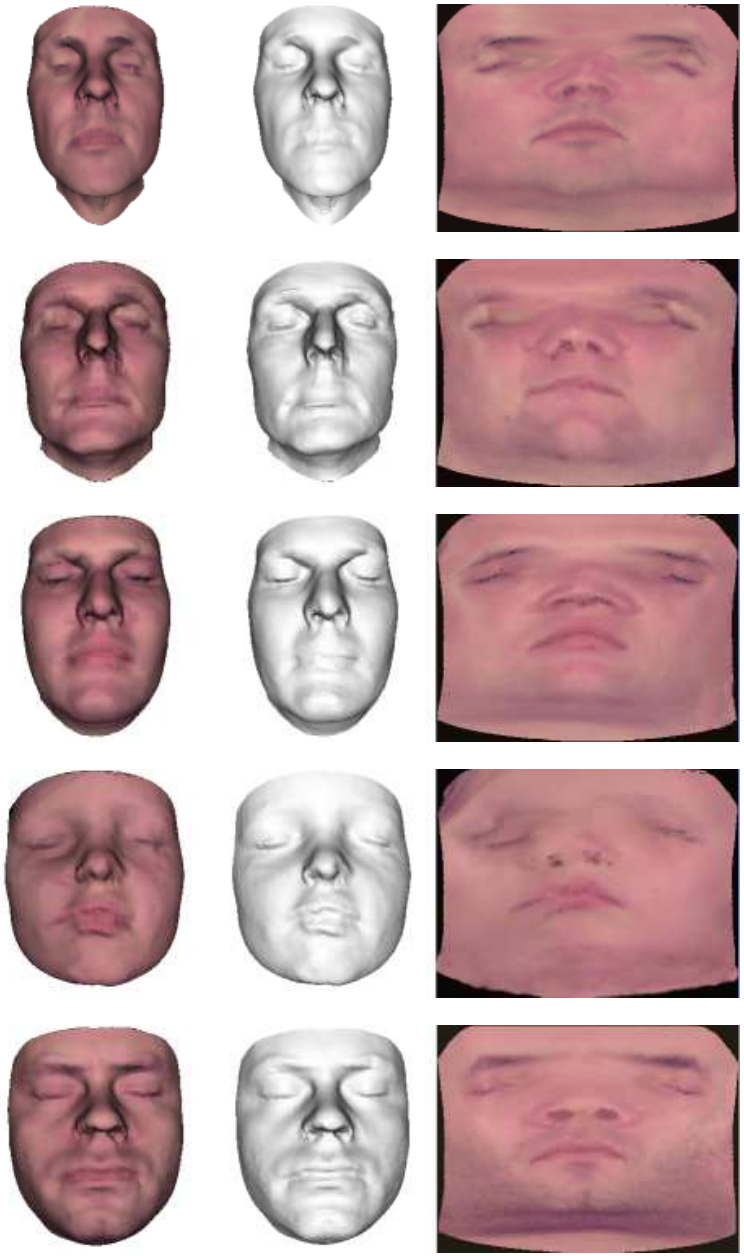


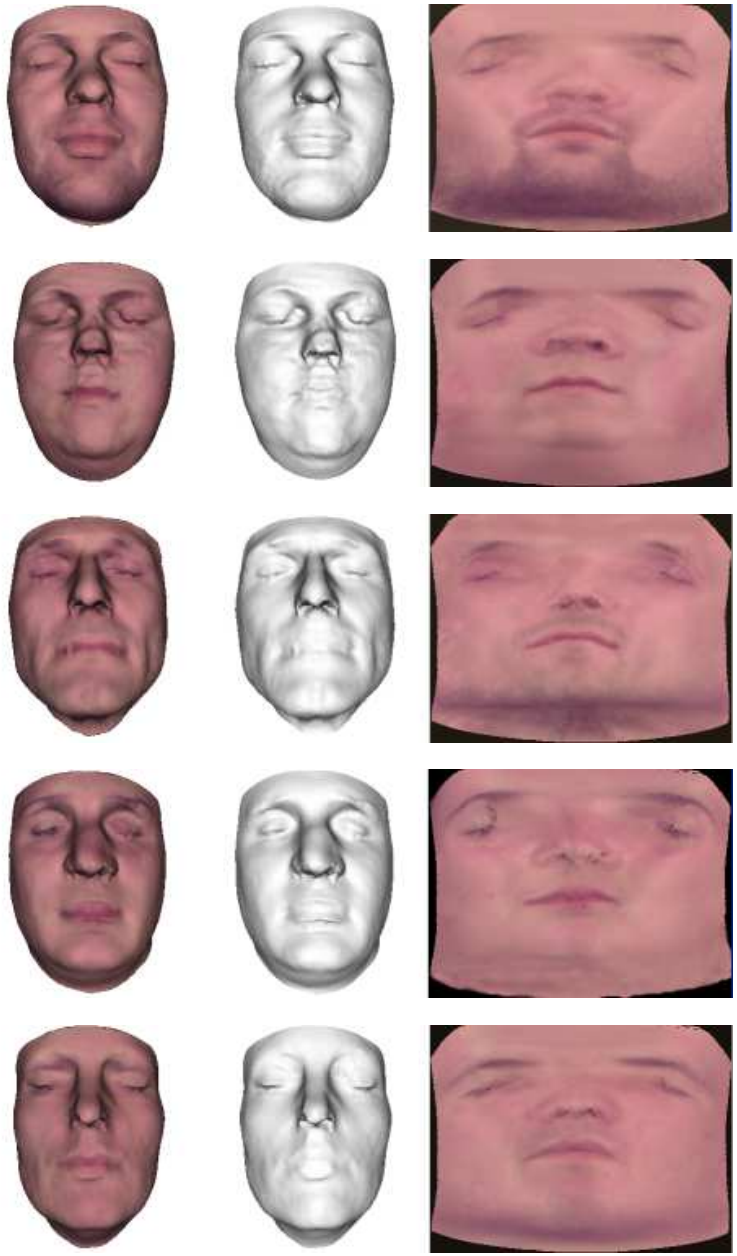


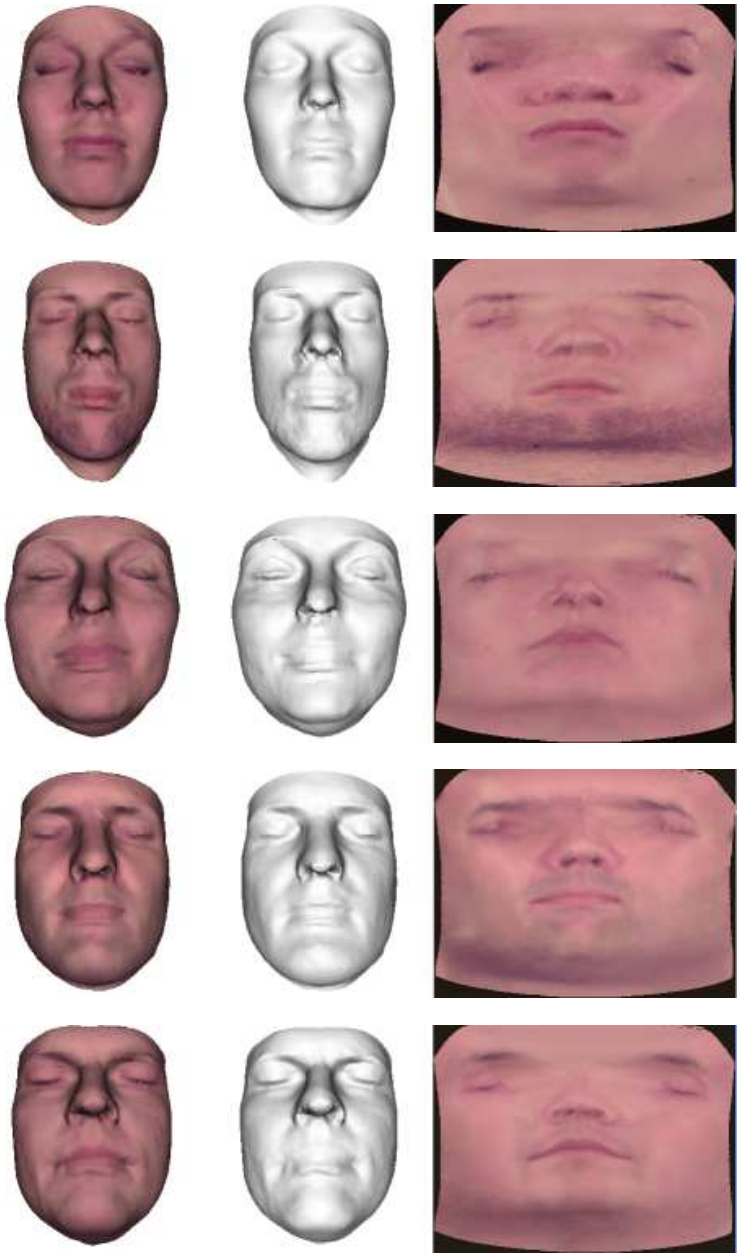


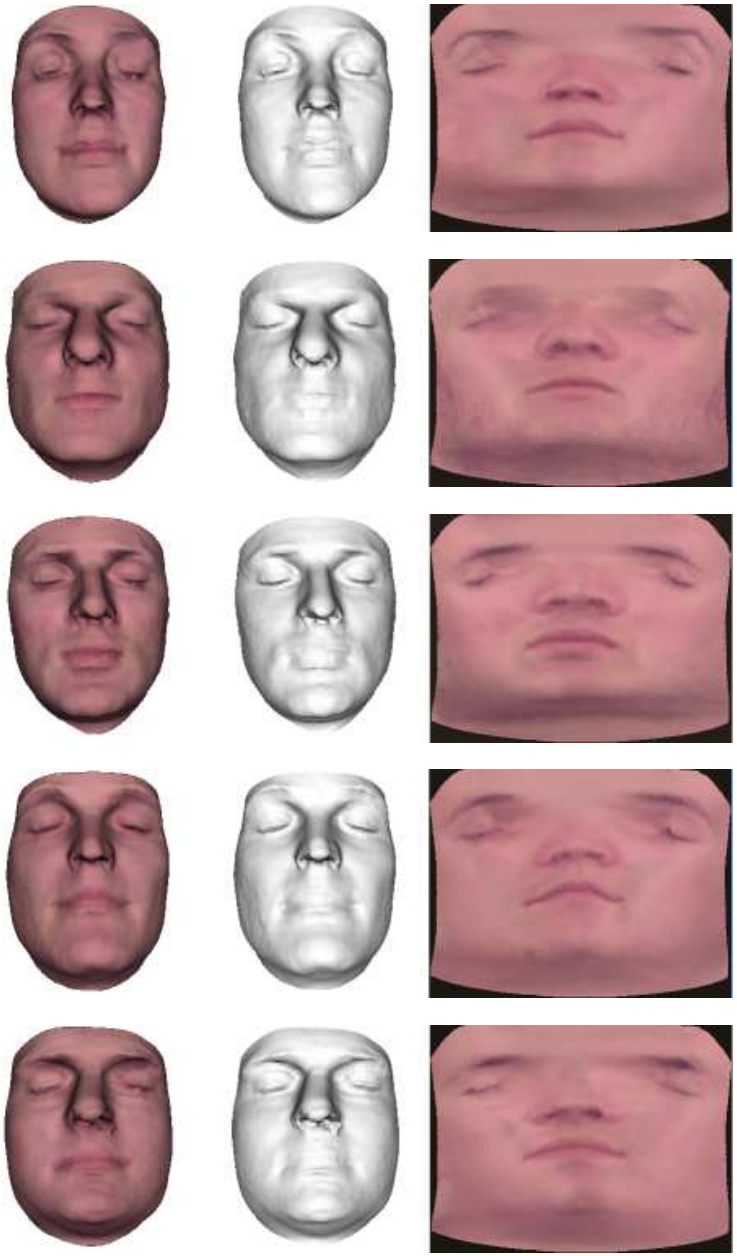
E.2 Processed Data

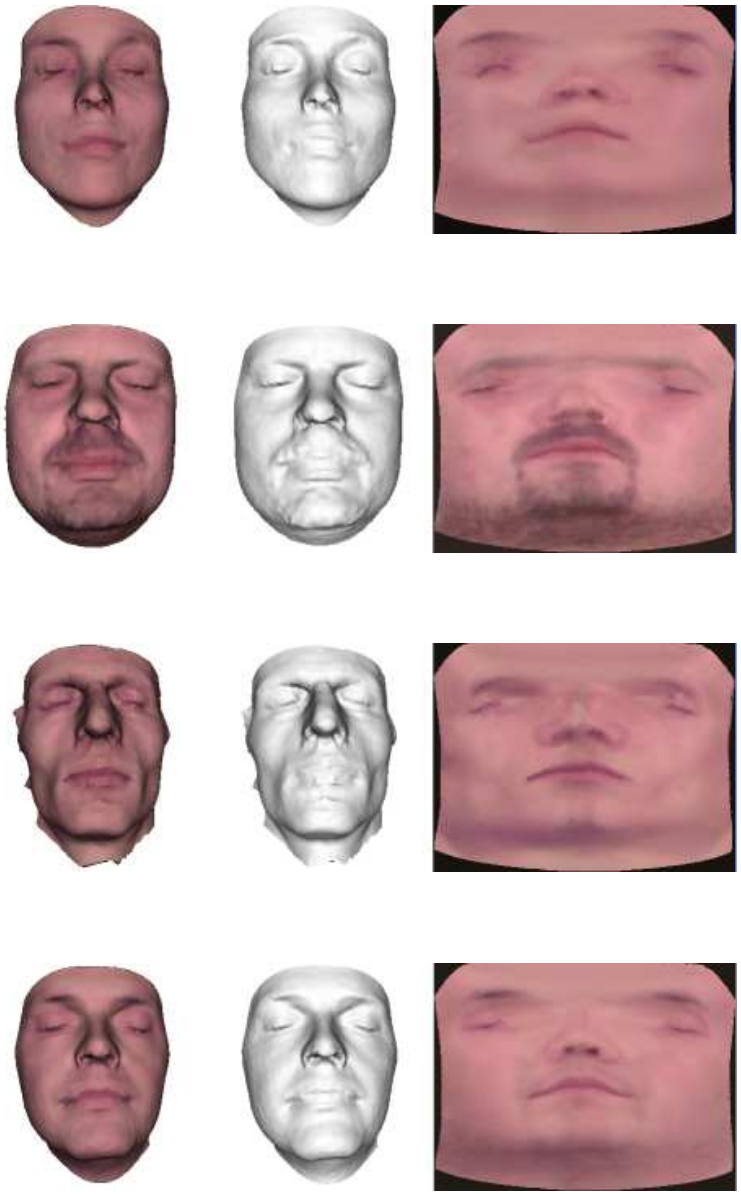
This section shows the data as used by the modelling software. The shapes are smoothed, registered and aligned. The textures are aligned with respect to geometry, intensity and color balance, and cropped.











Bibliography

[1] J. Ahlberg. *Model-based Coding*. PhD thesis, Linkoping University, SE-581 83 Linkoping, Sweden, 2002.

[2] J. M. F. Ten Berge. Orthogonal Procrustes rotation for two or more matrices. *Psychometrika*, 42:267–276, 1977.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. In *IEEE transactions on pattern analysis and machine intelligence*, volume 14, February 1992.

[4] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 187–194, Los Angeles, 1999. Addison Wesley Longman.

[5] Lars-Christer Boijers. *Lectures On Optimization*. KFS AB, 2001.

[6] F. L. Bookstein. Shape and the information in medical images: A decade of the morphometric synthesis. In *Computer Vision and Image Understanding*, pages 97–118, 1997.

[7] Knut Conradsen. *En introduktion til statistik*. Imsor, DTH, 4th edition, 1984.

[8] T.F Cootes and C.J. Taylor. *Statistical Models of Appearance for Computer Vision*. University of Manchester, 2001.

[9] T.F. Cootes, C.J. Taylor, Cooper D., and Graham J. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61:38–59, 1995.

[10] Ian L. Dryden and Kanti V. Mardia. *Statistical Shape Analysis*. John Wiley & Sons, 1999.

[11] T. Zhang G. Taubin and G. Golub. Optimal surface smoothing as filter design. Technical Report 90237, IBM & Stanford University, December 1996.

[12] C. Goodall. Procrustes methods in the statistical analysis of shape. 53(2):285–339, 1991.

[13] J. C. Gower. Generalized Procrustes analysis. *Psychometrika*, 40:33–50, 1975.

[14] B. K. B. Horn. Closed-form solution of absolute orientation using unit quaternitons. *J. Opt. Soc. Amer. A*, 4:629–642, April 1987.

[15] H. Hotelling. Analysis of complex statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[16] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Proceedings of the First International Conference on Computer Vision (Cat. No.87CH2465-3)*, pages 259–68, 1987.

[17] R. R. Paulsen and K. Hilger. Shape modelling using markov random field restoration of point correspondences. In *Information Processing in Medical Imaging*, IPMI, 2003.

[18] R. R. Paulsen, R. Larsen, S. Laugesen, C. Nielsen, and B. K. Ersbøll. Building and testing a statistical shape model of the human ear canal. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2002, 5th Int. Conference, Tokyo, Japan,*. Springer, 2002.

[19] S.C. Sharma. *Applied Multivariate Techniques*. John Wiley & Sons, 1996.

[20] M. B. Stegmann and D. D. Gomez. A brief introduction to statistical shape analysis, Mars 2002.

[21] M.B. Stegmann. Active appearance models - theory, extensions & cases. Master's thesis, Technical University of Denmark, 2000.

[22] Boudewijn P.F. Lelieveldt Rob J. van der Geest Johan H.C. Reiber Milan Sonka Steven C. Mitchell, Johan G. Bosch. 3-D active appearance models: Segmentation of cardiac MR and ultrasound images. 2002.

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

[24] K.N. Walker T.F. Cootes, G.V. Wheeler and C.J. Taylor. View-based active appearance models. In *Image and Vision Computing*, volume 20, pages 657–664. Elsevier Science B.V, 2002.

[25] H. H. Thodberg. Minimum description length shape and appearance models. In *Image Processing Medical Imaging, IPMI 2003*, 2003.

[26] B.F. Buxton T.J. Hutton and P. Hammond. Dense surface point distribution models of the human face, 2001.

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