Sketching 3D Faces

Örn Gunnarsson

Department of Computer Science

University of Sheffield

A thesis submitted for the degree of

Doctor of Philosophy

January, 2010

Acknowledgements

I would like to thank my supervisor Steve Maddock for his unwavering support and guidance during the past four years. I have learned a lot from his experience and analytical thinking, but I promise I will only use my honed skills for the good of mankind. I would also like to show my gratitude to Guido Sanguinetti who was always willing to spare time to answer my questions on statistical models.

This thesis would not have been possible without the support of my parents Gunnar and Erna, my siblings Valur, Kristín og Guðni, my grandparents Guðni og Valgerður, and my uncle Kristján.

I would like to extend special thanks to Ian Peter Frost and Murad Abouammoh for their invaluable friendship and help. I will always think fondly about Salih Tuna and Berna Keskin for pottering around with me in Sheffield. I am grateful to have shared my experience and exchanged knowledge with my colleagues in the Computer Graphics group.

I feel very lucky to have met my corazon Desirée during my research here. My love goes out to her and her wonderful family, Juan, Vicenta, Lorena, Noemi, and abuela Vicenta.

Abstract

Faces can be modelled using a number of techniques. Existing faces can be transferred to a digital form using equipment such as laser scanners. New faces can be constructed using commercial modelling tools or using specialist software such as parameterised based systems. This thesis presents a technique for modelling and posing 3D face models using an intuitive sketch-based approach. In contrast to existing sketch-based systems that manipulate faces, it gives the user an unhindered and natural sketching experience as there is no need to sketch helper strokes, incrementally alter parameters, or define which features are being sketched in order to aid the modelling/posing process. A Non-Photorealistic Rendering (NPR) technique creates a sketch-like representation of the 3D faces. Hidden Markov Models (HMMs) map a sequence of sketched points to the NPR data which is used in a maximum likelihood framework to generate a face model based on the observed data. It is shown that this method can produce a range of novel 3D faces for the purposes of modelling new facial features, as well as quickly posing existing features to create key poses in an animation sequence. Two user evaluations are presented here. The first evaluation verifies that the use of contour data of a face can be used to reconstruct the full face mesh while preserving the facial features. The second evaluation demonstrates how this sketch-based technique can be used to recreate faces from description by novice users.

Supporting Publications

Conference Papers

Gunnarsson, O. and S.Maddock (2010), "Sketch-Based Posing of 3D Faces for Facial Animation", *Theory and Practice of Computer Graphics*, The UK Chapter of the Eurographics Association (EGUK), September, 2010.

Gunnarsson, O. and S.Maddock (2008), "Sketching Faces", *Proc. Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling 2008 (SBIM 2008)*, Annecy, France, June 11-13, 2008.

Gunnarsson, O. and S.Maddock (2007), "A Statistically-Assisted Sketch-Based Interface for Creating Arbitrary 3D Faces", *Theory and Practice of Computer Graphics*, The UK Chapter of the Eurographics Association (EGUK), June, 2007.

Other

Gunnarsson, O. and S.Maddock (2009), "Sketch-based Facial Animation" (poster, refereed), *Facial Analysis and Animation*, One-Day BMVA Symposium, The School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton St, Edinburgh, UK, June 10th, 2009.

Gunnarsson, O. (2008), "Sketching Faces", Presentation at Data Modelling Workshop Symposium, The University of Sheffield, March, 2008.

Gunnarsson, O. (2008), "Sketching Faces", Presentation at VizNet Annual Workshop, Newcastle University, Culture Lab, January, 2008.

Gunnarsson, O. (2008), "Sketching Faces" (poster), Game Republic Student Showcase, First Prize in the Games Technology category, Bradford, 2008, First Prize.

Gunnarsson, O. and S.Maddock (2007), "Sketching Faces" (invited talk), Association of Chief Police Officers Facial Imaging Working Group, September, 2007, Chancellors Hotel, Manchester

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is an ever-growing need for efficient 3-dimensional object modelling in areas such as media, entertainment, gaming, architecture, and engineering. Traditional commercial software requires skillful labour to produce a range of editable meshes through low-level modelling and deformation techniques. This thesis focuses on the modelling of face models, which have a particularly complex structure. Creating faces is therefore a time consuming process, which relies on extensive modelling experience and anatomical knowledge to produce realistic and usable face models.

In recent years, both researchers and industry have introduced methods to make the face modelling process easier and more efficient for the user, with high-level tools based on prior knowledge and predetermined rules. The prior knowledge can range from a simple collection of model parts in the form of morph targets to a probabilistic generative model trained on face data [ABHS06]. In this thesis, a modelling process based on sketching is investigated.

More powerful 3D accelerated graphics hardware in tablet computers has given rise to a growing interest in 3D sketch-based modelling interfaces where they can typically be classified as gestural interfaces, or reconstructional interfaces. Gestural interfaces mimic a traditional modelling software where an object is created in steps using modelling operations in the form of simple gestures [ZHH96]. A desirable property of reconstructional systems is the ability to completely mask the underlying mechanism needed to create a 3-dimensional object. This allows a user to interact with the interface as he would a piece of paper where he draws

an object at its final stage, and the system tries to infer the meaning behind the sketched strokes and consequently reconstructs a 3-dimensional mesh that fits the strokes. Reconstructional interfaces can be further divided into free-form based methods such as Teddy [IMT99], and template based methods which are of particular interest for this work. They describe predetermined object classes using templates where a single template describes the features of an object class, and how the features can be reconstructed and assembled to build a complete 3D structure. Faces can be thought of as a class of templates as they all share the same facial features. A face template would therefore describe how to create a range of different features to produce different individuals.

Template based sketching interfaces need to interpret the meaning of any sketched strokes through classification where ambiguity plays a major role. A single arbitrary stroke is often hard to classify as it can mean any number of things. For example, a simple curve meant to represent an eyebrow can equally be interpreted as the upper part of an eye or lip. This ambiguity can be reduced by sketching more than one stroke where the strokes' relative context aids the classification (such as [SvdP06]), or sketching a single stroke in relation to an existing scene. Another ambiguity issue lies in the fact that the sketching plane is 2-dimensional. The third dimension representing the depth is missing and therefore has to be determined where the solution space is infinite. Applying restrictions and assumptions in order to map a 2-dimensional sketched point to a 3-dimensional coordinate is one of the key elements in any sketch-based interface.

1.1 Aims

The main aim of this thesis is to investigate the use of a template based sketching interface for face modelling by employing a generative model trained on 3D face meshes to act as a face template. This requires investigating ways to represent the training data in a low dimensional format which captures facial features from a sketch perspective. The nature of this low dimensional representation will determine the level of sketchability, i.e. what features can be sketched, and how accurately they can reflect the user's strokes.

In driving the modelling process with sketched strokes, a key element is to develop an efficient and accurate approach to map the strokes to this sketch-like representation. The goal is to adopt a what-you-sketch-is-what-you-get (WYSIWYG) sketching approach, where the user does not set any parameters, change modes or sketch additional strokes to aid the modelling process. Neither does the user have to specify which facial feature he is going to sketch. Instead, every stroke is treated as a new shape for a facial feature, where the system automatically determines which feature the stroke applies to.

The process of preparing face models as training data includes gathering already standardised faces as well as developing a framework to register unstandardised data such as laser scanned faces. Standardising face models offers the possibility of producing a novel data set without relying on external databases. Alternatively it can be used to incorporate novel face models into an existing data set with the purpose of expanding the variety of features.

Using a training set made up of facial poses instead of models of different individuals could offer the possibility of producing key poses through sketching. Some recent work involves posing 3D face models by sketching facial expressions [CJ06; LCXS07; SMND08]. This thesis investigates how the modelling approach can be adopted to create a sketch-based animation system by using a training set consisting of different facial expressions and visemes, where a viseme generally corresponds to the temporal midpoint of the acoustic segment of a phoneme in a particular language or dialect. This poses new problems as the sketched features in the modelling approach are always in the neutral pose (mouth closed, eyes open), but they can take different shapes in the animation approach.

1.2 Main contributions

The novel contributions of this thesis are:

- A registration method to standardise laser scanned models. A Radial Basis Function Network uses a small set of manually marked feature points to deform a generic model to approximate the scans. The smooth, approximated models are then registered to fit the scans accurately using depth maps.

- A statistical method to map sketched strokes to Feature Point (FP) clouds where each FP labels a facial feature. A Gaussian mixture model is fitted on the FPs where a single mixture component attaches itself to a FP cloud. Mapping strokes to the FPs is achieved by calculating the responsibility of each component for the stroke point coordinates. A sketch-based modelling method using this technique was published as [GM08], but is used in this thesis in Chapter 5 for the purposes of posing face models. This work has also been demonstrated at a Game Republic Student Showcase in Bradford, 2008, where it won first prize in the Games Technology category.

- A heuristic grading method for mapping sketched strokes to a contour cloud for face models. The method uses a scoring function based on weighted metrics which determine a stroke's context and preserve the continuity of the feature's vertex structure. This method could be generalised to provide a mapping from sketched strokes to any Non-Photorealistic Rendering (NPR) data set describing different classes of objects.

- A Hidden Markov Model (HMM) method for mapping sketched strokes to a contour cloud where the probability matrices are based on the metrics used in the heuristic approach. An additional continuity metric penalises large jumps between contours by measuring the vertex path length between two contours using a regionally limited version of Dijkstra's algorithm. The strokes are treated as a sequence of observations which is used to find the hidden (latent) states representing the contours, using the Viterbi algorithm. Similarly to the heuristic approach, this technique could be generalised to handle other objects than face models.

- A statistical mapping technique to deform a mesh using a small set of feature points. This is a common issue in computer animation where labelled markers are used to drive the face model's performance. By definition, a low dimensional feature vector can only represent a small portion of a high dimensional face mesh. Geometrical methods try to capture the remaining areas using blending functions which are not guaranteed to produce natural results, and fail to capture the mesh details. Using a hierarchical statistical

model trained on known poses, a more satisfactory mapping is achieved. This statistical mapping is adapted to produce correlation between face segments. This work was presented in [GS09].

1.3 Chapter overview

Chapter 2, *Background: Modelling and Sketching Techniques*, contains an overview of methods to create faces 3-dimensional face models in particular, and sketch-based modelling interfaces. 3D face models can be created through numerous approaches such as traditional modelling techniques and commercial packages, by recording real subjects, blending morph targets, or using a generative model. Sketching interfaces typically fall into one of two categories: Gestural interfaces, and reconstructional interfaces. The chapter focuses on the latter one.

Chapter 3, *Preparing Training Data*, describes two alternatives to gathering training data. The first option is to use a generative model to create a standardised set of faces The second option is to manually standardise a set of laser scanned models by replicating them using a standardised model. The advantages and disadvantages of both methods are discussed.

Chapter 4, *Modelling Faces by Sketching*, introduces techniques to map 2-dimensional strokes to 3-dimensional face data, whereby the act of sketching drives the modelling process. This involves using an inverse NPR method to create a sketch-like representation of known faces, where it is mapped to sketched strokes through a heuristic grading approach, or alternatively using Hidden Markov Models (HMMs). The maths and methods behind the generative models employed in this thesis are explained as well as how they are applied to the training set to perform novel face generation and feature correlation. The inverse NPR approach is justified, and examples of successfully generated models using simple strokes are presented. The technique presented in this chapter is evaluated using novice users where they were asked to perform two tasks involving sketching a model with specific facial features.

Chapter 5, *Posing Faces by Sketching*, uses a similar approach to the one provided in Chapter 4 to pose an arbitrary face model, generating different ex-

pressions and visemes, albeit using a simplified training set and mapping. The poses serve as keyframes used to create an animation sequence.

Chapter 6, *Conclusions*, presents the conclusions and directions for future work.

Appendix A, *Polygon File Formats*, contains information detailing the structure of the laser scan data, and how it is stored in a simpler, more efficient format in this thesis.

Appendix B, *Calling Matlab from C++*, explains how to use the vast number of vector based built into Matlab within a C++ program through an external Matlab engine.

Chapter 2

Background: Modelling and Sketching Techniques

It is important for this project to understand what techniques are available today to create 3D faces, and how they can be combined with sketch-based methods to derive at an appropriate solution to sketching 3D faces. Section 2.1 goes through methods for creating faces and determines what we can learn from that to aid with the sketching process. Section 2.2 gives an overview of sketch-based interfaces, focusing on the work most relevant to us and how it links to methods for creating face models. We conclude the chapter with Section 2.3 which details research methods that offer sketching capabilities to create or manipulate 3D faces.

2.1 Creating faces

Techniques available to create faces on a computer are numerous and choosing the appropriate one often depends on what the goal is, and what resources are available. In many cases a combination of methods are needed to deliver the desired outcome. The range of methods can be thought of as a spectrum, where at one end are methods that are primarily automatic such as 3D laser scanners where the subject is physically recorded. At the other end are methods that require a lot of manual labour, for instance creating a face using tools available in commercial modelling software. While the latter approach can generate very

efficient and flexible models, albeit not necessarily anatomically accurate, it requires expensive software and extensive modelling skills. The former method creates very accurate models, but it can only generate faces that physically exist, requires expensive equipment, and the models have to be cleaned up and adapted to gain the usability and flexibility of a custom made model.

Substantial amount of research has gone into creating applications that lie between these two extremes where they try to capture some automatic-based aspects to make the application easy to use for a novice user, while maintaining the flexibility of a manual-based solution. This is accomplished by limiting the application scope to faces only, and utilising prior knowledge of faces to create high level tools. We will discuss the range of methods by categorising them into three topics where the main focus will be on the last one:

- Modelling and editing faces
- Capturing faces automatically
- Building faces from prior knowledge

2.1.1 Modelling and editing faces

A 3D face can be modelled or edited using any 3D commercial software such as Maya, 3D Studio Max or Lightwave. The faces are generally created using patch surfaces, quad polygon modelling with subdivision smoothing, or non-uniform rational B-splines (NURBS) surfaces [Wat99]. It is important when modelling the face to have the polygons or patches flow naturally based on the human face structure and muscle shape (see Figure 2.1). It makes sure the face mesh is optimised with regards to the number of total polygons, and facial expressions look more natural [FLS+07]. Polygon modelling (see Figure 2.2) dates back to Parke [Par72] as one of the earliest attempts to digitise a face into a 3D model.

Similarly, existing models can be altered by moving the patch control points, polygon vertices, or by applying deformation techniques such as Free-Form deformation (FFD) [SP86]. Generally face models are stored as a set of vertices

and their polygon connectivity information. Although using this kind of software can be used to generate any model, it is time consuming, and it requires rigorous training in using intricate tools and operations which in most cases are not intuitive.

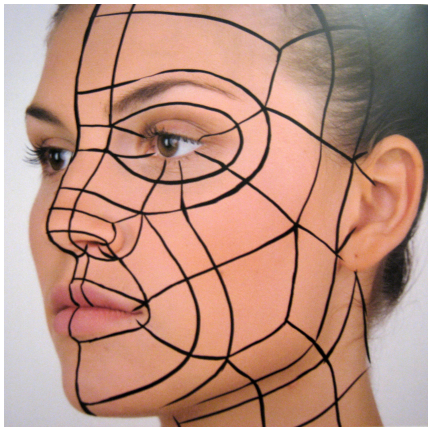


Figure 2.1: Modelled polygons should follow natural lines on the face (waiting for permission to use image from Ballistic Publishing [FLS+07])

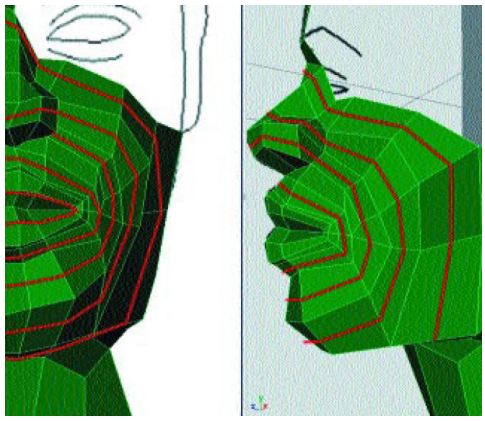


Figure 2.2: Polygon modelling following the natural lines and loops based on muscle movement (screenshots from www.computerarts.co.uk)

Z-Brush[1] and Autodesk Mudbox[2] take a different approach to modelling and editing 3D objects that is more intuitive from an artistic point of view. The user generally works with an existing object or a suitable primitive, and uses a brush tool directly on the object to edit or add to it. The main brush operations are pulling/adding and pushing/removing. The brush size can vary to apply to different tasks. A large brush with pull/add is used to construct large sections like a neck, whereas a small brush is useful to add details such as pulling out eyebrows, or using push to create wrinkles and other details. Figure 2.3 shows a popular method of creating a 3D head from scratch using Z-Brush taken from 3DLinks[3]. In step 1 the user has created a 3D sphere and started pulling out

[1]http://www.pixologic.com
[2]http://www.autodesk.com
[3]http://www.3dlinks.com

some basic facial features. Step 2 and 3 continue to pull out and form the main features until the user is satisfied with the overall shape. In step 4 the model complexity (vertex count) is increased and smoothed to allow finer details to be modelled.

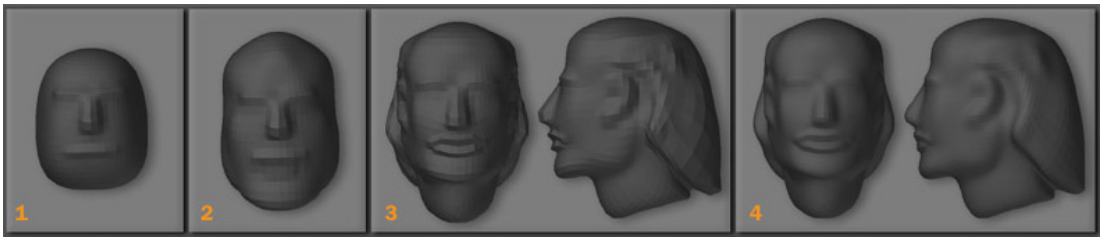


Figure 2.3: Modelling a head using Z-Brush (screenshots from Z-Brush [1])

Z-Brush and Mudbox are versatile and powerful tools, making it easy to tweak and add realistic details to face meshes. However, it requires a great deal of skill to create an arbitrary face or to make substantial changes to facial features. Since the process is unconstrained and free-hand it offers no guarantee of anatomical correctness.

Poser[4] also offers push/pull editing tools to create new morph targets by changing existing faces. Again the brush size can be changed to alter the size of affected area (see Figures 2.4 and 2.5). The area affected by the brush is shown before it is applied where red specifies the maximum influence (push/pull). Figure 2.4 shows how the cheek bones can be altered by applying a medium sized brush and then mirroring the effect. Figure 2.5 shows how to paint a crease by using a small brush. This technique is useful to make small changes to a face but it is not suitable for making large changes or constructing some features as it does not offer addition/removal of vertices.

N-Sided Quidam[5] has sculpting tools that build on the same brush concept. The user selects features from a library and then is able to edit them by moving polygons around where the polygons affected are determined by the brush size,

[4]http://www.smithmicro.com
[5]http://www.n-sided.com

type, and location.

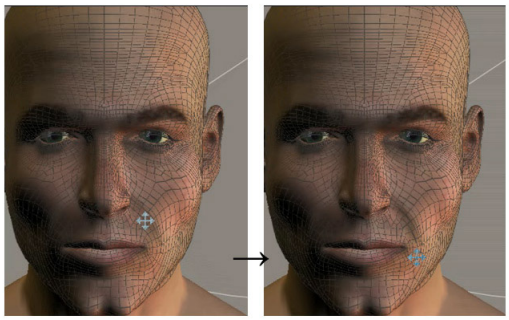


Figure 2.4: Large brush size to alter cheek (screenshots from [4])

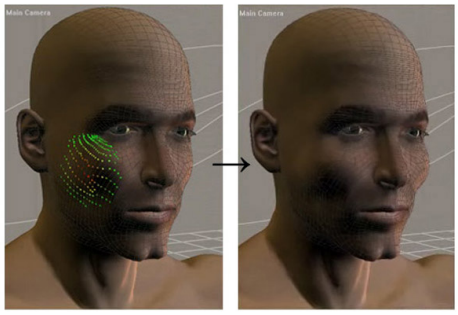


Figure 2.5: Small brush size to paint a crease (screenshots from [4]

Blacksmith3D Suite[6] differs from the other brush-based modelling interfaces. The affected area is painted first with a brush tool (see painting the nose in Figure 2.6), and then a number of transformations can be applied to the painted region. They include moving the selection in the plane of the viewport (Figure 2.7), extrusion, rotating, scaling, bulging and flattening.

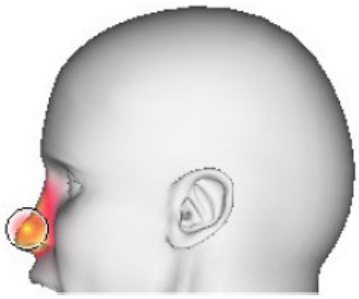


Figure 2.6: Painting a selected area (screenshot from [6])

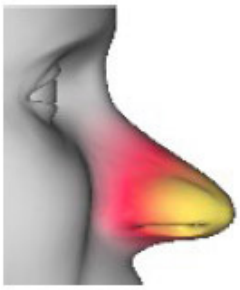


Figure 2.7: Transforming selected area (screenshot from [6])

[6]http://www.blacksmith3d.com

Nealen et al [NSACO05] offer contour sketching to create creases, ridges and ravines on any surface using Laplacian transformations. Laplacian coordinates are a simple form of differential coordinates where a vertex is represented by the difference to the centroid of its neighbours. By rotating the Laplacians in the region of the sketched contour by a specific angle with respect to an angle defined by a segmentation around the contour, a ravine can be created to add a cheekbone on a face model (see Figure 2.8). Additionally it is possible to alter features using silhouette sketching. This is done by first specifying the area of interest on the surface. Then after finding the original silhouette, a new desired silhouette is sketched to deform the object, where the sketched silhouette is interpreted as linear constraints when resolving the system to obtain the updated model (see Figure 2.9).

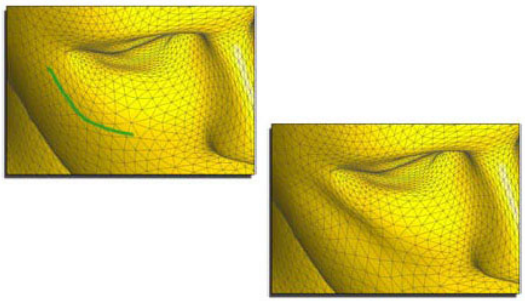


Figure 2.8: Sketching a strong cheekbone using a contour (image courtesy of Nealen and Alexa [NSACO05])

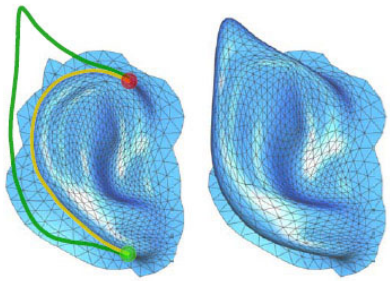


Figure 2.9: Silhouette sketching to alter shapes (image courtesy of Nealen and Alexa [NSACO05])

2.1.2 Acquiring facial geometry from subjects

Faces can be created by transferring information directly from physical entities such as a real person, photograph or sculpture. This is done by measuring the source either manually, semi-automatically or automatically. Digitisation is the use of a 3D locator device and involves measuring the face by marking points by actually touching it. That introduces a problem because flesh is a flexible

tissue and can deform when touched resulting in an unwanted variance in data accuracy.

In contrast, photogrammetry involves measuring the face without touching it. Instead it uses one or more pictures to construct a 3D head. Parke [Par72] drew a grid on a real face and took two orthogonal photographs from front and side which was then used to match up vertices to calculate the 3D coordinates. Akimoto et al. [ASW93] and Ip and Yin [and96] offer automation by identifying typical facial features on photographs and adapt a generic model to match the position and shape of the features. Pighin [PHL+98] relies on manually labelling a series of photographs with feature points and then using scattered data interpolation to warp a generic face to fit the desired face. The shape is further refined using correspondence curves between facial vertices and image coordinates. Blanz [BV99] uses a database of laser scanned heads to construct a statistical face model using a morphing function as a linear combination of the 3D scans. Constructing an arbitrary model using a photograph is then a mathematical optimisation problem.

The face can be measured automatically using 3D laser scanning where it is stored as a 3D point cloud. The cloud is very dense and tends to suffers at concave areas and the polar regions. The final stage generally involves cleaning up unwanted artefacts and polygonising the point cloud. Laser scanned data is used in a number of applications to achieve realistic models and animation [NLE+99; KHYS02; BPV06]. Issues concerning laser scanned faces and a proposed process of standardising is discussed in Chapter 3.

Williams [Wil90] handles missing data by using recursive blur filtering to create a smooth surface where the missing data is replaced with smooth neighbourhood estimates. Lee et al. [LTW95] fits the scanned data to a generic mesh using determined feature points to get a clean mesh of managable detail to be able to animate it efficiently.

Ypsilos et al [YHR04] and Zhang et al[ZSCS04] measure a face automatically by projecting a lighting pattern on the face and recording it using multiple cameras to calculate the point cloud, but as with laser scanning they require a post process to clean up the scanned data and make it usable for a practical application.

Dimensional Imaging[7] uses Passive Stereo Photogrammetry to automatically generate a 3D model using 6 digital cameras where 3 (2 monochrome and 1 colour) are located on either side of the face. Two range maps are created using the stereo information obtained from the corresponding monochrome cameras on the left and right side. Each range map is first re-projected onto the image co-ordinates of its associated colour image. They are then re-projected and merged into a single 3D point cloud which is polygonised [UGB06].

A non-existing face cannot be created using these methods unless a physical sculpture is created first using e.g. clay or plaster which is then recorded using the scanning methods above.

2.1.3 Building faces from prior knowledge

The most common approach to creating high level modelling applications is to limit the application's scope to a single class of objects, and supply it with knowledge of how these objects look like, and how it can be used to create new objects. It is important the prior knowledge can be tapped into through an intuitive and flexible interface to allow a user to quickly and accurately create the variation of the object he is after.

2.1.3.1 Building from parts

A straightforward way of building a face from prior knowledge is to store a number of individual facial features in a library and blend them in different ways to create new faces. This is referred to as a facial composite (see Figure 2.10). Facial composites are commonly used in law enforcement agencies where a suspect is created from a witness description. EFIT/3D EFIT[8], PROFit[9] and Identi-Kit[10]denti-Kit Solutions. http://www.identikit.net are commercial packages widely used by police, security organisations, and researchers [BPK00; NBHN02]. These applications can only generate 2D faces (with the exception of 3D EFIT), but 3D

[7]http://www.di3d.com/

[8]Aspley Identifying Solutions. http://www.efit.co.uk

[9]ABM. http://www.abm-uk.com

[10]I

applications such as Quidam and Poser store a range of 3D features in the form of localised morph targets that can be blended to create a novel face, and then refined in 3D using brush tools.

Wells et al [WCO05] point out several research documents that show conclusively that composite systems give poor results in terms of creating an accurate likeness of a suspect. They show that the process of building the composites can harm the builder's memory by decreasing the chances of him identifying the original face later. They maintain, along with [Bad79; BM96; Fro02], that the reason for this is that the composite systems are primarily a feature-based approach. The user selects individual facial features that combined give a representation of a face. However, people are believed to perceive faces holistically, remembering shapes and spatial configuration between facial features as opposed to remembering isolated features [SBOR07**?**]. Essentially, people are not good at remembering and describing particular facial features of another person, even close friends, but they would instantly recognise a familiar face if they saw it on the screen. This idea is used in EvoFIT [Fro02] which is detailed in the next section, and has been incorporated into EFIT-V. Furthermore, it has been found that caricatured versions of faces can improve the recognition performance [MK92; FBR$^+$07].

2.1.3.2 Generative models

An increasingly popular method is to fit a statistical model on a data set consisting of existing face data, often referred to as prior knowledge, and then use the model's generative properties to produce new faces. This can apply to whole faces or parts of a segmented face based on the main facial features, after which the generated parts are blended with the other segments. In facial animation this is commonly known as performance-driven models [Wil90; CB08], where data is gathered from an actor playing out different facial movements, and can serve as the prior knowledge in a probabilistic framework [LCXS07].

Principal Components Analysis (PCA) is the most popular statistical method as it is robust and easy to use. It forms a low-dimensional (decorrelated) representation of the original data made up of orthogonal eigenvectors and eigenvalues,

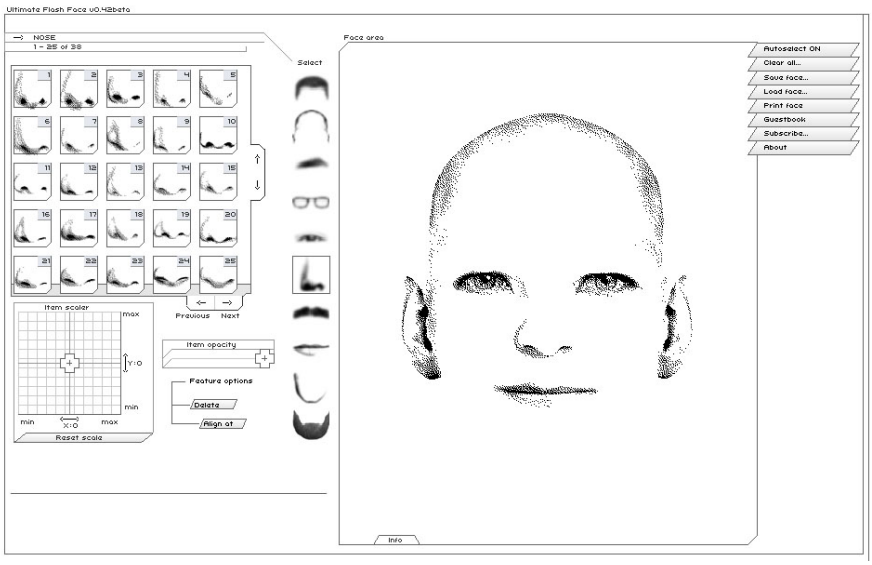


Figure 2.10: Building a face from parts (from Ultimate Flash Face, flash-face.ctapt.de)

often referred to as eigenfaces and weights respectfully where the eigenfaces with the highest weights make up a face space [MT91]. A linear combination of eigenfaces with a unique set of weights extracts a new face from the face space.

Frowd [Fro02] attempts to overcome some of the issues that plague composite systems with his software EvoFIT. He employs a holistic face model based on PCA with a genetic algorithm. The user is presented with a face with random characteristics and then through user selection, new faces free of unrealistic features are evolved. This way the user iteratively closes in on the desired target. It uses control points to morph between features in order to generate different facial shapes, and studies at the time showed it had significant benefits over traditional composite systems. Automating this process can be used to populate a world with virtual actors. Albin-Clark et al [ACH09] automatically evolve new faces from existing ones until their features differ enough to be considered distinctly unique among the population. This similarity measure is based making sure the shared number of normals does not exceed a certain threshold value.

DeCarlo et al [DMS98] randomises statistics based on anthropometric landmarks and measurements to generate plausible faces by fitting a prototype face model to match the randomly selected values. Similarly Rudomin et al [RBC02] can generate plausible face models but instead of using anthropometric measurements, they use a PCA face space based on MPEG-4 feature points marked on a set of face models. They generate new individuals by randomising the eigenvalue weights that specify how much each eigenface contributes to the reconstructed face.

Conformation parameters can be classified and used to control the proportion of individual facial features and overall shape as suggested by Parke and Waters [PW96]. Using these parameters, a 3D face can acquire desired properties that can e.g. further define differences between males and females, thick lips versus thin lips.

Blanz and Vetter [BV99] create a morphable model of faces using PCA where the face space is made up of a shape space, and a texture space, and a probability distribution is estimated for the eigenvalue weights to give realistic values. Conformation parameters are manually assigned with labels, and their corresponding weighted sums are found which can be added or subtracted from an arbitrary face model to acquire a particular property. Each of their 128 principal components is one standard deviation in magnitude. Navigating along these axes, new faces can be generated by adding different deviations to the average face located at the origin. A model can be generated from a 2D image, or registered from a scanned 3D head by defining a set of parameters for the input and the target model, and minimising a cost function which is the euclidean distance between the two sets of parameters. In the case of creating a 3D face from an image, the statistical model is constrained by finding the eigenvalue coefficients, and rendering parameters with maximum posterior probability given the input image.

FaceGen[11] is a commercially available software that is similar to the method by Blanz and Vetter [BV99]. Chen and Fels [CF04] proposed an alternative way of navigating through a space of 3D faces from FaceGen. They base their navigation on Adobe's approach to selecting colours, a colour wheel and sliders. The users

[11]FaceGen. Singular Inversion

navigate by iteratively clicking on a face on the face-wheel to get a rough likeness, and then tweak it using the sliders to get a more precise approximation (see Figure 2.11).

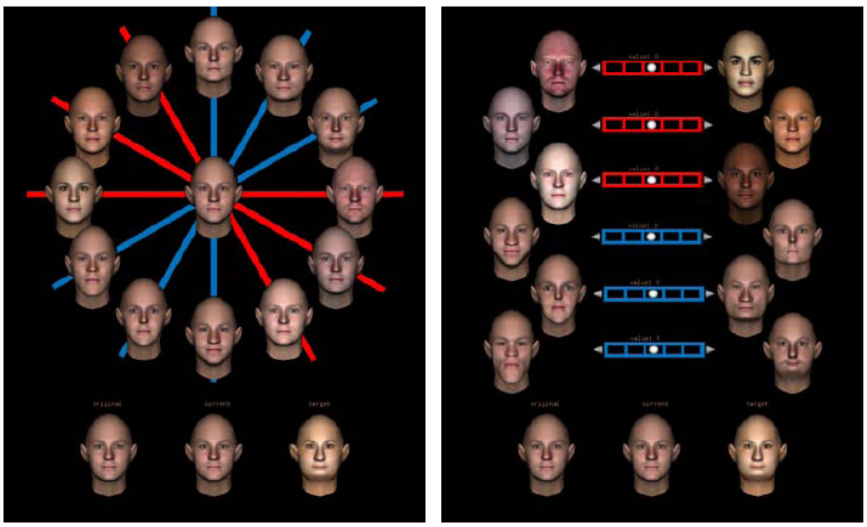


Figure 2.11: Exploring face gradients inspired by a colour wheel (image courtesy of Fels [CF04])

Blanz et al [ABHS06] extend the morphable model [BV99], and combine with their own method of replacing faces in images [BSVS04], to provide a tool to create faces from incomplete witness description. The user starts with the average face which is segmented based on the main facial features (see Figure 2.12). The user can select features from a database whereby a selection mask determines which segments are updated. Alternatively, features can be adjusted by navigating the gradients of grouped conformation parameters, or what they call facial attributes using sliders. A correlation between facial features is learned from the existing models, and the assignment of facial attributes is improved from their earlier work. It captures correlations between shape and texture, and constraints ensure specific attributes are maintained while others are changed where they would normally correlate otherwise. For instance, traditionally if the eyes are enlarged, the whole model becomes more feminine, but the constraints make sure the model keeps its masculinity. The feature correlation is useful when the witness description is

incomplete, as the unspecified features are automatically filled in to give the most plausible outcome based on the population of learned faces.

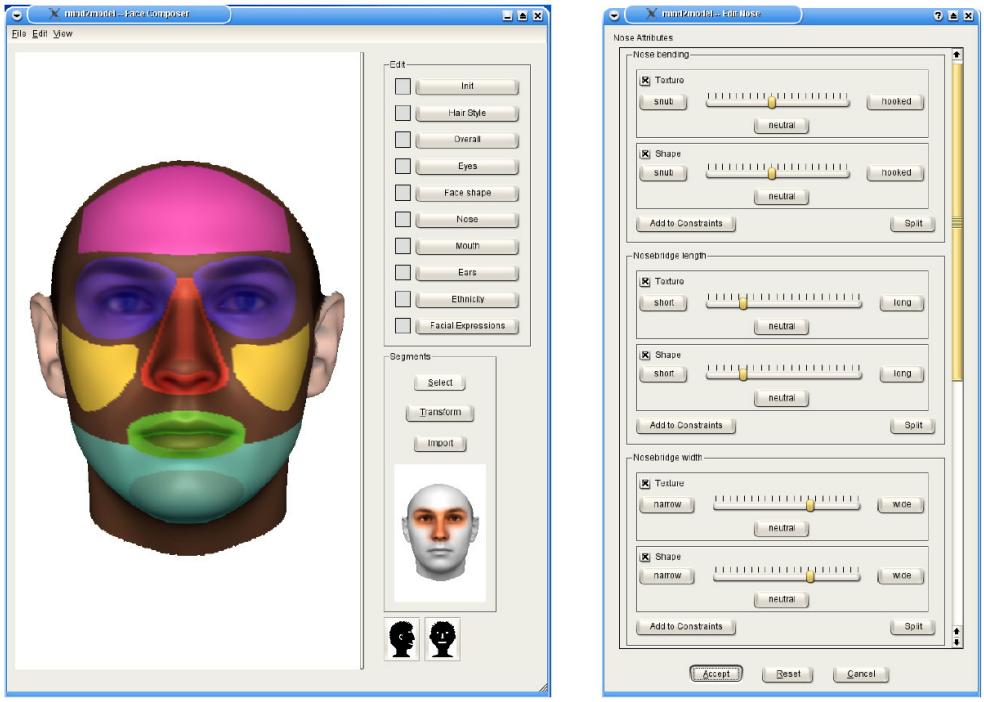


Figure 2.12: Facial regions and conformation parameters (image courtesy of Blanz [ABHS06])

Mohammed et al [MPK09] uses the generative properties of a factor analysis mixture model in conjunction with non-stationary image quilting to create consistent and detailed facial images. Random individuals can be sampled using the generative model, but the images tend to lack detail, particularly when using reduced number of factors to limit the dimensionality. The quilting algorithm can produce detailed images by segmenting existing face images into 9 by 9 patches, and creating a new face by selecting the top-left patch randomly, and then assigning the remaining patches one by one where each patch must be visually consistent with the patches above and to the left. However, this will produce faces with locally consistent features, but are not globally consistent. They find that realistic faces are achieved by combining the global aspects of the generative

model, with the details of the locally assigned image patches. A patch being processed must take into account visual consistency of its surrounding patches as before, and its global content in the form of a weighted combination. The generative model can be used to fill in incomplete parts of the face. This could be useful when identifying a partially obscured face of a suspect caught on camera. The system proposes a range of the most plausible complete faces to a witness by filling in the missing parts through maximum likelihood. When looking for a suspect in a police investigation, a vast number of photographs in a mugshot database are compared to a pictorial evidence in order to find a match. Often this pictorial evidence is a sketch of the suspect according to a witness description. Creating 2D portrait images from 2D sketches can minimise the difference in the comparison [TW04; CLR$^+$04; XCZL08; WD09].

Novel faces can be generated by ageing existing models. Studies into facial ageing dates back to 1917, and has today become particularly useful in the field of facial recognition [RCB09]. Patterson [PSAR07] compares face models that have been aged using active-appearance models to an age progression drawn by a forensic artist and were found to be similar based on a small user study. Shaw et al [RMW74] identified two invariant mathematical transformations for the growth of facial contours seen from the profile view. The cardioidal strain transformation, describing how the face contours stretch outwards, was extended by Mark and Todd [MT83] into 3D by expressing the transformation using spherical coordinates, and applied to the entire surface of the face.

Scherbaum et al [SSSB07] alters the age of people in photographs based on their own method of exchanging faces in images [BSVS04], using the 3D morphable model fitted on a database of a range of teenagers and adults. Here, an ageing prediction is performed on the 3D model before rendering back into the image. RBF Support vector regression is used to learn a function mapping age to each sample face represented as a single vector. The ageing prediction is made by calculating a trajectory in the face space where the trajectory is parallel to the gradient of the age function. The FG-NET Ageing Database [FN] contains 2D images of faces where the subjects are recorded at different ages.

Combining the FG-NET database with the 3D morphable model, Park et al [PTJ08] show that adapting a target facial image to match its current age improves the rate of successful facial recognitions. The morphable model is used to create simplified 3D models (81 vertices) on the range of 2D images in the FG-NET database using its 68 marked feature points assigned to each face.

This approach of limiting the scope to a prior knowledge of a particular class of objects will become an important factor in the next section on sketching interfaces, as it determines what approach is the most suitable with regards to modelling realistic face models.

2.2 Sketching interfaces

Sketching on a computer screen shares a limitation with an artist sketching on a piece of paper. Here, sketching is essentially a way to describe a 3-dimensional object on a 2-dimensional surface, whereby the dimension that represents the depth information is not preserved. Instead the projection onto the 2-dimensional clipping plane creates the illusion of depth perceived by an intelligent viewer.

Computer-based sketching interfaces aim to do the reverse process of reconstructing 3-dimensional objects from a 2-dimensional sketch by finding this perceived depth. Therein lies the problem with all 2-dimensional sketching interfaces. Without any assumptions or prior knowledge, the depth value has infinite number of possible solutions when the 2D screen coordinates are projected into the 3D world coordinates. The way this ambiguity is dealt with is a fundamental part of the research with any sketch-based interface. The solution depends on what assumptions are made with regards to how the user perceives the depth of an arbitrary stroke in the interface.

Sketchpad (1963) was the first interactive, graphical user interface where it was possible to draw simple shapes using a light pen pointed at the screen, and a collection of command buttons for operations such as draw, move, and delete [Sut80]. The pen was used to specify locations, line end-points and angular lengths. Sketch-based interfaces have been gaining momentum in recent years

with the advent of tablet computers where a paper-like drawing tablet mimicking the feel of different kinds of paper has been specifically designed to serve as an inpu-device for cartoon drawing and editing and is [HADF+05].

Categorisation of sketch-based interfaces varies between papers, and some interfaces are hybrids of more than one method making it hard to conclusively favour one category over another. Company et al [PCN05], Igarashi et al [LDI06; Iga07], and Olsen et al [OSSJ09] discuss and classify a wide range of sketching interfaces. The grouping of sketch-based interfaces in this thesis is based on common terminology and classification found in these surveys. The interfaces are divided into two main categories: Gestural interfaces (2.2.1) and reconstructional interfaces 2.2.2.

Reconstructional systems can be further divided into Free-form based systems and Template-based systems. Some reconstructional interfaces use gestures as well to signify modes instead of using icons and menus. Modes are command states that affect how the sketched stroke is interpreted and is specified prior to sketching. Modes can also serve as direct functions, such as to delete an object by marking it with a gesture. In order to retain the pen and paper feel as much as possible, it is important to limit the number of modes used when sketching. Additionally this mode-switching can cause problems. If a user forgets to specify the desired mode then he ends up sketching spurious strokes or meaningless gestures [SL03].

The following sections give an overview of different sketching techniques, but importantly explain how template-based systems are best suited to the requirements of the work in this thesis, and hence is covered in more detail.

2.2.1 Gestural interfaces

In gestural systems the user builds the 3D objects in steps, using traditional modelling operations commonly found in commercial CAD packages such as extrusion, sweeping, lofting, blending surfaces etc. The difference is that using gestures is far more intuitive than using traditional CAD tools because they are generally more meaningful than command buttons. Additionally, the gesture can also indicate the shape and magnitude of the operation it uses.

The following example demonstrates the difference and added value of the sketch gestures: When creating a sweep (loft) object in a commercial software, the base shape and sweeping path are created first. Then the sweeping operation is applied to the shape followed by specifying the path involved. In a gestural interface, the sweeping gesture could also directly serve as the path the shape will sweep along to create the 3-dimensional object.

Eggli et al [EBE95] is an early example of this type of gestural modelling interface. It is used to create mechanical objects and free-form ruled surfaces. Through gestural commands, it is possible to create objects through extrusion, revolution and sweeping (lofting). Once an object has been created, other features such as holes can be sketched directly on the object. Sketched strokes are rarely perfect straight lines as required in mechanical drawings, the system cleans up the sketched strokes to make sure they are straight and line up with parallel strokes. Other useful features such as snapping and constrained angles are also supported. However, the set of gestural commands are limited, and the lack of cutting and boolean functionality limits the complexity of objects.

SKETCH [ZHH96] came with a more extensive set of gestures and became a milestone to which subsequent sketching interfaces were compared with. Kim and Kim [KK06] attempted to improve the user interaction with the system by simplifying the gestures into single-stroke pen markings. Figure 2.13 shows how to create a cylinder through gestures using their system. A circle is created on the working axis plane, followed by a gestural command for extrusion implied by drawing an arrow (a). The user can then specify the extrusion amount h with a picking action (b) to create the completed object.

Cherlin et al [CSSJ05] (see Figure 2.14) added rotational and cross sectional blending surfaces to create more complicated object, using few but well defined gestures. A free-form outline was drawn (black/purple line), followed by a sketched cross section describing the 3-dimensional nature of the object (red line). Additionally, it allowed deformation through sketching gestures such as

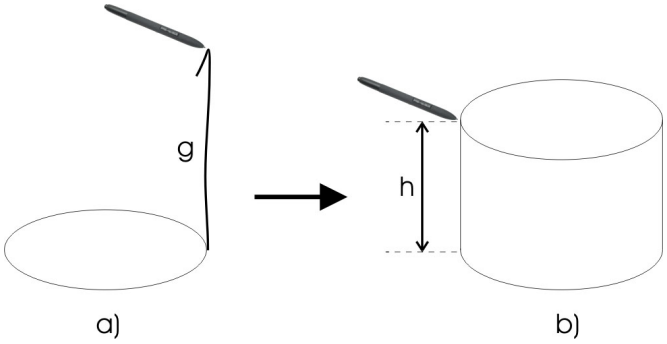


Figure 2.13: Cylinder created using a gestural interface (after [KK06]). Gesture g indicates an extrusion operation (a). The extrusion amount h for the cylinder is specified using a hover and pick action (b).

bending, and cross sectional oversketching.

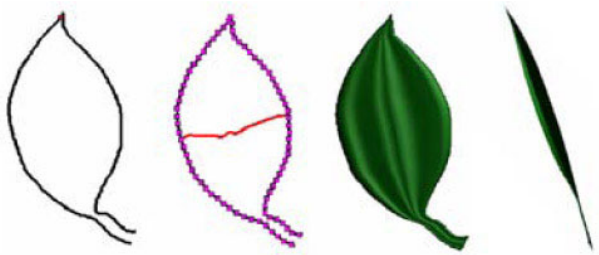


Figure 2.14: Blending surfaces (image courtesy of Samavati [CSSJ05])

Gestural interfaces are a useful enhancement to traditional modelling packages where objects are created through a series of low-level operations. Complex objects can be created, but the user is involved with choosing the modelling operations required to construct the objects from simple shapes. This is particularly challenging with objects of complex internal structures like faces. This aims of this thesis is to create faces in a sketching interface where the user does not need to know how the underlying polygonal structure is built, or what deformations and adjustments are needed to construct a particular feature. Therefore, gestural interfaces are not suitable for the purposes of the work outlined in this thesis.

2.2.2 Reconstructional interfaces

In gestural interfaces, objects are built incrementally using simple forms and gestures as we explained in the previous section. In contrast, reconstructional interfaces are able to create complicated 3D objects where the user has no knowledge of what process was used to make produce them.

The difference is clearly demonstrated by looking at how a reconstructional interface by Masry and Lipson [ML05] is used to create mechanical drawings (see Figure 2.15) which are commonly created with a series of extrusions applied to simple shapes, and therefore well suited to gestural interfaces such as Eggli et al [EBE95]. The user sketches the outline or shape of an object (left), and then the system reconstructs the object (middle) without the user knowing what methods are needed to create the polygon structure. When the object has been constructed, the user can examine the it from different angles (right) and make further additions or adjustments.

From this it can be concluded that reconstructional systems are closer to a pen and paper experience than gestural systems.

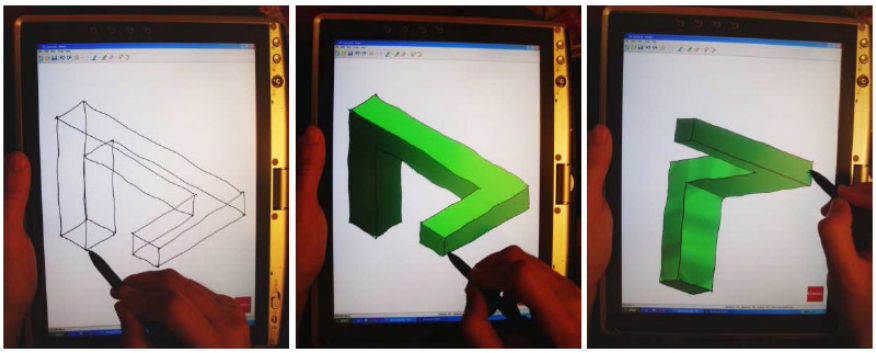


Figure 2.15: A reconstructional interface (image courtesy of Lipson and Masry [ML05]).

Reconstructional sketching systems can be further divided into two groups, free-form based systems (section 2.2.3) and template based systems (section 2.2.4). Free-form systems contain a set of geometric construction and transformation rules that when applied to a sketched stroke generate a free-form object.

Modes are used to decide which rule is active, where the mode is often set using sketched gestures. Given a sketched stroke and an active rule, the same reconstruction process is executed regardless of a potential intrinsic meaning behind a particularly shaped stroke, or its location. For example, in an inflation based system like Teddy [IMT99], a big circle with two smaller circles centred on the big circle, depicting eyes, will always be result in corresponding spheres of the implied circumference and location.

In a template-based system however, strokes are interpreted with regards to prior knowledge of templates that describe one or more classes of objects. Here, a stroke can serve a specific purpose based on its shape, and relative position to other strokes according to an interpretation laid out by the templates. A sketch recognition process determines how the sketched strokes map to known templates. The template controls the reconstruction process where the predetermined purpose of each stroke defines how it is used to generate a corresponding 3D object.

Template systems can therefore only create objects specifically described by the templates, while free-form systems can create any object that follows the reconstruction rules. Complicated objects are generally easier to describe with templates than rules. In free-form interfaces, complex structures are usually created using compound objects. Faces have a complex internal structure but are in essence a single class of objects as they all contain the same set of features. As we saw in Section 2.1.3, a successful high level approach to modelling faces is found by limiting the scope of the application to a single class of objects. We argue that a template scheme is therefore best suited to a sketch-based interface for faces.

2.2.3 Free-form based systems

Teddy [IMT99; IH03] was the first successful free-form interface and became an inspiration to other work based on the original technique. The user sketches a silhouette of an object, and the system creates the corresponding free-form object through inflation (see Figure 2.16). This is done by creating a polygon from the silhouette, applying Delauney triangulation and finding the chordal axis.

The depth problem is solved by assuming that the sketched object is situated at the origin and the vertices in the chordal axis are elevated in proportion to the distance to the polygon itself. This system supports gestural features such as cut and extrusion. Figure 2.17 shows how a extrusion is indicated with a closed stroke on an existing model (red circle). Then the extrude stroke is drawn (the one being drawn in the figure) and the depth issue is solved by assuming that the extruded form is perpendicular to the object's surface determined by the closed stroke. This system does not however allow any sharp features or sophisticated creases.

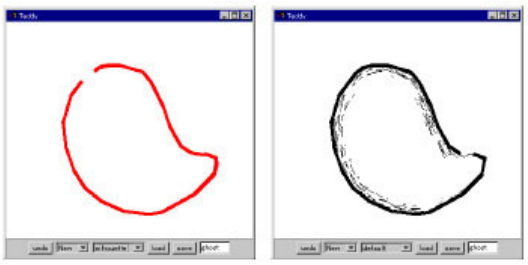


Figure 2.16: Teddy: Input stroke and the constructed 3D form (image courtesy of Igarashi [IMT99])

Figure 2.17: Sketching an arm using extrusion (image courtesy of Igarashi [IMT99])

Karpenko et al [KHR02] argued that the polygonal representation used in Teddy caused problems and instead introduced a method which used variational implicit surfaces. As a result a more complex object topology was possible. On top of that, objects can be merged described by a guidance stroke which illustrates the connecting surface and how it merges with the target objects. This is shown in Figure 2.18 where the black lines are the guidance strokes between the two objects. Oversketching is more intuitive when compared to Teddy where the user had to sketch two lines, a reference line and the target line. Here the user only has to draw a stroke near a silhouette and the object is then altered accordingly. Implicit functions are based on thin-plate interpolation and solved using scattered

data interpolation where the function is a weighted sum of radial basis functions [TO99]. When dealing with certain conditions, for instance when a guidance stroke is far away from a surface and therefore far outside the radius of the basis functions where their influence is minimal, unexpected interpolation results can occur. Implicit surfaces do not naturally support sharp edges and no attempt was made to support that feature. Tai et al [TZF04] addressed the issue of sharp edges with their implicit convolution model, and by specifying the cross-section profile of the sketched object. This added dimension can generate a larger variety of objects, including objects with semi-sharp corners.

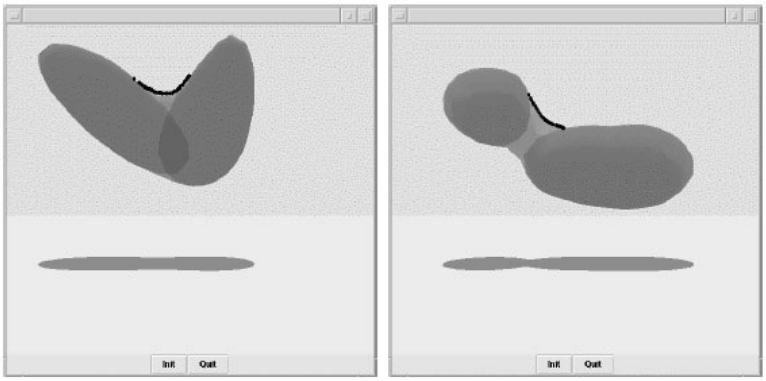


Figure 2.18: Merging with guidance strokes shown as black lines (image courtesy of Karpenko [KHR02])

Karpenko et al [KHR04] made further enhancements by proposing a way to overcome ambiguities concerning the depth problem. Because infinite solutions are possible, every system provides a method of choosing one solution from a specific viewpoint. This solution however is not necessarily the one the user is after. They attempted to overcome this by allowing the user to sketch an object from an initial viewpoint, and then alter the object's shape from a different viewpoint using oversketching (see Figure 2.19). The new stroke (green line) is projected on the epipolar lines (yellow lines) and the coordinates on the original stroke (red line) are updated to satisfy the new constraints. Similarly, Malik [Mal05] allows the user to alter sketched hair clusters by sketching strokes from

other viewpoints that push or deform the control curve for a particular cluster.

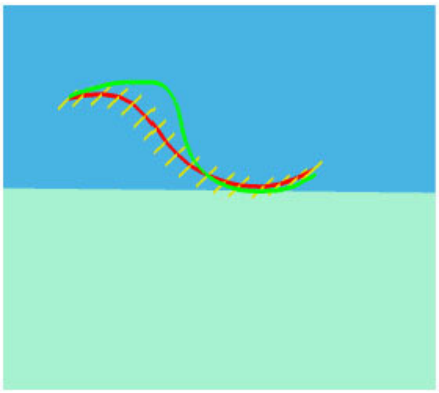


Figure 2.19: Curve edited from a different viewpoint using epipolar lines (yellow) (image courtesy of Karpenko [KHR04])

Zenka and Slavk [ZS03] extended the idea behind Teddy to create more detailed non-photorealistic (NPR) cartoon like characters and faces. The blob inflated objects serve as a 3D skeleton to which fuzzy lines can lie on and extend from. Figure 2.20 shows how this is used to sketch a bunny head. The user starts by sketching the blob skeleton (far left). Eyes and hairs are sketched on the skeleton object using lines that either lie on the surface or extend from it. The sketched lines are warped to reflect the current rotation. The user can refine the sketch from other angles if the warping is inaccurate which is used to generate a new estimate for the sketched output. Now the sketches can be rendered without the skeleton to give a view-independent cartoon objects. The implicit blob inflated skeleton does not support sharp edges, but the add-on silhouette and surface lines can be of any shape.

Wyvill et al [WFJ+05] combined inflated implicit models with linear sweeps and revolution to allow a bigger range of reconstructed objects. Karpenko et al [KH06] overcame the limitation of allowing only simple closed curves in free-form sketching. They deal with cusps and T-junctions (see Figure 2.21) by finding the hidden contours and cusp that have the highest probability shown as H in Figure

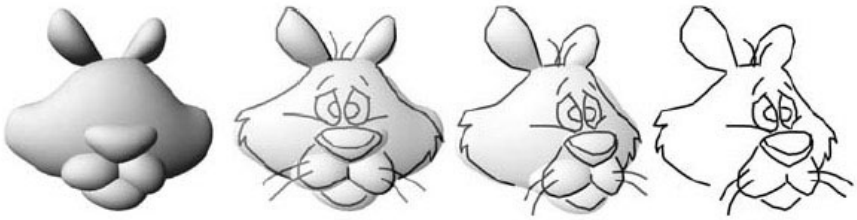


Figure 2.20: Silhouette and contours on blob surface (image courtesy of Slavik [ZS03])

2.22.

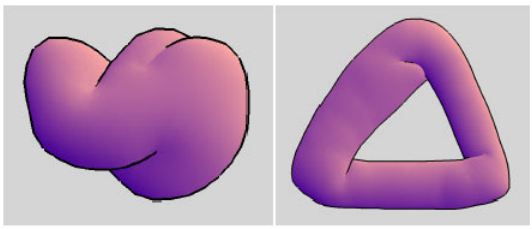


Figure 2.21: Cusps and T-junctions (image courtesy of Karpenko [KH06])

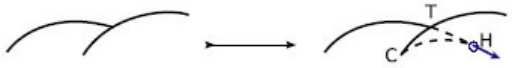


Figure 2.22: Finding the hidden cusp (image courtesy of Karpenko [KH06])

Mechanical drawings can be generated using free forms with orthogonal constraints as discussed earlier (see Figure 2.15). Masry and Lipson [ML05] offer an optimisation-based approach, assuming the sketch is an orthographic projection of a 3D object onto a 2D sketch plane with depth $z = 0$. To solve the depth problem, the axis system is determined by finding a vertex that best represents the angular distribution of the sketch using a histogram of the 2D angles of the strokes relative to the sketch plane. The assigned vertex has three connecting vertices, each determining an axis direction. The main vertex is assumed to lie on the sketching plane with zero depth, while the axis depths are found by minimising an optimisation function based on two assumptions about the axis system. The other edge nodes (endpoints) in the sketch are found by propagating depth values along a Maximum weight Spanning Tree (MST) that connects each vertex to the main vertex. To create 3D curves from the strokes, an optimisation function

relates the depth values for the stroke points to the depth of the corresponding stroke endpoints.

Malik [Mal05] developed an interface to model hairstyles. New clusters of hair can be added by sketching a control hair strand, shown as a green curve in Figure 2.23. Hair clusters originating directly from the scalp with no surrounding clusters are projected into the view plane where the depth is set to the depth where the strand touches the scalp. If there are surrounding hair clusters, the system assigns the 3D orientation for the new strand by averaging the orientation of nearby clusters.

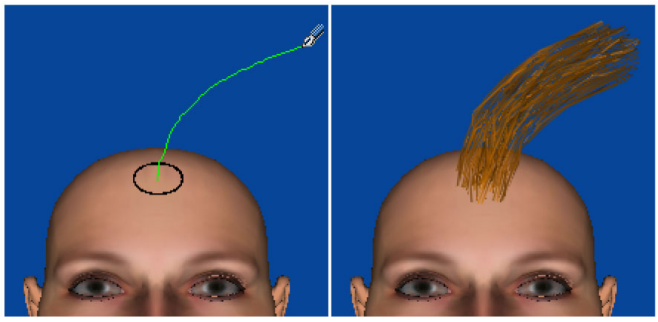


Figure 2.23: Implanting a hair cluster (image courtesy of Malik [Mal05])

Wither et al [WBC07] infer hair strand parameters from sketched strokes describing length, natural curvature (curliness), ellipticity (the shape of the strand's cross-section), and stiffness. The inflection points on the strokes are used to segment them where the parameters for each segment are calculated. These parameters are then applied to a physically-based hair modelling technique by Bertails et al [BAQ+05], where the parameters are used to model a circular helix (half turn) which represents a single strand segment. The user sketches on a head model or a photograph from a 2D side view. He uses simple strokes to define the scalp area, example strands, and volume (optional). The system uses this information to geometrically generate a full head of hair by creating a scalp object which is subject to triangular tessellation, where the triangles are then populated with

uniformly distributed hair particles from where the helix strands are grown.

Davis et al [DAC+03] uses 2D sketches of stick figures with a set of constraints to generate the most likely 3D human character poses. Figure 2.24 shows the sketched figures and the corresponding 3D poses. Unlikely poses are culled out where knowledge such as angle constraints, balance preference and coherence between neighbouring frames is used. After the culling there are typically still more than one estimated pose that satisfy the 2D sketch so system ranks them heuristically and assigns the most likely one. The user can select a different one if he finds it more suitable.

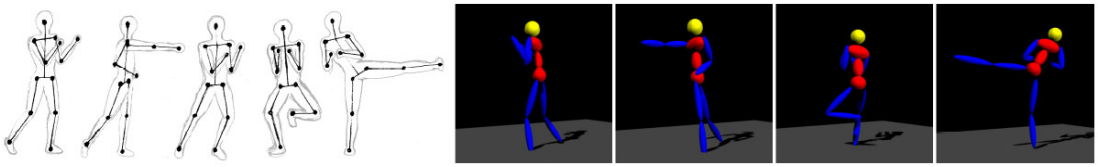


Figure 2.24: Estimating 3D poses from 2D sketches (image courtesy of Davis [DAC+03]).

Mao et al [MQW09] transfer a 2D sketch into a 3D posed character, but also create a skinned human 3D body as well that matches the pose (see Figure 2.25). The user sketches the stick figure with an arbitrary pose (left) where emphasised strokes are interpreted as being closer to the viewer in order to reduce the depth ambiguity. The user then sketches the skinned body and the pose and body proportion are used to morph a generic model to fit. A body fatness morph stage is performed based on predefined body cross-sections for individual parts and are estimated from the initial morph. This is followed by an automatic surface fitting to match the 3D template to the 2D sketch (middle). Simple local deformations are possible by sketching a suggestive contour on the body, and then a silhouette from a different viewpoint defining the new profile (right). The region of interest (ROI) is found based on the suggestive contour stroke, and a scaling influence for each vertex in the ROI is found which controls how far each vertex in the ROI is deformed. The influence is found by first examining the difference between

the projected source silhouette on the model, and the target sketched silhouette, where the scaling factor for the remaining vertices is found using Hermit spline interpolation. This method is able to produce a large number of posed 3D bodies, albeit without hands or feet, that would be difficult to model with a tool such as Teddy, and more time consuming with 3D Studio Max or Poser. However, there are limitations with regards to crossed limbs and viewpoints, and sketched local features because of low mesh density.

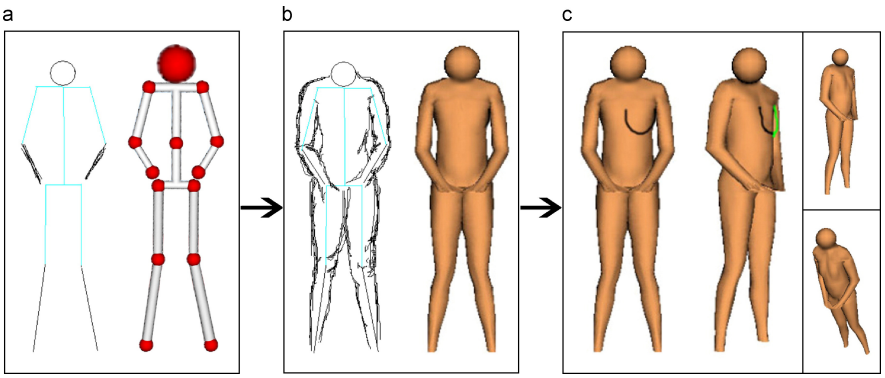


Figure 2.25: Sketch-based human body modelling and modification pipeline (image courtesy of Qin [MQW09]).

2.2.4 Template-based systems

There are a number of ways to represent, retrieve, and reconstruct class templates from a set of sketched strokes. Mitani et al [MSK00] use edge and node matching to create electronic gadgets. The templates are limited to objects with 6 faces of 4 sides, and mirror symmetry. This forces the user to sketch the gadgets in the same fashion as if the user was looking at the object from a fixed camera angle. Core parametric curves are found from bundles of strokes, and an edge graph is extracted and matched with the templates analysing the vertex coordinates. If a matching template is found, the 3D object is created using Coons patches based on the edges from each face, and sampled into a regular triangular mesh.

Cheon and Han [CH08] improve the edge based approach by using relational templates which use the relationship of neighbouring edges, or so called feature

vectors in the matching process. The process extracts the core curves from the sketch, finds the corresponding edge graph and match it with a template, and finally generates the 3D curves and uses them to reconstruct a 3D object with Coons patches. The user still has to sketch from a specific viewpoint, and the sketch has to be topologically identical to the template.

Yang et al [CYvdP05] similarly use 2D node and edges to describe classes of templates. Figure 2.26 shows how the strokes are mapped to a cup template. The expected locations for the template nodes (I,J,G,H) are found in the sketch using normalised coordinates based on the bounding box, where they search for correspondence within a certain radius shown as black ellipses surrounding the expected locations. Furthermore, curve feature vectors in the template and sketch are compared to determine if the edges are similar. A template consists of mandatory and optional parts. For the cup template, the handles and saucer are optional. Figure 2.27 shows an example of a cup sketch and the reconstructed object. More complex objects can be generated using a top view template in conjunction with the standard one to overcome the depth ambiguity. In this case the object is sketched first from the default view and if recognised the top view is sketched. Adding a new object class is time consuming as the procedural functions used to reconstruct 3D objects from the 2D strokes are defined manually for each template.

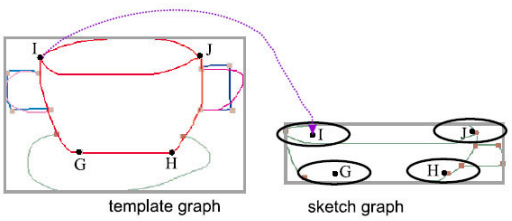


Figure 2.26: Finding expected locations for template nodes (image courtesy of Panne [CYvdP05])

Figure 2.27: Creating a cup using edge-based templates (image courtesy of Panne [CYvdP05])

Templates can consist of one or more class of objects where the sketch is used to look up the template/object that most closely satisfies the sketched strokes according to some criteria. Lee and Funkhouser [LF08] build 3D models by composing them from a number of different parts where each part can be considered a template. The template itself is made up of boundary contour images generated from 24 orthogonal viewpoints for the corresponding 3D object. A user sketched contour looks for the templates whose boundary contour images have the lowest sum of squared distance between the closest non-zero pixels [FKS+04]. The closest matches are presented to the user and the selected object is placed in the scene, where it is rotated, scaled and translated to fit the sketched stroke and lie on underlying objects (see Figure 2.28). An object can blend with previously sketched and retrieved object parts with polygon stitching.

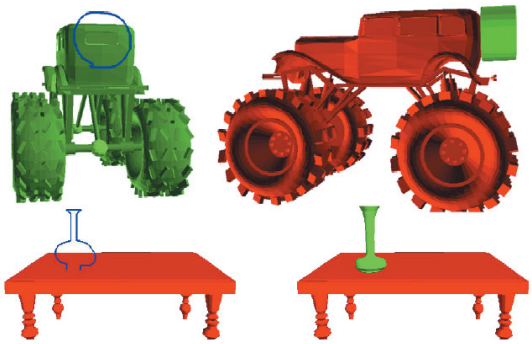


Figure 2.28: Sketch-based search for a part and scene placement (image courtesy of Funkhouser [LF08])

Instead of using the stroke to look up a template, it can be used to deform a generic template to fit the stroke. Murakawa et al [MYH+06] use this approach to customise ant models using a generic ant model as a starting point. The user selects the part of the model he wants to change, and then sketches the desired silhouette of the part from top, side, or front view. The model part is morphed by mapping control vertices on the generic part to sketched points. The difference creates movement vectors that affect the remaining vertices using either a parabolic or a linear weighting function. To help sketch a specific ant species, photographs can be fitted in the orthogonal views to enable tracing.

The aforementioned methods retrieve, and fit the templates using heuristic methods. That process can be handled in a probability framework where the templates are used as training data, and the fitting process utilises the generative properties of the statistical model. This echos what was discussed in section 2.1.3, where the same method can be used to learn different faces through a statistical mapping. This thesis argues that this approach is the most suitable for sketching faces as it combines the successful generative models for faces with template based sketching.

Sezgin et al [SD05] use Hidden Markov Models (HMM) to learn and classify classes of 2-dimensional diagrams and mechanical engineering drawings. HMM can detect sequential patterns in the data so the model can take into account stroke ordering and speed between users to improve the sketch recognition.

Kokai et al [KFS+07] build car templates from a set of polyline networks defined manually from 3D models of car meshes where internal features can be altered through sketching (see Figure 2.29). Deformation gradients map the polyline examples to a feature space of deformations where a new object is created through an inverse mapping of a linear interpolation of features. Principal Components Analysis (PCA) is calculated from the set of features and used to linearly blend a new model to enforce a restriction on the weighting coefficient, with an added regularisation term to satisfy additional constraints in the optimisation process. The model is segmented into parts (body, wheels, grill etc.) to allow local deformations that do not affect other parts. This segmentation simplifies each feature space which gives better PCA approximations due to its linear mapping. The sketched stroke is mapped to the closest corresponding points on the template where they form a corresponding curve in the 2D image plane. The curve is fitted using active contours and projected back into 3D world coordinates where the curve that minimises the optimisation function.

2.3 Sketching faces and face poses

This section covers any method that creates a new face model using sketching. Here, a new face model can either be a new individual with its own distinct fa-

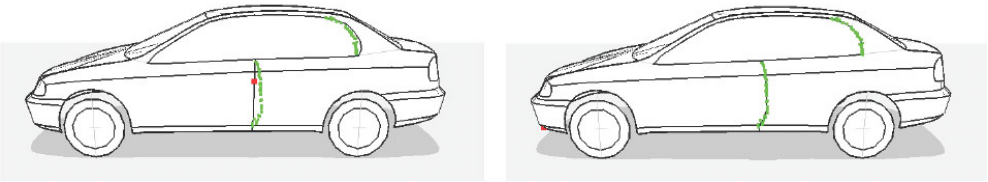


Figure 2.29: Editing a car model using sketched strokes (image courtesy of Kokai [KFS+07])

cial features, or a variation of an existing individual where the facial features are posed in a new way. A pose is a configuration of a person's facial features where they portray a particular expression, viseme, or a combination of the two. All the methods described here are template-based which further supports the approach taken in this thesis.

Xiao [Xia04] construct a simple face model using 9 sketched strokes in any order corresponding to 9 predetermined components to make up the model. The method relies on predetermined stroke features that specify object components. Two examples of such features are the angle of the bounding box diagonal (f_4) and the cosine of the initial angle of the gesture where $\cos\alpha = x_2 - x_0\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}$. These metrics are normalised and classified using K-means clustering [Mac67; Llo82]. The strokes are used to classify the 9 features with k-means and used to construct 3D objects where each shape is rendered as a simple 3D object. The 3D reconstruction is made up of simple shapes (ellipsoid, cones, cylinders) where the features do not form a continuous surface. Therefore it is not suitable for creating an accurate likeness of a real person.

Chiu [Chi05] quickly poses a 3D model by sketching simple cartoon-like strokes on a separate 2D template window. Weights are extracted from the strokes representing the eyes, eyebrows, and mouth. These weights are made to fit the D.A.N.C.E. interface by Shapiro et al [SFNTH05]. From these weights, D.A.N.C.E. constructs a facial pose on a standardised head model using blendshapes. This method cannot be used to create a new individual with particular features as the blendshapes and weights only describe poses for the facial features of a single face model, not diverse shapes.

Nataneli and Faloutsos [NF07] similarly sketch simple facial expressions in a separate 2D window which is used to pose a 3D model (see Figure 2.30. Predefined shape-attributes are found from the sketched strokes, and Support Vector Machine (SVM) classifiers are used to find the corresponding component (facial feature) on the face. Each component has a library of templates which map to a set of parameters that control the feature on the face model. The parameters can be used to pose a range of standardised face models. This method is robust and suited for quickly generating a facial expression that resembles the simplified sketch, but cannot make detailed changes, and is not aimed at modelling novel features.

Figure 2.30: Posing a 3D face with a simple 2D sketch (image courtesy of Nataneli [NF07]).

Instead of sketching strokes into the separate 2D screen space, Sucontphunt et al [SMND08] use a pre-made 2D sketch template where the strokes are made of connected feature points (see Figure 2.31). The feature points are moved around to depict new poses, which are then reconstructed in the 3D space. A prior knowledge in the form of feature points is gathered from motion data and used in a hierarchical Principal Components Analysis (PCA) model. At the top of the hierarchy lies a PCA space for the entire face (all feature points), the middle level is divided into the upper and lower parts of the face, and at the bottom are PCA spaces for individual segmented features, such as an eyebrow. If the shape of an eyebrow is changed by moving one of its feature points, the change

is projected into the eyebrow PCA space which gives more accurate results than projecting the small change to the whole face at the top of the hierarchy. This projection onto a segmented space is sometimes described as a way to increase the expressiveness of the overall generative model [BV99; KFS+07]. The projected coordinates are then projected to the PCA spaces higher up in the hierarchy which correlates the remaining features. This method is robust but currently the interactive sketch template is limited to the front view, and although it can be applied to different models as the methods discussed above, the template shape does not convey information about the distinguishing conformation properties of the facial features.

Sucontphunt et al [SDN09] applies the same hierarchical PCA approach to modelling 3D faces where a user can pull eight facial contour curves on a template 3D model, made up of anthropometric landmarks (or alternatively any vertex on the mesh). The deformed contours are subject to anthropometric constraints made up of 35 landmarks, along with measurements and proportions based on them. A source image can be transferred as a texture map for the final model using an image analogy algorithm, focused on segmented regions of the face.

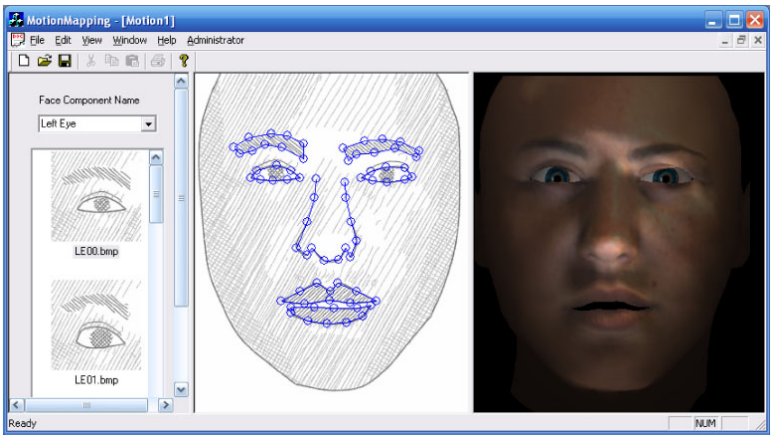


Figure 2.31: Feature points on a 2D face template are manipulated or recorded features are selected to manipulate a 3D face model (image courtesy of Sucontphunt [SMND08])

Sharon and Panne [SvdP06] map constellation models using a single-variate

probability distribution to classify sketched strokes. A constellation model for a face is shown in Figure 2.32 and is stored as a feature vector. The constellation models define relations between predefined features on the face where the dotted relations are optional. As Figure 2.32 shows, 5 labels are mandatory for the face, and 10 are optional. A collection of feature vectors are used to fit the probability distribution. Given an arbitrary sketch of a face that conforms to the constellation model, the labels are classified using a maximum likelihood search and the strokes subsequently colour coded according to the labels (see Figure 2.33). A label cannot contain more than one stroke which limits the artistic freedom, and the labels are not used here to reconstruct a 3D object.

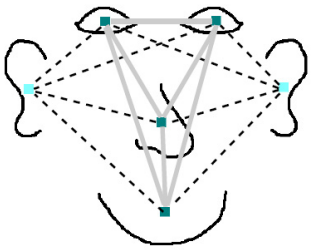


Figure 2.32: Constellation model for a face (image courtesy of Panne [SvdP06])

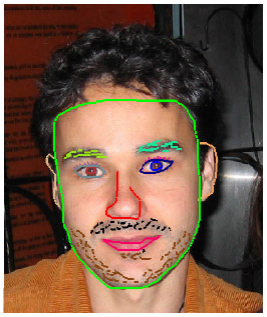


Figure 2.33: Classified strokes (image courtesy of Panne [SvdP06])

Wu et al [WTBS07] can create a 3D object by defining how normal directions are spread across its surface using an intuitive technique called shape palettes. A shape palette is an orthographic projection of an existing 3D object and its normal directions. The user starts by sketching a silhouette (outer contour) of the object which by definition have normals perpendicular to the viewing vector. Since the viewpoint is fixed as the front view, the normals for the silhouette lie on the sketching plane pointing outwards. Internal features can be sketched but now the normal directions have to be defined by sketching a reference stroke on a shape palette. The normal directions embedded in the palette can be read and

transferred from the reparameterised reference stroke to the corresponding target stroke. This gives a sparse set of normals which can be interpolated to create dense normals using a MRF optimisation. Figure 2.3 shows how a face mask can be created using an existing face as a palette. The boxes transfer normals from a particular surface region with the help of marked feature points within the defined region. The generated surface can be textured using the backdrop image which can consequently be viewed from different viewpoints.

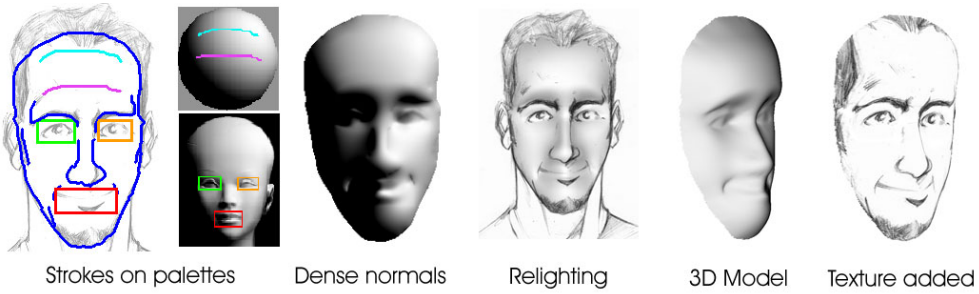


Figure 2.34: Sketching a face mask with ShapePalettes (image courtesy of Wu [WTBS07]).

Chang and Jenkins [CJ06] look for the optimal pose in a collection of key poses which they call articulation space (see Figure 2.35). They do this using a reference and a target curve, and search for the optimal articulation weights which minimise the distance between the two curves, using a downhill simplex method. This collection can be either made up of blendshapes, or alternatively a set generated using their own sketch-based approach to creating new articulation poses. These new poses are created by specifying particular regions of interest on the face, followed by a specific region of interest that controls the articulation weights and are fixed in the 2D image plane. The user then draws a reference curve, and a target curve which is reparameterised to contain the same number of points as the reference curve. Curve interpolation acquires intermediate poses using translation or rotation and scaling. The mesh is deformed according to the curves where the influence on a vertex in the region of interest is stored as a proportional distance to the projection onto the reference curve. Same interpolation are used as when interpolating curves, but it can be constrained to

move along a surface of a sphere by maintaining a particular distance from the sphere. This is done to avoid the eyelid polygons inadvertently intersecting with the eyeball polygons. This articulation process is not guaranteed to give realistic poses but is able to create new poses without any prior knowledge.

Lau et al [LCXS07] further improve the notion of using a reference and a target curve to find the optimal pose in an space of pre-posed models by tackling the problem in a probability framework. Their data set consists of motion capture recordings of facial movements characterised by 55 reflective markers. PCA is performed on the captured examples to obtain a low-dimensional representation of the data set. Using a maximum a posteriori (MAP) framework, they relate the most likely low-dimensional poses to a set of user constraints. As there might be a number of poses that satisfy any given constraints, the generated pose is forced to lie in a space of realistic poses which is learned from the recorded examples using mixtures of factor analysers (MFA). Their interface is shown in use with a graphics tablet in Figure 2.36. They validate their work by comparing the optimal reconstructed mesh with the original data. They show that the reconstruction error is considerably smaller than one based on interpolated blendshapes, and that using MFA outperforms PCA optimisation.

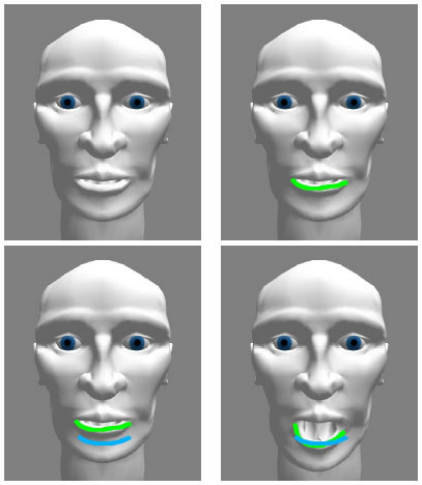


Figure 2.35: Finding an articulated pose using sketched curves (image courtesy of Chang [CJ06])

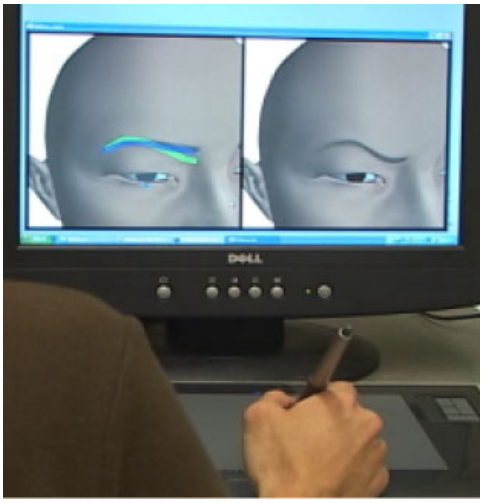


Figure 2.36: Sketching a lowered eyebrow in FacePoser (image courtesy of Lau [LCXS07])

Chapter 3

Building a training set and registration of face data

It has been established that the approach taken in this thesis is to apply a template-based sketching method where the template class is a generative model based on prior knowledge of faces. This prior knowledge consists of observed face models and must specify the facial features in such a way it is possible to determine how they differ in relation to each other. This could either mean they share the same topology, or some set of measurable and comparable properties. A collection of faces satisfying these conditions is denoted as standardised set. Obtaining a large number of anatomically accurate 3D face models is generally achieved in one of two ways.

- Use a program that is capable of generating a range of standardised face data. These face models are noise free and share the same vertex topology. However, these models are often too smooth and lack detail.
- Record or use a database of recorded faces using a tool such as a digital laser scanner. This face data is usually raw, noisy and not standardised. The advantage of these models however lies in their intricate details.

This thesis uses the first approach to populate the training sets used in subsequent chapters, where they are generated using FaceGen. The noise free data is

well suited for creating a controllable testing environment, but the absence of particular features and subtle details demands the future consideration of adapting recorded data into the training set.

Section 3.1 quickly describes the standardised models produced by FaceGen, while the main thrust of this chapter is contained in Section 3.2 involves demonstrating how an arbitrary laser scanned face model can be standardised by registering a generic, standardised model to the scan. Once the face has been standardised, it can be added to the existing training set.

3.1 Collecting face data: FaceGen

In order to get an efficient and controlled testing environment with a noise free data set, FaceGen [1] is employed to generate a number of low resolution face models (863 vertices). Some face samples with texture maps are shown in Figure 3.1. The high resolution versions (6174 vertices) were also included and tested although the main thrust of this thesis focuses on the low resolution models. A

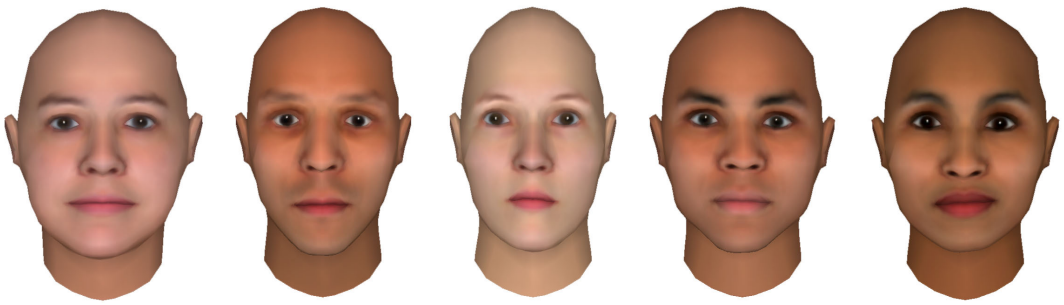


Figure 3.1: Sample faces taken from FaceGen

generative model trained on a set of faces struggles to create unobserved features. Importantly, FaceGen has the facility of customising facial features which is used to generate a diverse range of features which are not necessarily present in an arbitrary set of scanned faces. However, some features are difficult to generate with FaceGen, and the models are very smooth, generally lacking important details such as dimples, noticeable internal creases, and wrinkles. This is particularly

[1] Singular Inversion. http://www.facegen.com

true when using the low resolution models. These artefacts are added using the texture map and therefore not picked up in the contour generator described in the next section.

Each low resolution model contains 863 vertices, where the face mask consists of 775 vertices, and each eye has 44 vertices. The high resolution models have 6174 vertices, where 5832 belong to the face mask, and 171 to a single eye. 182 models were generated to fit the statistical model, and another 48 to assist with the contour mapping process.

3.2 Collecting face data: Laser scans

The advantage of using laser scans is that they may include features not present in an existing generative model such as FaceGen, both in terms of the shape of the main features and internal facial details. Scanned data is often noisy but that can be resolved by applying smoothing. A problem with scanned data is that it contains a very large number of vertices where the polygons are structured in an inefficient way (see Figures 3.2 and 3.3), making them unsuitable for a real-time application. If the grid structure was simplified by reducing the number of vertices, the simplified model would still not have an optimal structure that follows the natural lines along the facial features as shown in Figures 2.1 and 2.2.

More importantly, unlike data generated using a tool like FaceGen, a set of laser scanned face models each has its own vertex topology making any direct comparisons between them difficult. Additionally, the vertices have no intrinsic meaning with regards to the facial features so any higher order comparable properties are not present. Either one of these properties or both are needed to build and sustain a useful knowledge base of morphable faces which can be used to generate a variety of faces. The scanned models only consist of the face mask, and are often asymmetrical whereby facial features are sometimes partly missing on either side (see Figure 3.3).

A statistical data set has to be made up of samples with identical number of dimensions. Therefore all the models must share the same, efficient vertex topology. This is achieved by using a single, generic face model and deforming it

to approximate every single laser scan. As a result, there exists a representation of each individual made up of the same set of vertices and polygons. The method used here to accomplish this is well known in the literature and involves labelling each of the laser scanned models and the generic face model with a set of common landmarks denoted as feature points (FPs), and deforming the generic model where the deformation process is modelled with a procedure known as Radial Basis Functions (RBF).

The overview of the standardisation process is pictured in Figure 3.2 and the following is a brief description of the four main steps involved:

- Collect a data set consisting of 3D laser scans of real people where each model has its own unique vertex topology (Section 3.2.2).
- Label predefined features on the face models with a set of FPs (Section 3.2.3).
- Use the FPs to generate a collection of face models approximating the laser scans by deforming a generic head model (Section 3.2.8). All the approximated models share the same polygon structure as the generic model.
- Fit the approximated models to the original laser scanned models. This process is called registration and is explained in Section 3.2.10.

The outcome of this procedure is a collection of anatomically correct, standardised face models. Anthropometric measurements based on the labelled FPs, and shared vertex topology can be utilised to perform statistical analysis.

3.2.1 Related work

There are a number of databases available consisting of 3-dimensional face models:

- USF HumanID 3-D Database [BV99]
- The University of York 3D Face Database [oY03]
- GavabDB: a 3D Face Database [MS04]
- 3D RMA Database (part of the M2VTS project) [SS]

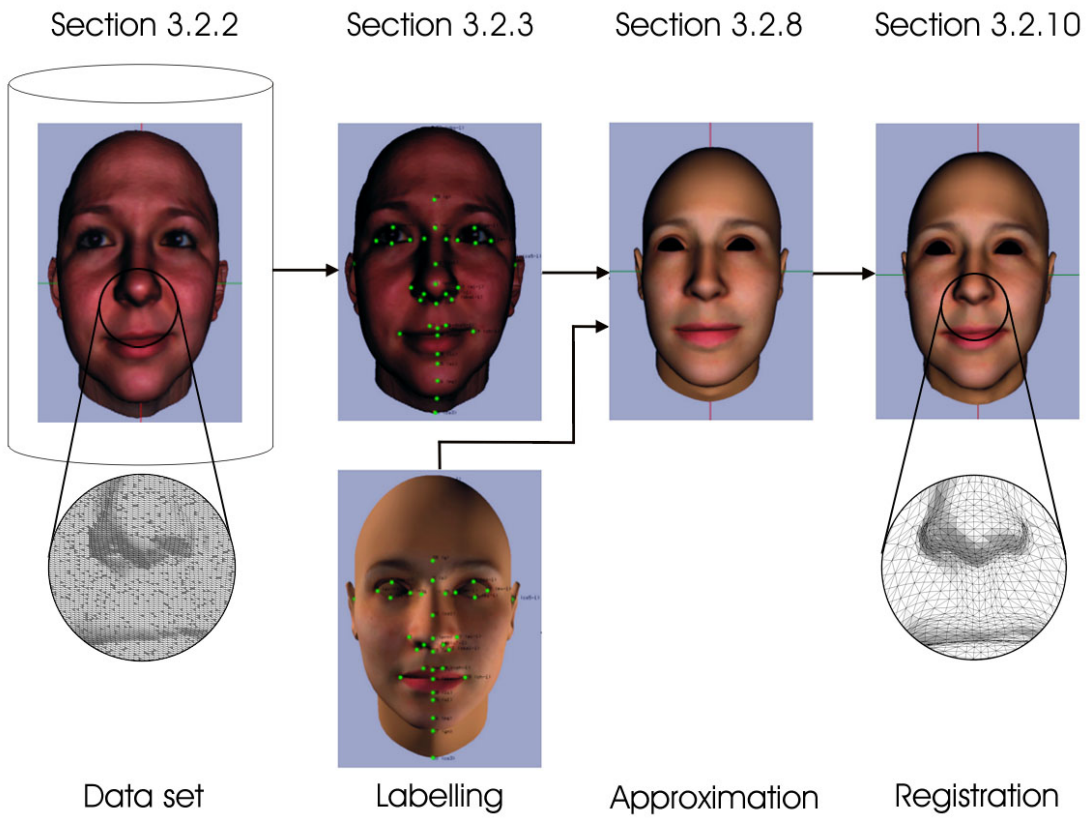


Figure 3.2: Steps taken to standardise a collection of scanned face models

- Notre Dame Biometrics Database [oND]
- Face Recognition Grand Challenge (FRGC) Database [PFS+05]

Their usage covers facial reconstruction, facial animation, and statistical face analysis and recognition. In order to use the data for these purposes, the models have to either share some common descriptors, or have the same structural build-up. The process of acquiring this state is called standardisation. The term registration is used when an approximated face is fitted onto the laser scan. The registered face is therefore a standardised model where it looks nearly identical to the original non-standardised scan.

A number of methods use FPs and approximate a generic model using RBFs. Khler et al [KHYS02] standardise the laser scanned models with an initial set of FPs using RBFs, and then iteratively find and add new FPs to generate a new approximation until the algorithm converges onto an acceptable reconstruction. Zhang [Zha06] registers the RBF approximation by moving its vertices to the closest vertex in the laser scan. The method cleans up the scanned texture by replacing the hair and the cap used in the scanning process with the appropriate skin colour. Additionally, the ears, which normally cause problems in the registration process, are registered separately and then added onto the final model.

This method is also widely used to transfer animation parameters between meshes of different structure [Eck91; NLE+99; PHL+98; yNN01; KHYS02; LEKM03; Zha06]. The difference between these methods with regards to the standardisation process lies in how they register the model after approximating it using the RBFs. Pighin [PHL+98] uses labelled images to interpolate a textured 3-dimensional head using RBFs. Neumann et al [NLE+99] use RBFs to approximate a standard model to a laser scan. An energy function is defined from the difference between the approximation and the original scan which is then subject to minimisation through cylindrical projection. Neumann and Noh [yNN01] and Edge et al [LEKM03] get a dense mesh correspondance between a source model (or in the latter case a control mask) and a target model that enables the target to be animated using the animation parameters from the source. The source is

fitted to the target using RBFs and then a cylindrical projection is used to register the model correspondence.

As a part of the IDENT project, Fieller et al accumulated a standardised data set by labelling over 3000 individuals with 30 landmarks [1]. The landmarks were placed on photographs of each individual from different angles, and then 3D coordinates were obtained by the Geometrix software. The Expectation Maximisation (EM) algorithm was used to locate missing or obscured landmarks. A Cyberware scanner was used to verify the Geometrix readings but the scanned models have not been processed here to obtain a standardised data set because of difficulties obtaining a parser for the laser scan file format.

Colbry et al [CSJ05] detect feature points automatically using a gradient filter recording the change in distance along the depth axis, a statistical model trained on distances between feature points, and a shape characteristics based on surface curvature. The gradient filter can easily detect the tip of the nose, and after finding the top of the head, the model can be aligned and the statistical model generates bounding boxes for the features based on its prior knowledge. The features are then found within their bounding boxes using the shape characteristics.

Blanz and Vetter [BV99] use the Max Planck database [TN96] to build a set of morphable 3-dimensional faces. The database consists of 200 laser scanned heads and registers each head by using an optic flow algorithm to minimise the difference between a laser scanned head and a reference model created by reconstructing the scan using a combination of principal components with Gaussian distributed coefficients. Curzio Basso et al [BPV06] later improved the morphable model, making it more accurate and robust, as well as adding the possibility of reconstructing missing data.

FaceGen [2] is a parametric database of 3D faces that uses Principal Components Analysis (PCA) to construct a face space, similar to the morphable model

[1] http://www.pas-postgrads.group.shef.ac.uk/morecroft

[2] Singular Inversion. http://www.facegen.com

by Blanz and Vetter [BV99], where each of the 128 principal components is one standard deviation in magnitude. By navigating along these axes, new faces can be generated by adding different deviations (classified parameters) to the average face located at the origin.

3.2.2 Laser scanned face data

The laser scanned faces will provide a platform for creating the generative properties used in a modelling or animation system. It is therefore important the scanned models captures a wide range of consistent features. The USF HumanID 3-D Database fulfills this requirement where it consists of a collection of 138 high quality laser scanned faces of different ethnic origin (97 male, 41 female) [BV99], and their corresponding texture images (512x512 pixels). The faces are stored in a binary Inventor (iv) V2.1 file format and are processed with a program called *ivfix*, written by James Ward [1], where the output file is an ascii Inventor file format. An example of a face from this dataset is shown in Figure 3.3. The models are exported to the Polygon File Format (ply) which is a simpler, and more efficient format for the purposes of reading an ASCII file. Appendix A gives a detailed description of the IV and PLY file formats.

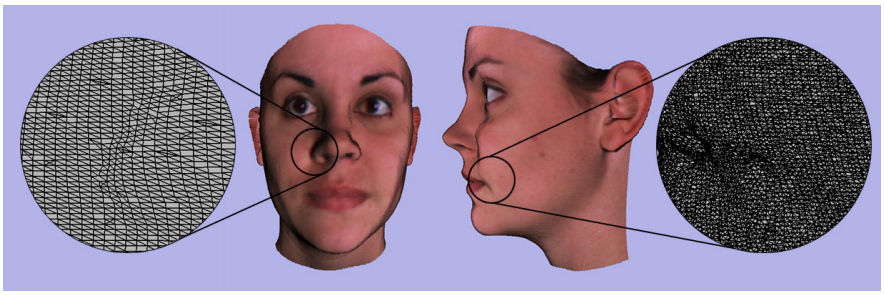


Figure 3.3: An example of a laser scanned face from the USF HumanID 3D database

[1] http://www2.dcs.hull.ac.uk/simvis/personnel/jww.htm

3.2.3 Labelling faces

The laser scans are raw data and therefore do not contain any higher level information about the location of individual facial features. Additionally, every scan is made up of a different vertex and polygon topology, making it hard to perform direct comparisons and meaningful measurements of features without further information. An additional data set specifying feature characteristics is therefore needed to perform these tasks. A popular approach is to label the models with a set of Feature Points (FPs) where each point represents a facial feature such as the tip of the nose, or the corner of an eye. Faces have a well defined set of features so any method of labelling will share a large number of FPs with other methods.

The most common sets of FPs used are anthropometric or craniofacial landmarks, and the MPEG-4 landmarks which focus on facial animation. Sections 3.2.4 and 3.2.5 briefly describe the two sets, and Section 3.2.6 explains what set of FPs are chosen to perform the interpolation and registration. Section 3.2.7 shows the effect of altering the set of non-crucial FPs on the final registered model.

3.2.4 Anthropometric landmarks

Anthropometry is the measurement of living subjects. Anthropometric landmarks identify visible or palpaple features on the subject. On the head they are called craniofacial landmarks. Hussuna [EH03] use craniofacial landmarks used in plastic and reconstructive surgery, provided by Kolar and Salter [KS96], to calculate statistical variations of 3-dimensional face models and show that the faces have a reasonable Gaussian distribution around their mean. Figure 3.4 shows the craniofacial landmarks on a photograph of a young boy.

3.2.5 MPEG-4 landmarks

MPEG (Moving Picture Experts Group) is a working group of ISO/IEC in charge of the development and standards for codec representation of digital audio and video. MPEG-4 is an object-based multimedia standard and with it comes the

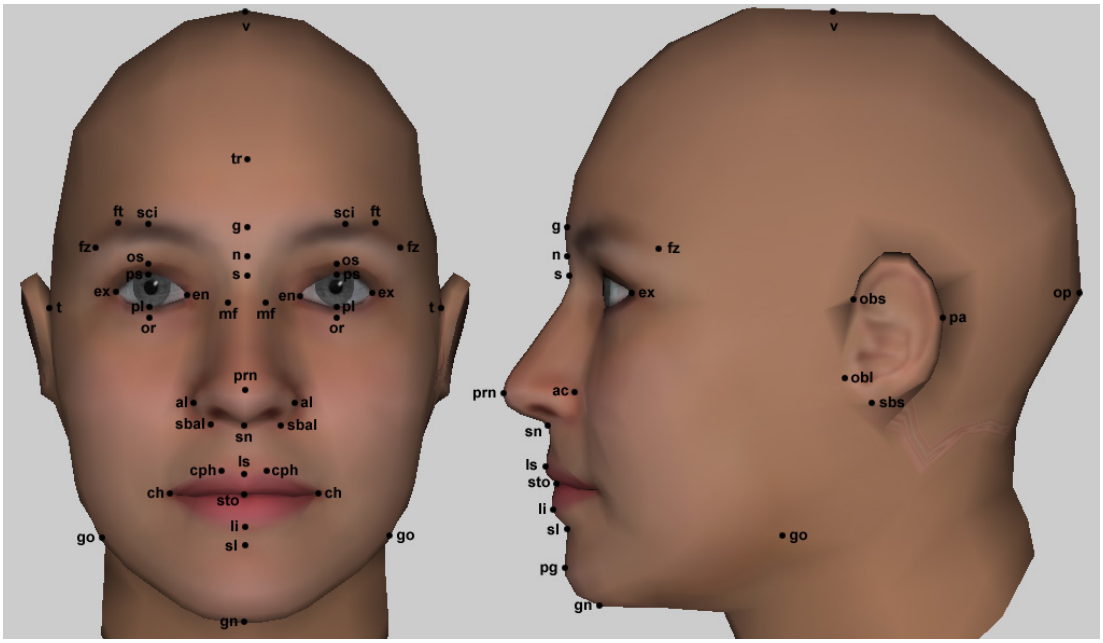


Figure 3.4: Pictures show craniofacial landmarks super-imposed on a photograph (after Kolar and Salter [KS96])

possibility of defining a face model in its neutral state, where its features are labelled using 84 Feature Definition Points (FDPs), and a set of Facial Animation Parameters (FAPs) defined by deriving spatial relations from the FDPs [Pak02]. Figure 3.5 shows the 84 FDPs used in the MPEG-4 standard. Labelling and standardising a set of face models using the FDPs would mean the registered models are ready for animation without the need for further labelling.

3.2.6 Choosing feature points

Choosing the right set of FPs improves the accuracy of the registered model. FPs are omitted where the location on the 3D model is ambiguous or hard to place as inaccuracies or notable variations in the labelling process can cause the registered model to have a slightly skewed polygon structure. The risk is higher in areas with a more complex surface curvature as they tend to have higher density of vertices. This is demonstrated in Figure 3.6 where the craniofacial landmark *ac*

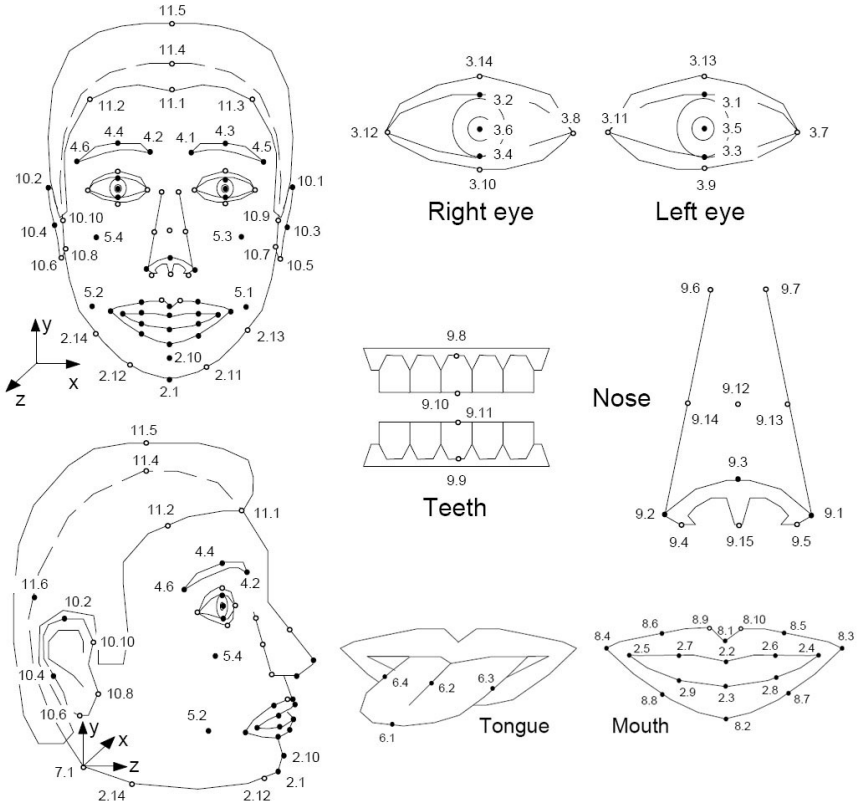


Figure 3.5: MPEG-4 Feature Definition Parameters (Points)

(9.13-14 in the MPEG-4 standard) is added when approximating the shape of the nose. The picture on the left shows that after the model has been fitted to the laser scan, the polygons circled in red have been compromised and do not produce a smooth surface for that particular region. Omitting this landmark results in a better fit (right).

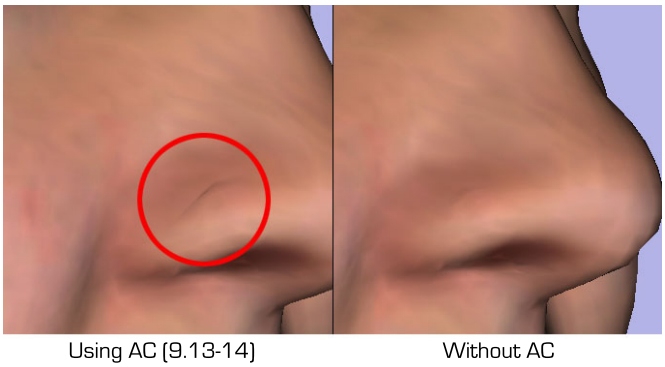


Figure 3.6: A possible side effect of using a landmark which is hard to place on the model.

The number of FPs are kept to a minimum to avoid unwanted artefacts. Neither the teeth nor the tongue are used which discards all MPEG-4 landmarks relating to these features. Only one point per ear is used to interpolate an estimated position as they are not registered.

The chosen set consists of 37 FPs where the majority can be found both in the cranofacial and MPEG-4 sets. However, mutually exlusive landmarks consist of the craniofacial landmarks *s*, *g*, *c'*, *sl* (see Figure 3.4), and the MPEG-4 landmark *9.12* (*cs1*) (see Figure 3.5). Two customised landmarks are used for the neck area (*cs2* and *cs3*).

The FPs are pictured in Figures 3.7 and 3.8 and listed in Table 3.2.6 where the corresponding cranofacial and MPEG-4 label is given where it exists. The extension -l (left side) and -r (right side) are used where appropriate to distinguish different sides of the face where -r stands for the right side according to the model, not the right side of the face from a front-facing observer. The FPs are shown on

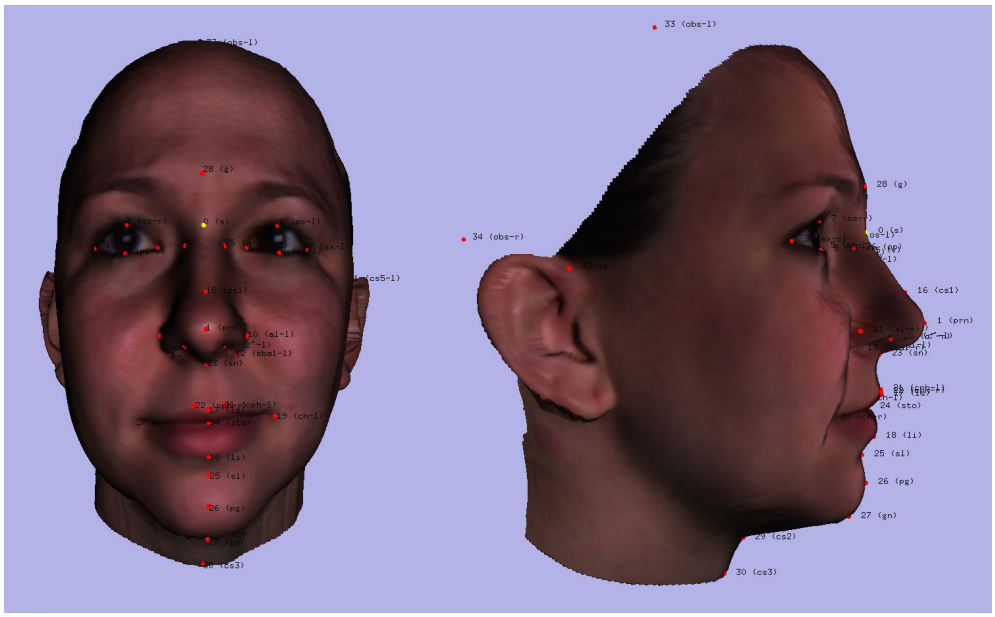


Figure 3.7: Landmarks used (laser scan)

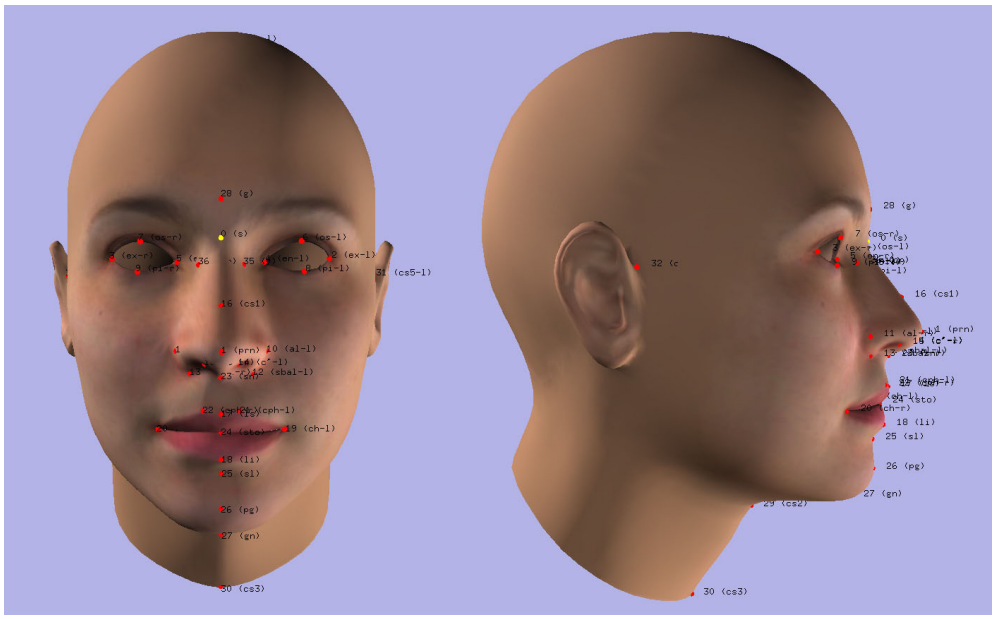


Figure 3.8: Landmarks used (standard female head)

a laser scanned model in Figure 3.7, and on a generic female head in Figure 3.8, shown as green dots.

The FPs *v* and *op* cannot be placed on the laser scans because the scans only consist of the face mask. Anthropometric proportions are used to locate these points on the scanned models. Alternatively, a statistical model could be used to determine the most probable location for these FPs based on pre-labelled models achieved with the following approach. The FPs are mapped to the nearest vertices on the generic model. A FaceGen model is used here to act as the generic model which shares its vertex structure with every model generated in FaceGen. Creating a number of models with FaceGen creates a range of FPs which is used as a training set for a probability model. Once the FPs have been labelled on a laser scan, the FPs that cannot be placed can be determined by the probability model based on the location of the known FPs.

3.2.7 Using different sets of FPs

It is of interest to find out how important individual feature points are with regards to the accuracy of the complete registered model as a reconstruction. The fitting process requires the models line up so points defining the main structure and orientation such as the tip of the nose and approximate skull width (defined by where the ears connect to the skull) are required. Points lining the main facial features are equally important to maintain the structural integrity between the approximated model and laser scan.

Other FPs are omitted if they do not contribute to a noticeable improvement to the registered model. This is done by measuring the difference between the depth maps of two registrations where one has been generated by including all 37 FPs, and the other by using the subset of FPs being examined.

It was found the FPs *g, s, c'-l, c'-r* and *sl* can be omitted with negligible effect on the fully registered model as shown in Figure 3.9. Figure 3.9 visualises the difference between the RBF approximated models (left), and the fully registered models (right), where the colour scale has been restricted to a narrower interval to

Cranof.	MPEG-4	Region	Definition
s	-	Nose	Sellion
prn	9.3	Nose	Pronasal
ex-l	3.7	Orbits	Exocanthion left
ex-r	3.12	Orbits	Exocanthion right
en-l	3.11	Orbits	Endocanthion left
en-r	3.8	Orbits	Endocanthion right
os-l	3.1	Orbits	Orbitale superius left
os-r	3.2	Orbits	Orbitale superius right
pi-l	3.3	Orbits	Palpebrale inferius left
pi-r	3.4	Orbits	Palpebrale inferius right
mf-l	9.7	Nose	Maxillofrontal left
mf-r	9.6	Nose	Maxillofrontal right
al-l	9.1	Nose	Alare left (The most lateral point on the nasal ala)
al-r	9.2	Nose	Alera right
sbal-l	9.5	Nose	Subulare left
sbal-r	9.4	Nose	Subulare right
(cs1)	9.12	Nose	Middle lower edge of nose bone (or nose bump)
c'-l	-	Nose	Columella apex left
c'-r	-	Nose	Columella apex right
ls	8.1	Orolabial	Labial superius
li	8.2	Orolabial	Labial inferius
ch-l	8.3	Orolabial	Cheilion left
ch-r	8.4	Orolabial	Cheilion right
cph-l	8.10	Orolabial	Crista philtre left
cph-r	8.9	Orolabial	Crista philtre right
obs-l	10.9	Ears	Otobasion superius (Highest point of attachment)
obs-r	10.10	Ears	Otobasion superius (Highest point of attachment)

Table 3.1: Feature points and their corresponding anatomical and MPEG-4 landmarks

Cranof.	MPEG-4	Region	Definition
sn	9.15	Face	Subnasal
sto	2.2-3	Face	Stomion, point buccal
sl	-	Face	Sublabial
pg	2.10	Face	Pogonion
gn	2.1	Face	Gnathion or Menton
g	-	Head	Glabella
v	11.4	Head	Vertex (Highest point of the head)
op	11.6	Head	Opisthorcranion or occipital point
(cs2)	-	Head	Point where the chin and neck meet
(cs3)	-	Head	Lowest point on neck

Table 3.1: Feature points and their corresponding anatomical and MPEG-4 landmarks

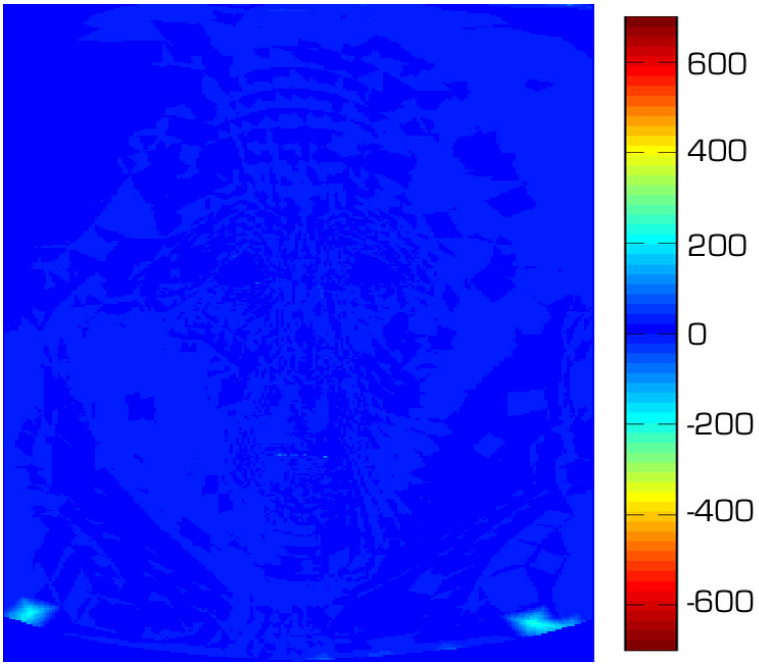


Figure 3.9: The difference between using the full set of FPs and the omitted version shown in the original context (same scale)

show the differences more clearly (red for difference values -80 and 80 as opposed to -600 and 600). As expected the areas in the interpolated model (left) where the feature points were removed are affected, but are corrected in the fully registered model (right).

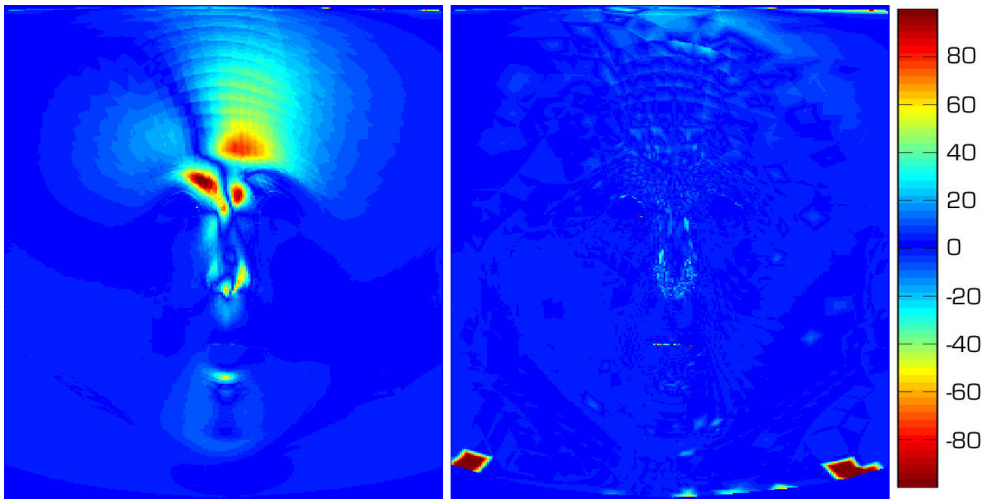


Figure 3.10: The effect of omitting non-crucial FPs showed as the difference between the full set of FPs versus the omitted version. The RBF model is on the left and fully registered model on the right. The scale has been changed to see the difference more clearly

3.2.8 Approximating faces

A database of 138 laser scanned face models is used where the models share a set of FPs identifying common features. Two topologically efficient, complete, and clean head models (male/female) are used as a generic head model (top left in Figure 3.11) which have been marked with an identical set of FPs. The generic models are deformed to approximate the structure of every laser scanned face mask in the database. This gives a collection of complete head models sharing the same vertex topology. The quality or polygon complexity of the standard head should be good enough to produce a large range of unique facial features, without sacrificing the possibility of near real-time calculations and rendering.

A function f is needed to map the FPs on the generic models to the FPs on the laser scans, as well as providing interpolation for every vertex on the meshes that lie between the FPs. The is solved here using a Radial Basis Function Network which is explained in the following section.

3.2.9 Radial Basis Functions

Let $\mathbf{p_i} \epsilon \mathbb{R}^3$ and $\mathbf{q_i} \epsilon \mathbb{R}^3$, $i = 1, ..., n$ where $\mathbf{p_i}$ are the source FPs that lie on the standard face and $\mathbf{q_i}$ are the target FPs that lie on the laser scanned face.

Radial Basis Function (RBF) fitting is an approach to scattered data interpolation and is a bridge between instance-based and neural network learning algorithms [Mit97]. It is widely used for face model fitting [KHYS02; LEKM03; PHL+98; yNN01; Zha06]. The basic idea is that a model, often referred to as a standard model or source model, is deformed to approximate another target model using interpolation. Figure 3.11 shows the approach. A set of feature points (FPs), indicated with red circles, are placed on both models where the sequence and location of the FPs are equivalent. The RBF process calculates interpolation coefficients that represent how the coordinates for the FPs on the source model become the coordinates for the FPs on the target model. These coefficients are then used to calculate the new coordinates for all vertices on the standard, source model.
The RBF Network maps the FPs $\mathbf{p_i}$ to $\mathbf{q_i}$ and is of the form

$$\mathbf{q_i} = f(\mathbf{p_i}) = \sum_{j=1}^{n} \mathbf{c_j} \theta_j(\mathbf{p_i}) + \mathbf{\Lambda p_i} + \mathbf{t}, \tag{3.1}$$

where $\theta_j(\mathbf{p_i}) = ||\mathbf{p_i} - \mathbf{p_j}||$, $\mathbf{p_i}$ is a feature point i, and $||\mathbf{p_i} - \mathbf{p_j}||$ is the euclidean distance between feature points i and j where $i \neq j$.

It therefore consists of a weighted linear combination of n basis functions θ_j where the value of the function depends only on the distance from its centre, and an additional polynomial (affine transformation) factor where $\mathbf{R} \epsilon \not\geq^{3\times3}$ adds rotation, skew and scaling, and $\mathbf{t} \epsilon \mathbb{R}^3$ is a translation component. This assures

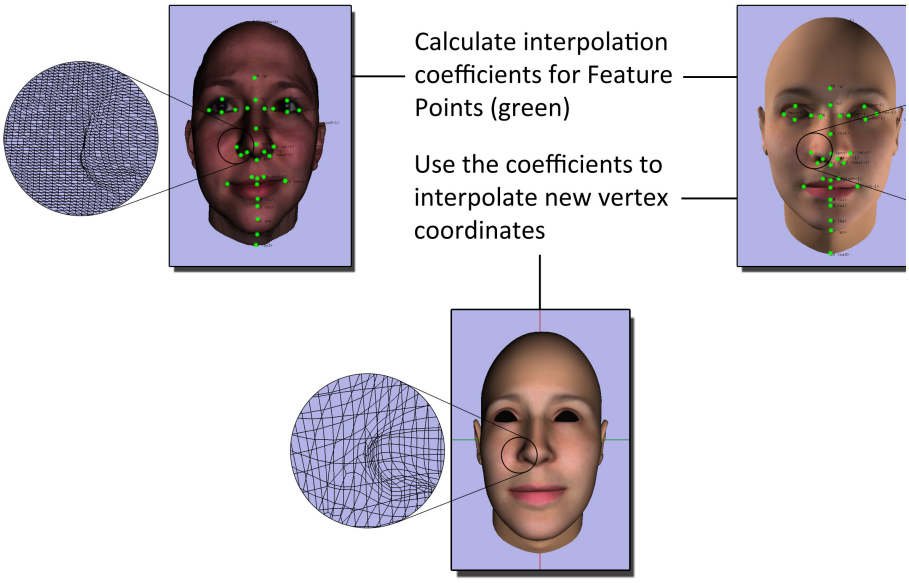


Figure 3.11: The RBF interpolation process

a certain degree of polynomial precision to avoid a poor approximation of the unknown function away from the feature points [KHYS02; PHL+98]. $\mathbf{c_j} \epsilon \mathbb{R}^3$ are the linear weights (or coefficients) that will be determined using Equation 3.4, and then used to interpolate the model vertices using Equation 3.1.

Zhang [Zha06] compares the result using different basis functions and finds that the multi-quadric gives the best approximation but, as suggested by [LEKM03], the inverse multi-quadric gives a better result

$$\theta_j(\mathbf{p_i}) = (||\mathbf{p_i} - \mathbf{p_j}||^2 + \delta_j^2)^{-\lambda},\ \lambda > 0, \tag{3.2}$$

where $\lambda = 1$ and δ_j is the stiffness constant measured by the euclidean distance between $\mathbf{p_j}$ and the nearest $\mathbf{p_i}$ as suggested by Eck [Eck91]

$$\delta_j = min||\mathbf{p_i} - \mathbf{p_j}||,\ i \neq j. \tag{3.3}$$

This leads to a smaller deformation on widely scattered feature points and larger deformation for closely scattered ones.

Figure 3.12 shows the effect of some FPs where the FP in question in shown as a white dot and the magnitude of deformation on the surrounding area is colour coded. The coding ranges from red which indicates the highest influence on the deformation, to blue having the lowest influence. The head on the far right of Figure 3.12 shows the combined effect of the 32 FPs used for the deformation process, and it is apparent that the most influenced area is around the nose and mouth where the FPs lie closer together.

The coefficients of the basis function and the polynomial can now be found by solving the linear system $\mathbf{AX} = \mathbf{B}$ where:

$$\mathbf{A} = \begin{bmatrix} \theta_1(\mathbf{p_1}) & \theta_2(\mathbf{p_1}) & ... & \theta_n(\mathbf{p_1}) & 1 & \mathbf{p_1^T} \\ \theta_1(\mathbf{p_2}) & \theta_2(\mathbf{p_2}) & ... & \theta_n(\mathbf{p_2}) & 1 & \mathbf{p_2^T} \\ ... & ... & ... & ... & ... & ... \\ ... & ... & & & & \\ \theta_1(\mathbf{p_n}) & \theta_2(\mathbf{p_n}) & ... & \theta_n(\mathbf{p_n}) & 1 & \mathbf{p_n^T} \\ 1 & 1 & ... & 1 & 0 & 0 \\ \mathbf{p_1} & \mathbf{p_2} & ... & \mathbf{p_n} & 0 & 0 \end{bmatrix} \epsilon \mathbb{R}^{(n+4)\times(n+4)}, \tag{3.4}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{c_1} & \mathbf{c_2} & ... & \mathbf{c_n} & \mathbf{R} & \mathbf{t} \end{bmatrix}^T \epsilon \mathbb{R}^{(n+4)\times 3},$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{q_1} & \mathbf{q_2} & ... & \mathbf{q_n} & 0 & 0 & 0 & 0 \end{bmatrix}^T \epsilon \mathbb{R}^{(n+4)\times 3}.$$

This system can be solved using a linear solver such as Singular Value Decomposition (SVD) or Least Squares [GVL96]. In this work, the main program is written in C++, but calls a SVD function which comes packaged with Matlab (see Appendix B). Once the coefficients have been found, the mesh vertices on the standard, generic faces can be interpolated using Equation 3.1.

3.2.10 Face registration

The approximated models acquired using the RBF interpolation do not accurately depict the original scans as shown in Figure 3.13. Figure 3.14 visualises the difference between the depth maps for the laser scan and the approximated RBF model. This is because the FPs used in the interpolation are sparse, and, as discussed in Section 3.2.3, adding more points can cause unwanted artefacts,

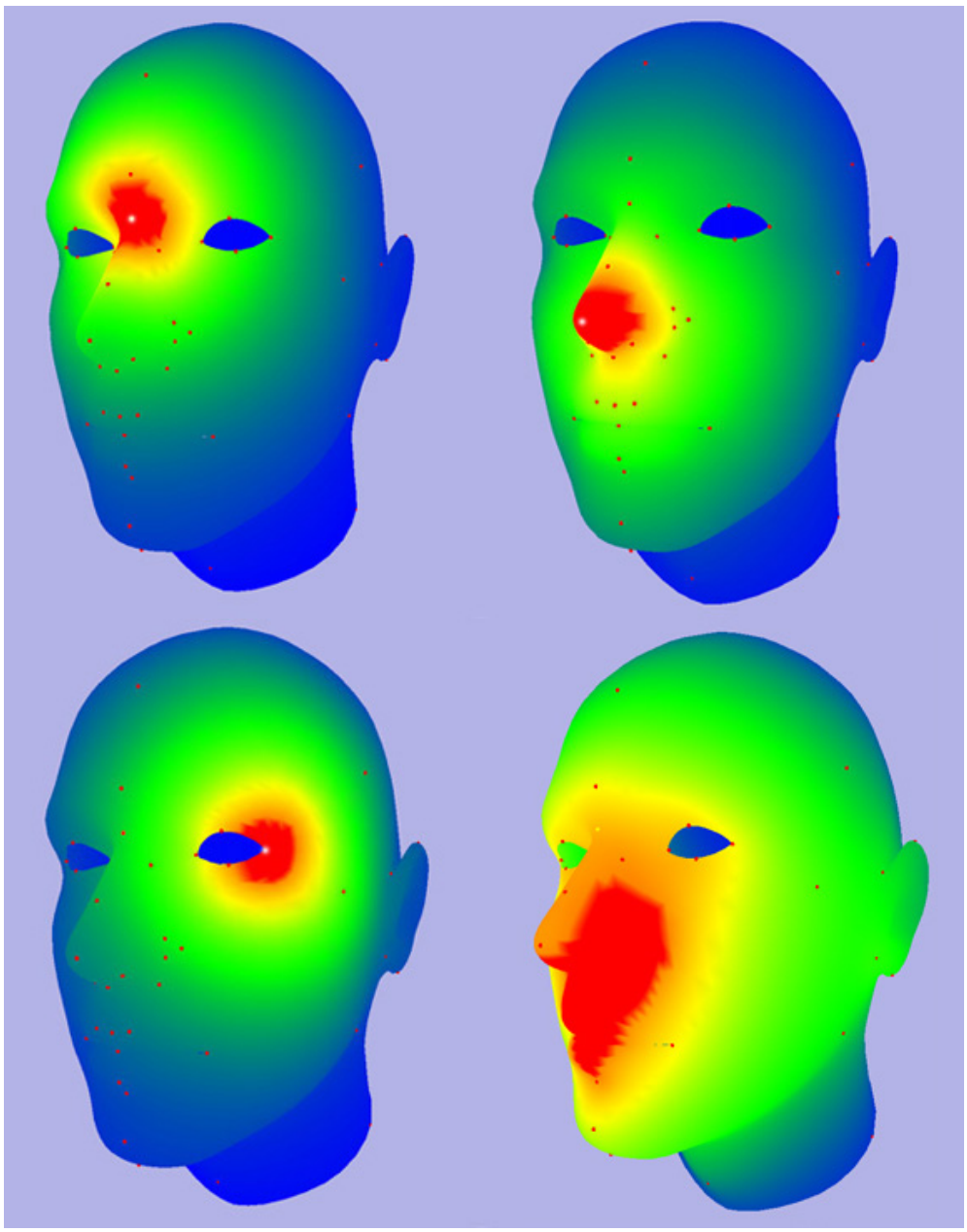


Figure 3.12: The impact of FPs on deformation in the RBF process. Red represents the maximum influence and blue the minimum influence.

particularly when they are hard to place.

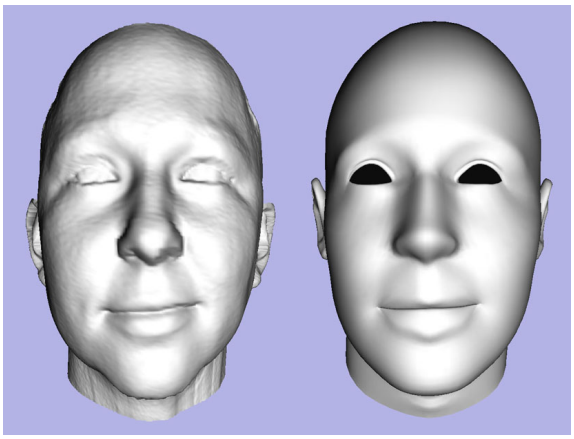


Figure 3.13: A laser scan and its corresponding RBF approximation

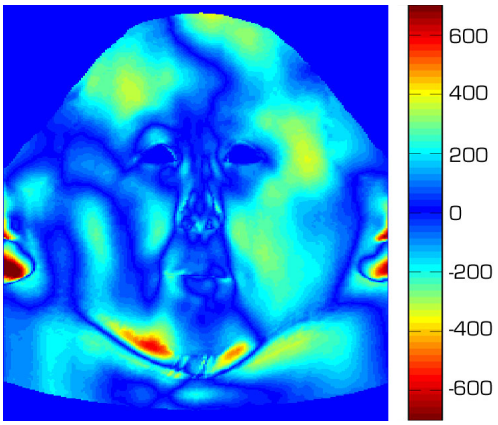


Figure 3.14: The distance between the RBF and laser scan

The difference can be even more drastic if the scanned person has a lot of fat tissue as it will not be replicated on the approximated model. Using FPs to encapsulate every type of facial feature that can come up such as fat cheeks or the particular type of chin in Figure 3.13 quickly mounts up to a large number of FPs. These FPs would need to be present even when the features are absent making them both redundant and extremely hard to place. Instead a minimum number of FPs are used to approximate the scan roughly, and then a fitting process is applied to capture the variations in shape and finer details. This fitting process is denoted as registration.

The idea is that after the RBF interpolation, the approximated RBF model and the original laser scan are aligned together where the features have roughly the same size and proportion. The RBF model is then fitted onto the laser scan by adjusting the coordinates for every vertex in the RBF model with the aim that the vertices lie on the scanned surface, thus minimising the difference between the two models. Three ways of doing this are:

- Move each vertex in the RBF model to its nearest vertex on the laser scan (see Figure 3.15). This method is straightforward as there is no need to convert the models into cylindrical coordinates. However, it is found here that this method gives less accurate results and there is no guarantee a vertex exists on which to map to because the laser scan is incomplete. Additionally, in vertex dense areas, this could compromise the vertex structure if a displaced vertex crosses over a polygon edge, or if two vertices on the RBF model map to the same vertex on the laser scan.
- Project each vertex in the RBF model onto its underlying polygon on the laser scan and calculate its coordinates (see Figures 3.16 and 3.18). This is explained briefly in Section 3.2.11.
- Create depth maps for the laser scan and RBF model of a predetermined resolution. The depth maps are created by fitting a grid on top of an unwrapped model and recording the interpolated depth in the centre of each cell. The laser scan depth map is merged into the RBF depth map, where pixels containing non-zero depths replace the corresponding pixels in the RBF map. New vertex coordinates are then retrieved by referencing the updated depth map. This is explained in more detail in Section 3.2.12.

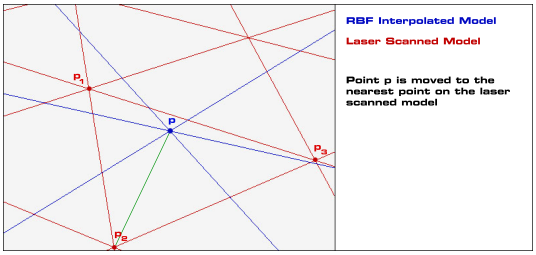


Figure 3.15: Fitting by finding the nearest vertex in the laser scan

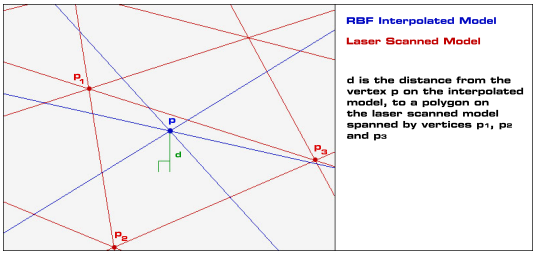


Figure 3.16: Fitting by projecting the vertex onto the laser scan

3.2.11 Projecting onto the underlying polygon

The first stage unwraps the models by transforming the vertex coordinates from euclidean coordinates into cylindrical coordinates which is a $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ mapping:

$\{x, y, z\} \rightarrow \{\theta, r, z\}$ where

$$\begin{aligned} r &= \sqrt{x^2 + y^2}, \\ \theta &= \tan^{-1} \tfrac{y}{x}, \;\; \theta \in \{0, 2\pi\}. \end{aligned} \tag{3.5}$$

Figure 3.17 shows the interpolated head unwrapped. If a vertex in the interpolated model has the same cylindrical coordinates (θ, z) as a vertex in the laser scan model, the depth value r is replaced and converted back to euclidean coordinates to get the updated location. However, it is very unlikely that a vertex in the laser scan and a vertex in the RBF interpolation lie at the exact same location in the cylindrical plane, so either each vertex in the RBF plane is moved to the nearest vertex in the laser scan plane, or interpolated values are calculated using barycentric coordinates.

For every vertex on the interpolated RBF head, a corresponding depth value is found on the scanned head using barycentric coordinates [yNN01]. There is no need to find the projected intersection point on the triangles because the two meshes lie in the same unwrapped 2D plane which means that the intersection point is always going to lie on that plane. This is shown in Figure 3.18 where every vertex $\mathbf{p}$ in the interpolated (source) head lies on a polygon ($\triangle \mathrm{p}_1\mathbf{p_2}\mathbf{p_3}$) in the scanned (target) head (unless the underlying region is missing as mentioned earlier), and the interpolated depth is found by checking its barycentric coordinate coefficients, in relation to the vertex location.

If $\mathbf{p} \epsilon \mathbb{R}^2$ is the vertex location on the interpolated head, and $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3} \, \epsilon \, \mathbb{R}^2$ are the points forming the triangle it intersects on the laser scan, giving the affine combination

$$\mathbf{p} = \alpha_1 \mathbf{p_1} + \alpha_2 \mathbf{p_2} + \alpha_3 \mathbf{p_3}, \tag{3.6}$$

where $\alpha_1, \alpha_2, \alpha_3 \epsilon \mathbb{R}$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$ if the point lies on the plane of the

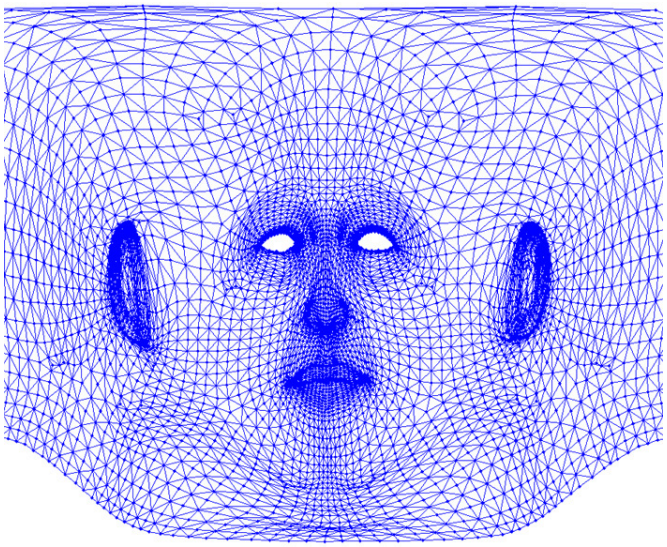


Figure 3.17: The interpolated head unwrapped

triangle $\triangle \mathbf{p_1 p_2 p_3}$. The scalars $\alpha_1, \alpha_2, \alpha_3$ are found by solving the linear system

$$\begin{bmatrix} p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \tag{3.7}$$

If the sum of the coefficients is in fact equal to 1, then the depth value r for the vertex should be an interpolation of the depth values of the vertices making up the intersected triangle in the laser scan. The interpolated depth value d_i is in fact a linear combination of the depth values where the contribution of each vertex in the triangle is multiplied with the corresponding calculated α values

$$d_i = \alpha_1 d_1 + \alpha_2 d_2 + \alpha_3 d_3. \tag{3.8}$$

At the same time it is possible to interpolate the texture UV coordinates by using the same coefficients

$$\begin{aligned} u &= \alpha_1 p_{1u} + \alpha_2 p_{2u} + \alpha_3 p_{3u} \\ v &= \alpha_1 p_{1v} + \alpha_2 p_{2v} + \alpha_3 p_{3v}. \end{aligned} \tag{3.9}$$

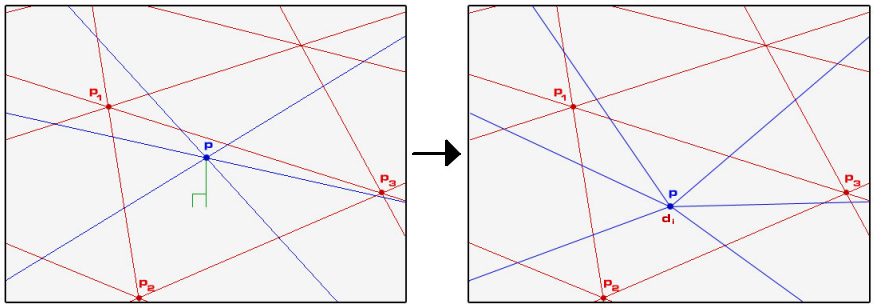


Figure 3.18: Projecting a vertex in the RBF onto the laser scan (red is scanned model, blue is RBF interpolated model).

3.2.12 Using depth maps

As shown in Section 3.2.2, some areas in the laser scan are missing and therefore it is not always the case that a vertex can project to any region on the laser scan, or that there is a vertex nearby. For these vertices it is possible to interpolate the depth based on the surrounding vertices, granted that they were successfully projected onto a polygon. This only works well for small areas as a gradual degradation occurs when an interpolation uses a previously interpolated value. Large regions need to be interpolated because the laser scan consists only of the face mask. Figure 3.19 shows the distance measured between an RBF approximated head and a laser scan using the method described in the previous section, where red indicates the largest distance and blue the smallest. Here the values for the missing parts have been interpolated and it is clear that areas where a large amount of information is missing give poor approximations. These areas become the largest contributors to the overall distance measured, making it hard to use the overall distance as a metric to compare how well different sets of FPs approximate the original laser scan.

To overcome these issues, a depth map is generated for the interpolated head (source) and its corresponding laser scanned head (target). Simple image manipulation techniques are applied which merge the two maps and interpolate the missing regions. The new approximated vertex values are then found by looking up a new depth from the pixel corresponding to cylindrical values for a given ver-

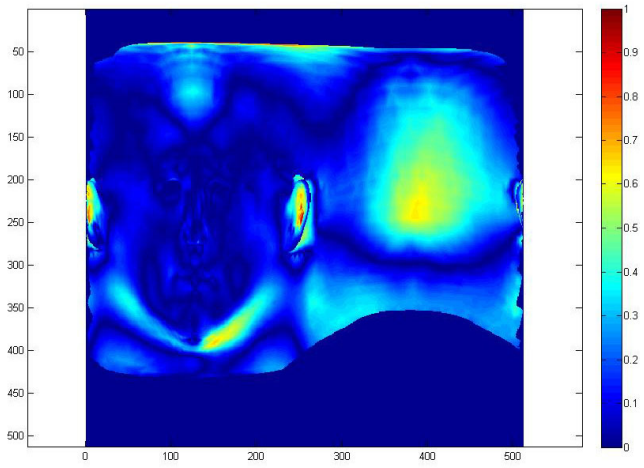


Figure 3.19: Using interpolated values for areas where no prior knowledge exists in the original data result in an unrealistic registration

tex. Furthermore, the depth maps are a form of storing the deformations needed to fit a standardised face model to the set of targets.

3.2.12.1 Generating depth maps

The depth map is created by sorting the unwrapped polygons in the interpolated model and laser scan model according to their UV coordinates for a given pixel resolution. It can be thought of as putting a grid of a fixed resolution on top of the unwrapped model, checking if the polygons are within the boundaries of each cell on the grid. The UV grid is processed by going through each cell and using barycentric coordinates to find the interpolated depth value of the underlying polygon. This gives a more accurate interpolated value and therefore a better and smoother result than taking the highest average depth value of the vertices that belong to the polygons in the cell. Sometimes polygons occlude others and using the barycentric coordinates makes sure an accurate highest depth value is taken to assure continuity.

Figure 3.20 shows a depth map of an RBF interpolated head, and Figure 3.21 shows a depth map of its corresponding laser scanned head (512x512 pixels).

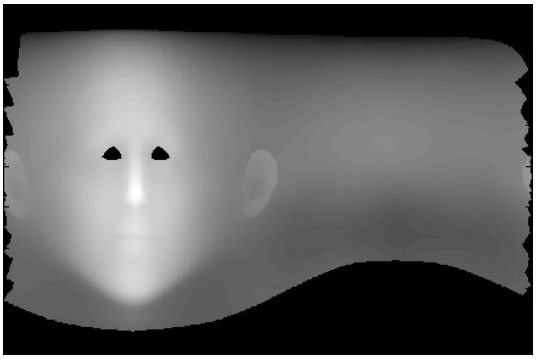


Figure 3.20: A depth map for the RBF approximated model (source)

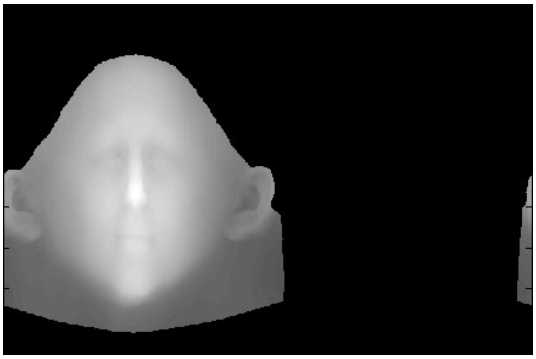


Figure 3.21: Depth map for the laser scan (target)

Figure 3.22 shows how the maps are merged together where the target values replace the source depth values. The new vertex coordinates for the registered model are found by looking up their new depth values from their UV coordinates, and then projecting the updated barycentric coordinates back into XYZ coordinate space.

The seams where the new copied values meet the old values have to be processed to produce a smooth transition because the sudden jump in depth value creates unwanted artefacts. The ears pose a problem as well because in addition to having arbitrary parts missing, they need more than one depth value because of their complex structure. To resolve this, only the area between the ears on the target map is stitched onto the source map which in effect updates the main features and face mask while retaining the remaining structure. The boundaries where the target and source map meet are smoothed by interpolating intermediate depth values over a configured interval. The merged and smoothed depth map is shown in Figure 3.23.

Two problems regarding depth maps need to be solved in order to achieve an acceptable registration:

- Depth maps are 2-dimensional and can therefore not deal with a surface that folds on itself, and occluded polygons. To obtain a smooth, continuous surface, the maximum depth value is always chosen when generating the

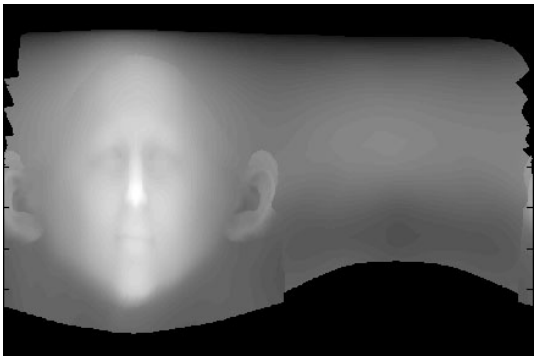


Figure 3.22: Merging the depth maps without blending

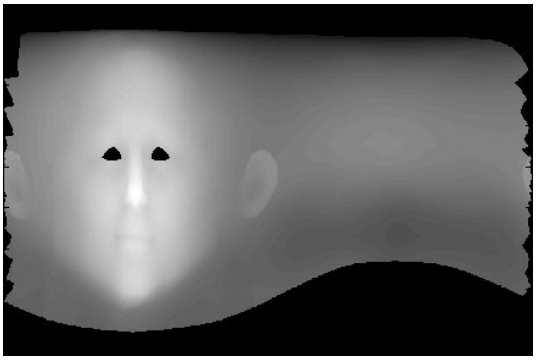


Figure 3.23: Merged depth map with blending

depth maps. The generic face model contains occluded polygons which cause problems when they project to the same depth as the polygons that occlude them. Section 3.2.12.2 discusses this problem in more detail and how it can be solved.

- The depth maps are made up of discrete pixels and not continuous floating values which causes aliasing. Section 3.2.12.3 describes how this is corrected.

3.2.12.2 Depth ambiguity

Since a depth map is a 2-dimensional image, it can only contain the top-most surface to maintain continuity. However, some meshes contain polygons that are hidden from view (occluded by other polygons). Occluded vertices will project to the same UV coordinate, and therefore the depth, as the polygon that occlude them. The generic models used here have hidden polygons behind the lips which are useful during animation to avoid unwanted gaps, as well as polygons around the eyes to fit the eyeball meshes in the sockets without producing any gaps. Other common facial parts that might be included are teeth and tongue. These problem areas are pointed out in Figures 3.24 and 3.25. Figures 3.26 and 3.27 show the effect of registering the model without dealing with this problem where

vertices are fighting to exist in the same space, showing up as black areas.

Figure 3.24: Example of occluded polygons

Figure 3.25: Hidden and occluded polygons

There are two steps involved in fixing this. The first one is when the depth map for the RBF model is generated. During that process, the maximum depth value is always assigned to the cell, but every polygon mapping to the same cell with a lower depth value is labelled as being occluded along with the index of the occluding polygon. Polygons that do not share an edge and do not lie on the model's boundaries are ignored.

This secondary data set knows what parts of the model should not get their new depth values directly from the depth map. Figure 3.28 shows the overlapped areas that will receive this special treatment. The top picture shows the occluded areas as light gray on the unwrapped model. The two pictures below show the areas on the model marked in red.

The second step is when the model is registered. The information regarding the overlapped polygons is read in with the depth maps. The vertices that have been labelled obstructed are not projected according to the depth map. Instead they are translated depending on how much the polygon that obstructed it was projected. In effect the hidden polygons follow the other ones around, keeping the same relative distance. Figure 3.29 shows the effect of using this method

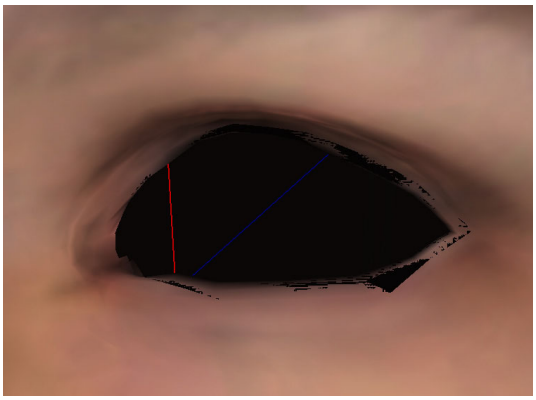


Figure 3.26: The occluded vertices project to the same depth and therefore fight to co-exist with the occluding vertices

Figure 3.27: The hidden vertices project to the same depth and therefore fight to co-exist with the occluding vertices

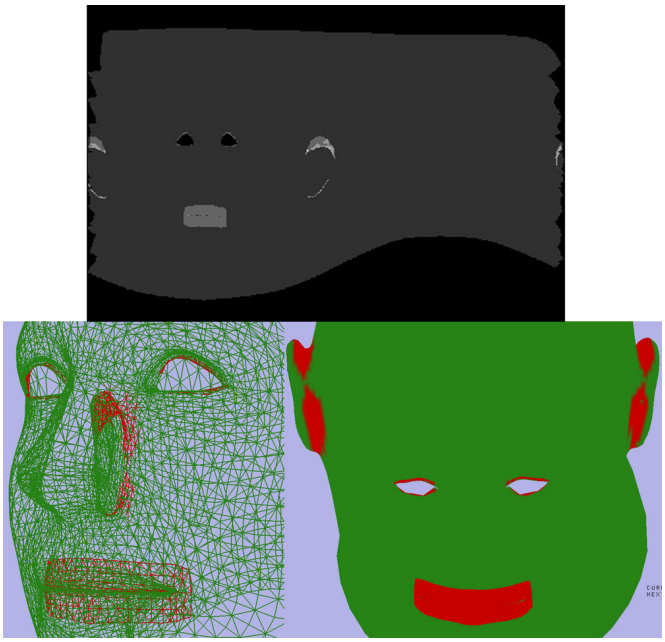


Figure 3.28: The hidden and obscured polygons are labelled and treated separately

in the registration. The problem areas have now been fixed and no black areas appear.

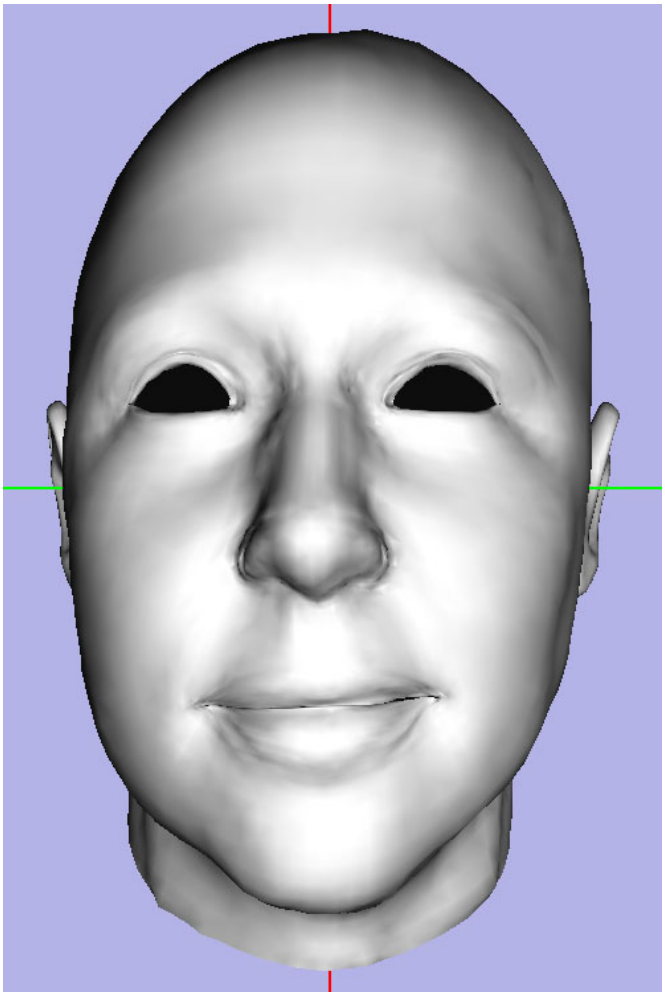


Figure 3.29: A registered model without smoothing using a 512x512 depth map

3.2.12.3 The aliasing problem

Because the depth maps are made up of pixels and not continuous floating values, a discrete sampling artefact is inevitable as some vertices map to the same UV coordinate and therefore the same depth. The problem is most apparent where there is a large amount of detail in a small area because the limited resolution cannot capture all the detailed features as they fight for a limited number of

pixels. Two regions on the face might end up sharing a single pixel in the depth map where neither region is correctly represented.

This problem is best shown using low resolution depth maps (100x100) to register the model (see Figure 3.30). The problem can be reduced by using

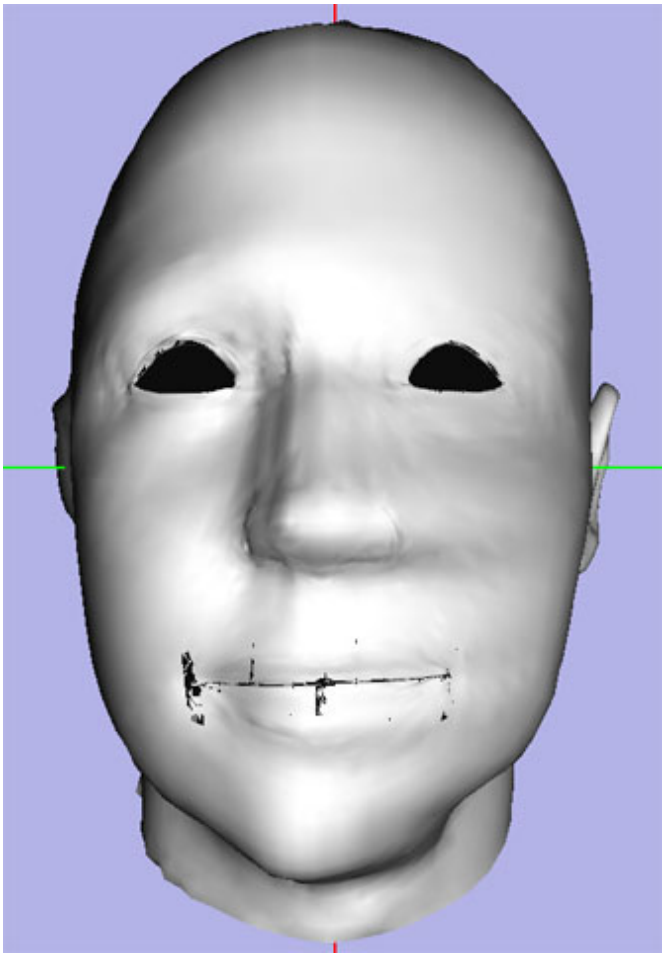


Figure 3.30: Registered model using a low resolution depth map (100x100)

maps of higher resolution, but it is still always visible and higher resolutions result in drastically slower calculations. Figure 3.29 shows the registration using a depth map of size 512x512 pixels. While clearly improved, the model looks crinkled in places. For example, the problem is particularly visible on the nose (see right picture in Figure 3.31), where it is affected both by the limited depth

map resolution in relation to the vertex density, causing the aliasing effect, and the depth information around the nose is less accurate due to scanning artefacts shown in the left picture in Figure 3.31.

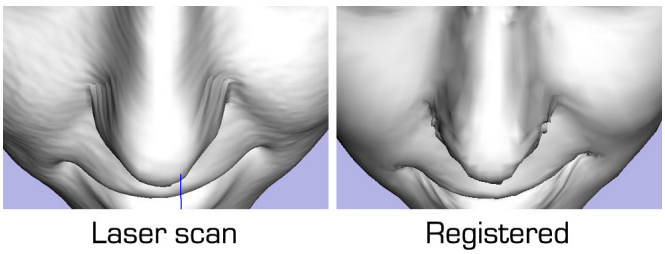


Figure 3.31: Laser scan artifacts on the nose and limited depth map resolution (512x512) result in a crude, aliased registration

Using more detailed depth maps (1024x1024) produces a marginally better registration, but the problem is still not resolved and takes considerably longer to generate and merge the depth maps, and register. The laser scanned face itself is not perfect which means the registration process can never be perfect either. Instead it is better to use a depth map of size 512x512, and apply a smoothing effect during the registration process.

The smoothing process starts by iterating through the vertices of the RBF interpolated model. The UV coordinates are calculated for each vertex and used to look up its new depth value from the appropriate UV cell. The cell is a pixel in the depth map found by rounding the UV coordinate to its nearest integer values.

Instead of using the fetched depth value for the cell directly as before, the depths in the surrounding cells are considered and used to calculate a linear combination. To increase the accuracy, the relative vertex location in the current UV cell is incorporated to assign the weights to the surrounding cells. This is shown in Figure 3.32 where the grey cell is the current cell in which the vertex UV coordinate lies, and the vertex is shown as a black dot. Its relative position in the cell is (0.2, 0.75).

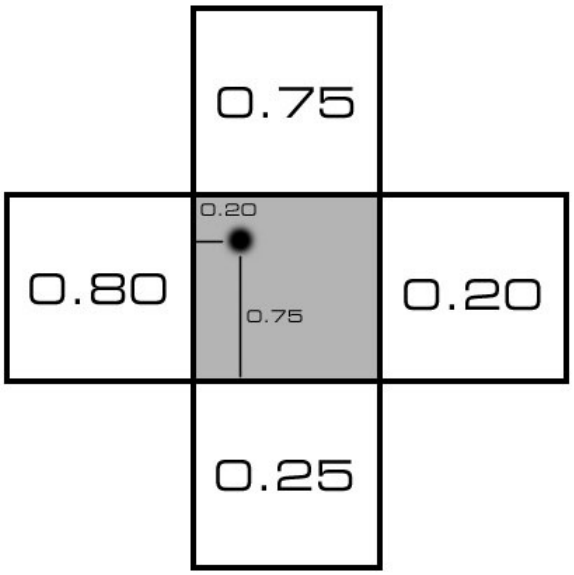


Figure 3.32: The smoothing process takes into account neighbouring cells, proportionally to the vertex location within the cell

The cells are assigned weights according to their inverse relative position. Here the left cell gets the weight 0.8, and the right 0.2. The top and bottom cells are given weights in the same fashion where the top cell gets the weight 0.75 and the bottom cell 0.25.

This produces three values: The depth value for the current cell, and weighted depth values for the left/right and top/bottom pairs. The next step is to calculate a linear interpolation using these three values where a weight coefficient is assigned controlling how much the surrounding cells influence the final depth. This is in essence a smoothing coefficient. A higher weight given to the surrounding cells equates to increased smoothing. Figures 3.33 and 3.34 show the effect of this using the smoothing coefficients 0.5 and 0.9 respectively. Figure 3.35 shows clearly how this solves the issue around the nose, and the overall model appears smoother and more natural.

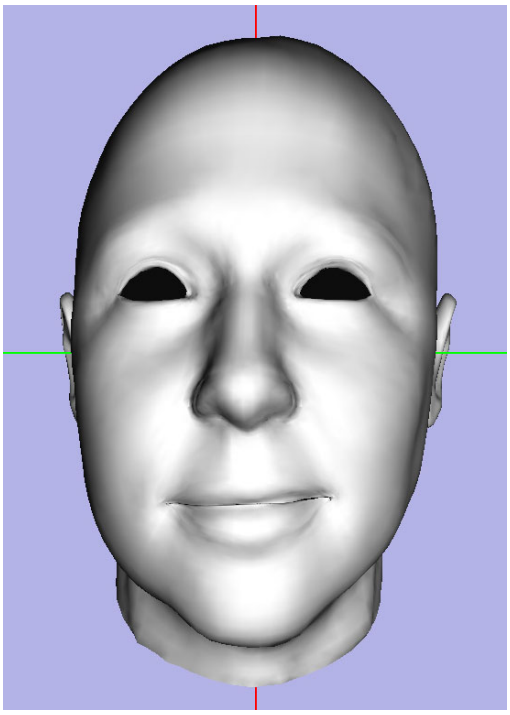


Figure 3.33: Registered model with smoothing coefficient 0.5

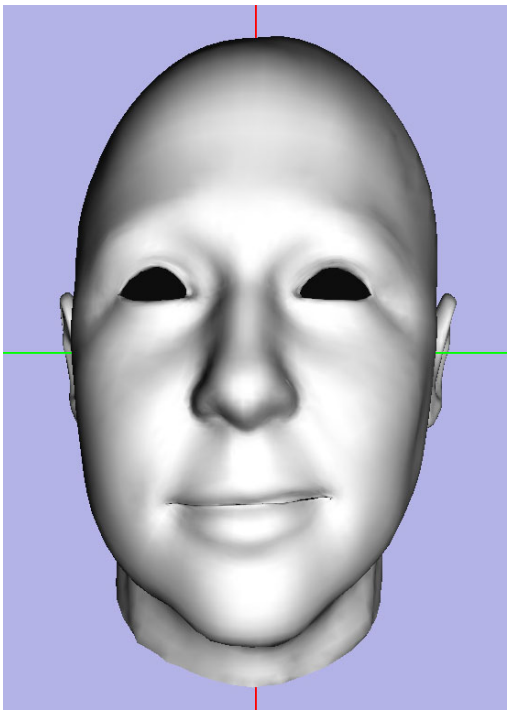


Figure 3.34: Registered model with smoothing coefficient 0.9

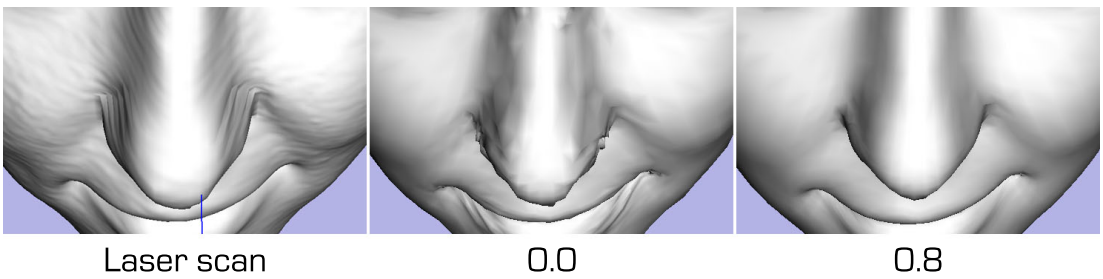


Figure 3.35: Using smoothing solves the problems inflicted by the laser scan and limited resolution. The picture in the middle is the original unsmoothed version, and the picture on the right is the registered model with smoothing coefficient 0.8

3.2.12.4 Measuring the effect of registration

Depth maps for the registered faces and the laser scan are generated and compared to see how well the the registration process reconstructs the original laser scan. Figure 3.36 shows the difference for the RBF model, and two registered models with smoothing coefficient 0.0 (no smoothing) and 0.8 (high smoothing), respectively. Applying smoothing gives a better reconstruction of the laser scan when compared to the non-smoothed registration.

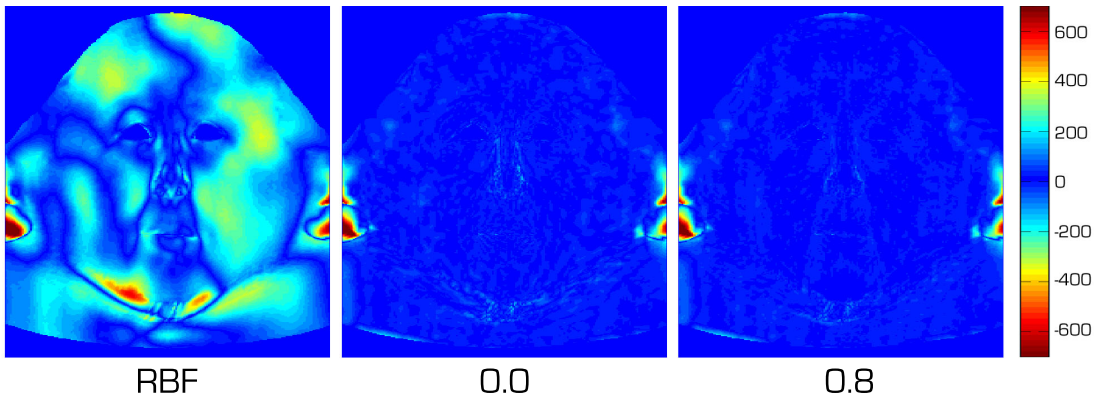


Figure 3.36: Visualisation of how well the approximated model and registered models reconstruct the original laser scan. The pictures show the difference between a standardised model and the laser scan. The picture on the left shows the RBF approximation is not a faithful reconstruction. The middle and right show that applying the fitting process creates a fairly good reconstruction, and by using the smoothing filter, an even better representation is created (Note: The ears are not processed).

3.3 Choosing appropriate data for testing and development

Both the FaceGen data, and the processed laser scan data can be used to model different faces. The generated faces from FaceGen are already standardised, sharing the same proportions, alignment and vertex topology. It could be argued they

are too artificial because they are noise free and generally lack unique details as the range of available facial features have been labelled and parameterised. However, noise can be added incrementally to see where a break down might occur. Additionally, although details are lacking and particular features are difficult to reproduce, the available features can be customised to make sure the data set incorporates a decent range of every feature. The focus here is making sure the training set contains a decent mixture of different facial features shapes without regards to ethnic groups. However, a diverse set of individuals from different ethnic groups helps to ensure a diverse range of features. The training set is made up of roughly an equal number of male and female face models, and a small group of aged faces.

These aspects can be seen as an advantage when developing a sketch-based modelling tool. In the case of inaccurately reconstructed features given a set of strokes, either the sketch detection and feature reconstruction process, or the data set could be the cause. Having a good range of noise free data reduces the risk of inaccurate reconstructions originating from a bad data set, thus putting the focus primarily on correcting the sketching and reconstruction process.

Laser scanned data might contain features that are missing in FaceGen, but there is no guarantee that the set of scanned faces includes the entire range of desired features. The data contains noise but it can be reduced by applying smoothing in the registration process, or as a post-process.

FaceGen data will be used. It is better suited for building a sketch-based interface that produces predictable and accurate results, adding registered faces from laser scans as a future extension to introduce new features and a range of details. These faces would be an add-on where they comply with the structure of the FaceGen data. This is done by registering them using a generic face from FaceGen as shown in the previous section.

The laser scanned database used here does not contain different facial expressions for a single individual. This makes FaceGen the default choice for the animation work detailed in Chapter 5.

Chapter 4

Modelling Faces by Sketching

This chapter focuses on creating new face models through sketching. An integral part of this work is relating strokes sketched by a user to relevant face data which can be used to construct a range of facial features. The approach taken here is to look at sketching process as an inverse Non-Photorealistic Rendering (NPR) where the strokes are mapped to contour data which make up a low-dimensional sketch-like representation of any 3D object (Section 4.1). Sketched strokes are often ambiguous and made up of an arbitrary number of stroke points, while the contour data is defined by contour points which represent an accurate, and non-redundant description of facial features for a particular face model. This poses the challenge of determining the best approach for mapping stroke points to the set of contour points which best describes the facial features that were intended with the sketched strokes.

Figure 4.1 gives an overview of the technique proposed in this chapter. The first thing to note is that the system consists of an online part (left), and an offline part (right). The offline part contains a face generator in the form of three statistical models (Section 4.2), trained on a collection of 3-dimensional face models described in Chapter 3. This Chapter continues to explore both the low and high resolution models although it focuses on the low resolution models in the implementation and testing of the system due to considerable difference in speed and memory usage. Additionally, curvature data is stored offline and is

used to generate contours for a range of face models at runtime, including those in the training set (Section 4.3.4).

The online part contains a simple interactive sketching interface where a template face model is used as a starting point to guide the user. A tracking model with identical vertex structure to the template model is used to keep track of the sketched features. Initially, the vertex coordinates in the tracking model are unknown but are replaced with observed values based on two modes of input supported by the interface:

- Automatically generated contour data from existing faces which simulate sketched points. This is used to verify that contours can be used to reconstruct complete 3D faces (Section 4.5.1).
- Strokes sketched by a user which is used to interactively alter one or more facial features until the desired face has been constructed (Section 4.6).

The contour generator uses the offline curvature data to create a set of contour candidates taking into account the current viewpoint and bounding box (Section 4.3). The sketched strokes are then mapped to the best set of contour candidates based on a predefined criteria where each candidate is associated to a vertex index. The indexed vertices in the tracking model are updated using the coordinates from the corresponding candidate (Section 4.5). Here, two methods for mapping sketched points to contours are explored. The first method is a heuristic approach which grades each potential point-to-contour, where the grades are calculated using linear functions based on a set of defined sketching coefficients. Each assignment takes into account the previously assigned point and contour. The second method involves Hidden Markov Models (HMM) which is a probabilistic approach where the sketching coefficients form a probability density function. This method takes every assignment in the sequence into account when determining the appropriate mapping for each point.

The partially known tracking model is passed on to the face generator which uses the already observed data to find the remaining unknown vertices to create a complete, reconstructed face model where it is adapted to comply with the face space. This creates the most probable features based on the training set,

and can optionally correlate any unsketched features (Section 4.2). The complete vertex structure is used to update the visible model in the sketching interface, and the observed vertices in the (hidden) tracking model are updated with the corresponding vertices in the reconstructed model.

This Chapter concludes by presenting an evaluation where users were given two tasks which involved sketching a specific person based on a verbal description (task 1), and from a picture (task 2).

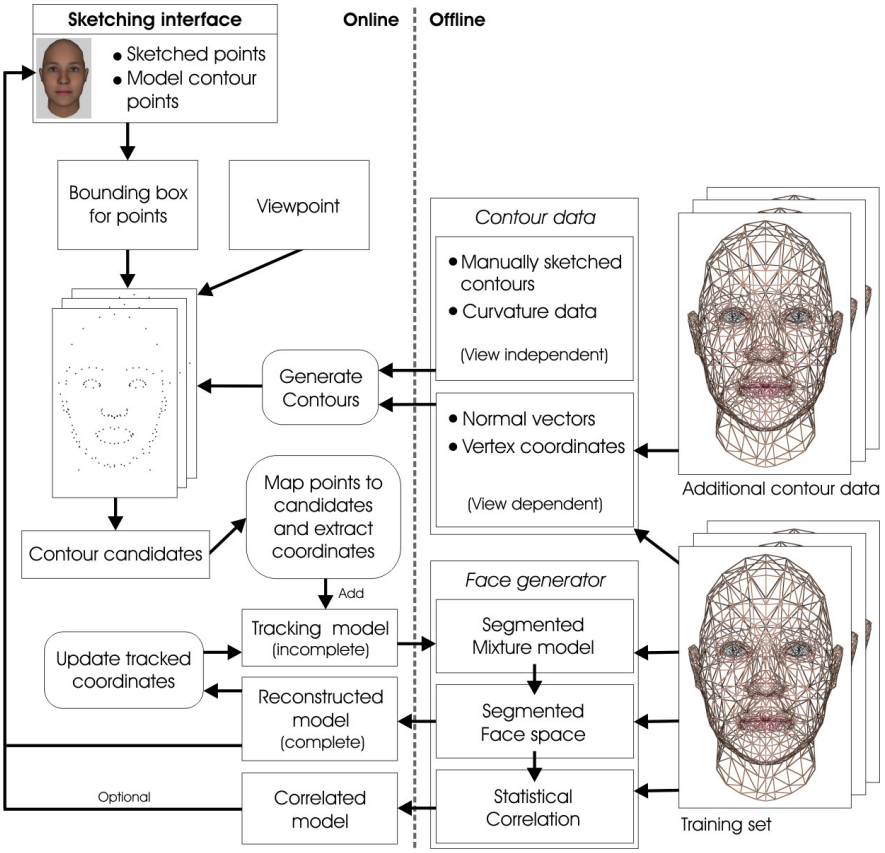


Figure 4.1: An overview of the modelling approach

4.1 Mapping sketched strokes to 3D models

This section discusses how to represent 3-dimensional faces in such a way that they can be mapped in a meaningful way to sketched strokes in order to create new

facial features based on the sketch. The first thing to consider is how the sketched strokes are stored and represented in sketching interface. Two approaches are discussed and contrasted, labelled feature points and automatically generated contour points, where the latter approach is justified.

4.1.1 Sketching interface

The strokes are first detected and stored as a sequence of 2D points in screen space. The surface normals for the points are not assigned as they are not restricted from representing any feature on the face model's surface. The depth problem is solved by projecting the points onto an intersecting object in the 3D world space where the point attains the depth from the object's surface. The possible choices of objects are displayed in Figure 4.2. The first possibility (a) allows the user to sketch directly on the underlying face mesh where the stroke points are projected onto the mesh surface if they intersect it from the viewer's perspective (in XY screen space). At least one point on the stroke has to intersect the model as the points who do not will get an interpolated depth value based on the successfully projected points. The second (b) and third (c) option overcome this limitation by including an axis plane that extends further than the mesh boundaries where the user can sketch freely. The fourth and last option (d) is a concept that is only briefly touched on in this thesis. It allows the user to sketch on a flat plane simulating traditional sketching-on-paper, where contour points for the mean face have been projected onto the plane for guidance. It could be argued that the strokes should be used entirely in screen space and avoid this projection step because it is shown in later sections that the mapping process operates primarily in the screen space. However, the depth proves useful to cull unlikely candidates in the world space to reduce ambiguity.

Sketching hastily can result in a stroke with very sparse points. As the stroke is rendered as a connected line it may seem as there is some information between sparsely connected points when in fact the line segment contains no intermediate points which can be mapped to contours. Instead of reparameterising the stroke, additional points are added between two connecting points using cardinal spline interpolation, if the distance between them exceeds a certain threshold.

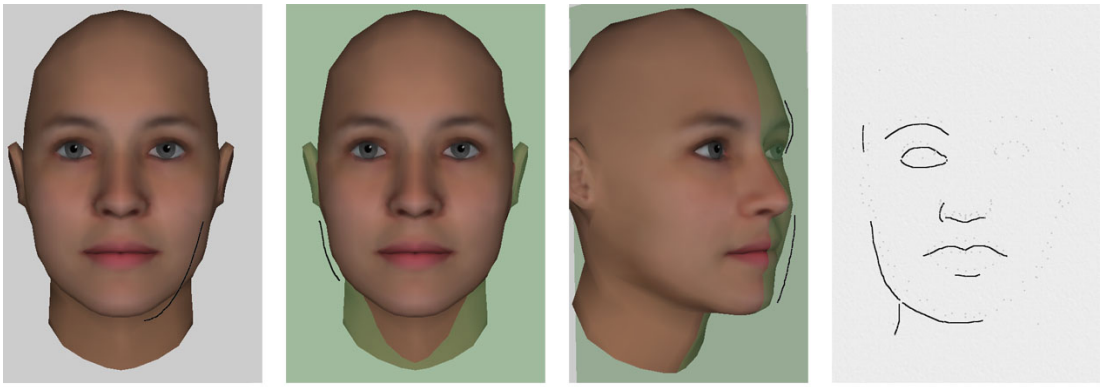


Figure 4.2: a) Sketching on mean face b) Sketching on front axis plane c) Sketching on profile axis plane d) Sketching on 2D paper plane

4.1.2 Feature point representation

Feature Points (FPs) define the boundaries of facial features using meaningful, and carefully placed points based on anthropometric landmarks (see Section 3.1). Sketched strokes outlining the main facial features will often include these FPs, particularly when considering the front view. Since the FPs can be used to generate a range of face models using RBFs, it is reasonable to assume that sketched strokes can drive this creative process by mapping the points in the stroke points to FPs.

However, the FPs are very sparse compared to the mesh density so any information carried in a stroke that lies between assigned FPs will be ignored. Also, since the FPs are view-independent, their location will never change regardless of the viewpoint angle which causes some FPs to be redundant or misleading in the context of sketching facial features from an arbitrary viewpoint. This is depicted in Figure 4.3 where the red dots represent the FPs. Examining the front view (left), the FPs circled in yellow are redundant as they only apply when seen from the profile view. These FPs will therefore put added strain on the FP mapping process as they fight other FPs for individual stroke points. The only FP specifying the outline of the overall face shape is the bottom of the chin. Without data defining the face outline it cannot be sketched, but adding more points will not solve the problem because the outline is never the same from any two viewpoints.

This is clear when looking at the same model and FPs from a different angle in Figure 4.3 (left). If a sketched line was connected through the points defining the nose area, it would give a skewed picture of its shape as the FPs do not lie along the edges of its current outlines or internal features.

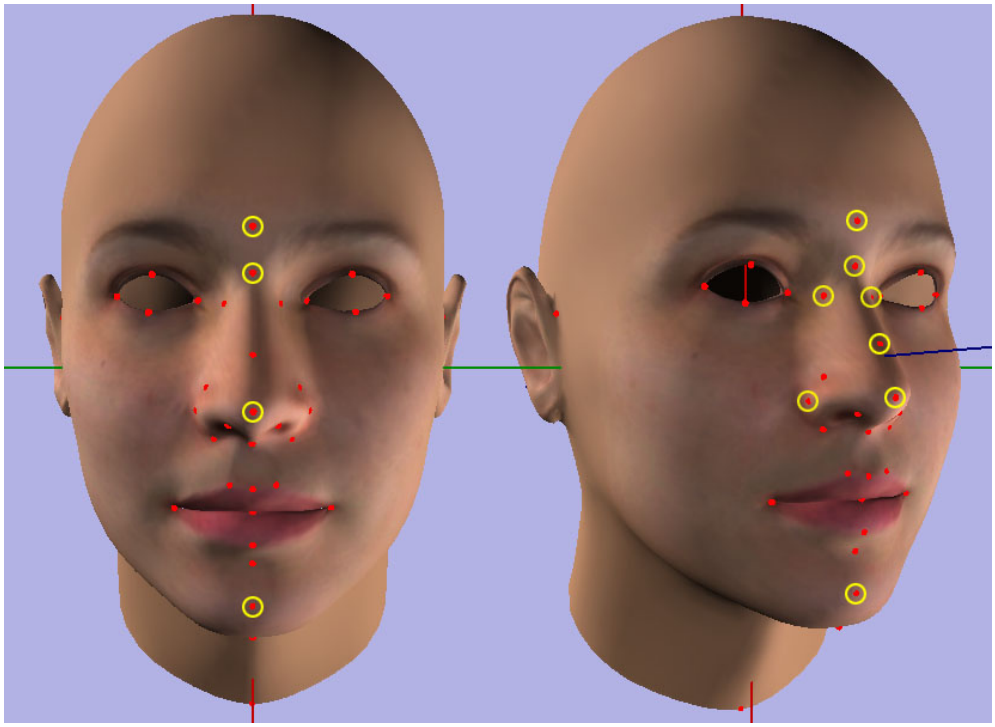


Figure 4.3: FPs become redundant or inaccurate (yellow circle) when viewed from different angles

4.1.3 Contour representation

Non-Photorealistic Rendering (NPR) methods are used to create sketch-like representations of 3D models [Can86; HZ00; PHWF01; KMM+02; DFRS03; DFR04; OBS04; JDA07; LMLH07; HMR+09]. Nealen et al [NSACO05] suggested that the process of sketching should be considered as an inverse NPR method where the sketched strokes are interpreted as a silhouette of a 3D object. Cole et al [CGL+08] lend support to this idea by comparing how real artists sketch a range

of 3-dimensional objects to recent NPR work. They find that the combined output generated by four NPR methods account for 86 percent of the sketched strokes made by artist where each method has its strengths and weaknesses. The three object space methods combined covered 81 percent, sharing 95 percent of the generated lines with the image space method, which on its own scored 76 percent. Furthermore, they examined how well a line drawing depicts the 3D shape of an object to a human viewer [CSD+09]. They found that for most objects, the NPR versions performed nearly as well as a shaded image the object. Faces were not included in their study, but it indicates that line drawings are an intuitive way to indicate the desired shape of 3D facial features.

Image space methods render the object into an image, and the use image processing methods such as edge detection to create the sketch representation [Can86; LMLH07]. They create visually pleasing results but lose the 3D scene information when rendering the object into the 2D image space. The goal of this paper is to use the generated NPR data as a vessel, carrying the sketched points to the 3D vertices so an object based approach is a more direct choice.

The silhouette of a 3D object is easily described using contours in object space. A mesh vertex p is a contour if its normal $n(p)$ is perpendicular to the view vector $v(p)$ (see more details in Section 4.3). The internal features can be described using a range of object space methods where they examine the differential geometry of the object's surface which generally involves looking at its curvature properties.

Ohtake et al [OBS04] look at ridges and valleys which occur at points of minimum and maximum curvature in the principal directions (see more on principal curvatures in Section 4.3), Judd et al [JDA07] include the viewing projection with the curvature to define view-dependent curvature creating apparent ridges, DeCarlo et al [DFRS03] create suggestive contours at points where the radial curvature is zero, and DeCarlo and Rusinkiewicz [DR07] extend that work by introducing suggestive highlights and principal highlights based on related definitions and geometric creases. These methods work well for smooth surfaces but they are sensitive to noisy data. Huang et al [HMR+09] adapt suggestive contours to work on noisy range images through filtering, and by using the analytic

connectivity information provided with the images.

Cole et al [CGL+08] suggest that a combination of methods is ideal to capture the widest range of features. Suggestive contours connect directly to the silhouette contours, and often become true contours in nearby views. They describe important areas but miss ridge-like features on convex shapes like the nose [JDA07]. Ridges [OBS04; JDA07] ridges define convex shapes but tend to be noisy, and exaggerate curvature in some cases.

The approach taken here is to use suggestive contours based on DeCarlo et al [DFRS03] while developing and testing the sketch-based interface, and leave the combination with other techniques for a more precise coverage of facial features as future work.

Examples of recent sketch-based work using contour data include object deformations by Zimmermann et al [ZNA08]. The contours are generated in image space by examining discontinuities in the depth and normal map using convolution filters which approximate the surface derivatives. The sketched contour is used to find a reference contour on the target mesh, and then a particular region of interest around the contours is deformed to fit the sketched contour using Laplacian Surface Editing, which is based on their own work on Laplacian deformation [SCOL+04]. Kraevoy et al [KSvdP09] align and deform template models to fit sketched silhouette contours by mapping stroke point to vertices on the mesh using a Hidden Markov Model (HMM) optimisation whose criteria is based on normal differences and the proximity of vertices to the stroke points. HMMs are sensitive to sequential data so the vertices are not mapped in isolation, hence making sure they are mapped with regards to the vertex structure. Their approach allows processing of multiple strokes at a time, and progressively deforms the mesh to fit the sketched contours using mean-value encoding.

This thesis focuses on using sketched contours to manipulate face models through the use of templates providing prior knowledge of their 3D structure. The templates described here differ from [KSvdP09] as they use a single template for a particular object, while here a number of templates are used for a single class of

objects, namely faces. Additionally, their method uses a progressive deformation technique to fit the model which works well, but the work here fits a face model and features using a statistical, generative model through maximum likelihood from incomplete input data. The collection of templates serve as a training set to fit the generative model.

Figure 4.4 shows the generated contours (black) for the average high resolution face model, where the data is more dense than the FPs and adapts to the current viewpoint. This enables sketching a bigger range of features with greater accuracy, and allows for a range of person-specific details that cannot be captured with a standardised set of points. Section 4.3 details how the contours are calculated.

Figure 4.4: Contours create a dense view-dependent of the relevant facial features

4.2 Learning faces

This section goes through the maths and methods behind fitting probabilistic models on a collection of face data, referred to as a training set. It is demonstrated

how a complete sample can be reconstructed given incomplete data. This is an important element as a 2D stroke can only provide limited information about the location and distribution of a small number of vertices. It is therefore pivotal to have a statistical model which is capable of creating an elaborate 3D vertex structure from the limited data which is extracted from the sketched strokes.

PCA is a popular statistical approach but suffers from a number of limitations such as not allowing a principled way of handling missing data. PCA projections based on changes from a small proportion of data points produces compromised approximations of the intended features. This is apparent by looking at the projections in the following sections where performing a projection on a complete vertex structure often compromises the reconstructed features. The strokes could be used to translate a set of vertices that will act as deformation parameters for the remaining vertices. This however will not guarantee that the reconstructed will be a realistic human face. It is shown here that using clustering and probability techniques, a realistic face model based on a known set of faces can be produced where the reconstructed features are faithful to the sketched strokes. The section concludes by exploring how different parts of a segmented face can be correlated which can be useful when sketching a face where only certain facial features are known or required.

4.2.1 Clustering and probabilistic techniques

4.2.1.1 K-medoids and K-means clustering

K-medoids and K-means are hard clustering methods. K-medoids is a heuristic method and is used here to find the initial clustering parameters for the K-means algorithm which uses the EM algorithm to converge to an optimised clustering solution [Bis07].

To find K clusters in a set of N observed D-dimensional data points, the K-medoids is executed in the following manner:

1. Create the first cluster c_1, by picking any data point as the cluster centre $\mu_\mathbf{1}$ (randomly or using specific criteria).

2. Add a new cluster c_k, $k = 2..K$, by assigning the cluster centre $\mu_\mathbf{k}$ as the data point furthest away from all other centres, $c_k = max\left(\sum_{i=1}^{K} \|\mu_k - \mu_i\|\right)$.
3. Once all the cluster centres are found, assign every data point $\mathbf{x_n}, n = 1..N$ to a cluster based on the closest centre.

The K-means algorithm minimises the objective function

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x_n} - \mu_\mathbf{k}\|^2, \tag{4.1}$$

where $r_{nk}\,\epsilon\,\{0,1\}$ defines if data point n belongs to centre k. Based on the inital values given by the K-medoids method, the EM algorithm optimises the model parameters by iteratively calculating the function J with respect to r_{nk} and $\mu_\mathbf{k}$ until values are found that minimise the function [Bis07]. This hard clustering method is not probabilistic, but is useful to initialise the parameters in a probabilistic mixture model described in Section 4.2.1.3. The mixture model introduces soft clustering which means that a data point can belong to more than one cluster where the probability factor for each potential cluster is calculated.

4.2.1.2 PPCA

Principal Components Analysis (PCA) is a popular approach in computer vision where high dimensional data is decorrelated and approximated using a lower dimensional space where each dimension is orthogonal to each other so that the retained variance under projection is maximal. However, conventional PCA suffers from many limitations. Importantly it is not a density model so it cannot be used with Bayesian inference, it cannot handle missing data, and it cannot be extended to a mixture model which is used to estimate non-linear projections.

PCA can be defined in a maximum-likelihood framework based on a Gaussian latent variable model to derive a Probabilistic PCA (PPCA) [TB99]. A latent variable model linearly maps an observed d-dimensional vector $\mathbf{t}$ to a q-dimensional, Gaussian latent variable $\mathbf{x}$ with mean vector μ (where $d \gg q$) such

that

$$\mathbf{t} = \mathbf{W}\mathbf{x} + \mu + \epsilon \tag{4.2}$$

where ϵ is a Gaussian, independent noise model $\epsilon \sim \mathbf{N}(0, \psi)$. This means that the observed vectors $\mathbf{t}$ are also Gaussian distributed, $\mathbf{t} \sim \mathbf{N}(\mu, \mathbf{C})$. By using an isotropic noise model and setting $\psi = \sigma^2\mathbf{I}$, and therefore the model covariance to $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^\mathbf{T}$, the columns of $\mathbf{W}$ span the principal subspace of $\mathbf{t}$ after fitting the model. Fitting the latent variable model can be done either in closed form or using the EM algorithm as described by Tipping and Bishop [TB99]. The distributions for a single latent variable model are summarised here from their paper as a reference for the following sections where the marginal distribution is used to calculate the posterior responsibilities and probability when handling incomplete data. The probability distribution over t-space for a given x is of the form

$$p(\mathbf{t}|\mathbf{x}) = (2\pi\sigma^2)^{-d/2} \exp\{-\frac{1}{2\sigma^2}\|\mathbf{t} - \mathbf{W}\mathbf{x} - \mu\|^2\}. \tag{4.3}$$

Using a Gaussian prior $p(\mathbf{x})$ they obtain the marginal distribution

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{x})p(\mathbf{x})\mathbf{dx} = (2\pi)^{-d/2}|C|^{-1/2} \exp\{-\frac{1}{2}(\mathbf{t} - \mu)^T C^{-1}(\mathbf{t} - \mu)\}. \tag{4.4}$$

The Bayes rule gives the posterior distribution $p(\mathbf{x}|\mathbf{t})$, and the log-likelihood of observing the data is

$$L = -\frac{N}{2}\{d\ln(2\pi) + ln|\mathbf{C}| + tr(\mathbf{C}^{-1}\mathbf{S})\}, \tag{4.5}$$

where the sample covariance matrix $\mathbf{S}$ is found with

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{t_n} - \mu)(\mathbf{t_n} - \mu)^T. \tag{4.6}$$

4.2.1.3 Mixtures of PPCA

The high-dimensional training data is generally found to be non-linear and that is indeed the case here. Using a single PPCA model to fit a univariate Gaussian function on the data set creates an unrealistic likelihood function. If an imaginary data were to form two clusters, a non-existing data point lying between the clusters would be seen as equally likely as an actual observed point in either cluster.

A multi-variate approach is needed to fit a non-linear data set. A Gaussian mixture model approximates a non-linear model by expressing the probability density function as a linear combination of basis functions

$$\mathbf{p}(\mathbf{t}) = \sum_{k=1}^{K} \pi_{\mathbf{k}} \mathbf{p}(\mathbf{t}|\mathbf{k}), \tag{4.7}$$

where $\mathbf{p}(\mathbf{t}|\mathbf{k})$ is a single PPCA model and $\pi_{\mathbf{k}}$ is the mixing coefficient or prior probability for component i, $\pi_{\mathbf{k}} \geq 0$, and $\sum \pi_k = 1$. $\mathbf{t}$ is the observed input vector, m is the number of clusters or centres, and $\mathbf{p}(\mathbf{t}|\mathbf{k})$ is the cluster density function.

The parameters for this mixture model can be determined by maximising the data likelihood. For convenience, the problem is converted into an equivalent form where the goal is to minimise the negative log-likelihood which is treated like an error function. This cannot be calculated in closed form so the EM algorithm is employed to optimise the model parameters. Tipping and Bishop [TB99] detail the steps needed to fit the model parameters, but the following is a summary of the steps that were taken.

The E-step calculates the following:
$R_{nk} = p(k|\mathbf{t_n})$ is the posterior responsibility of mixture k for generating data point $\mathbf{t_n}$ given by

$$R_{nk} = \frac{p(\mathbf{t_n}|k)\pi_k}{p(\mathbf{t_n})}. \tag{4.8}$$

The 'old' values are used to evaluate the expectations $\langle \mathbf{x_{nk}} \rangle$ and $\langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle$

$$\langle \mathbf{x_{nk}} \rangle = \mathbf{M_k^{-1}} \mathbf{W_k^T} (\mathbf{t_n} - \mu_\mathbf{k}), \tag{4.9}$$

$$\langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle = \sigma_{k^2} \mathbf{M_k^{-1}} + \langle \mathbf{x_{nk}} \rangle \langle \mathbf{x_{nk}} \rangle^T, \tag{4.10}$$

where

$$\mathbf{M_k} = \sigma^2 \mathbf{I} + \mathbf{W_k^T} \mathbf{W_k}. \tag{4.11}$$

The complete likelihood is

$$\begin{aligned} \langle L_C \rangle &= \sum_{n=1}^{N} \sum_{k=1}^{K} R_{nk} \{ \ln \pi_k - \frac{d}{2} \ln \sigma_k^2 - \frac{1}{2} tr(\langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle) \\ &\quad - \frac{1}{2\sigma_k^2} \| \mathbf{t_{nk}} - \mu_\mathbf{k} \|^2 + \frac{1}{\sigma_k^2} \mathbf{W_k^T} \mathbf{W_k^T} (\mathbf{t_n} - \mu_\mathbf{k}) \\ &\quad - \frac{1}{2\sigma_k^2} tr(\mathbf{W_k^T} \mathbf{W_k} \langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle) \}, \end{aligned}$$

where $tr(\cdot)$ is the trace of a matrix which is the sum of its diagonal elements.

The M-step calculates the 'new' estimated values denoted with the symbol $\tilde{}$:

$$\begin{aligned} \tilde{\pi}_k &= \frac{1}{N} \sum_n R_{nk}, \\ \tilde{\mu}_\mathbf{i} &= \frac{\sum_n R_{nk} (\mathbf{t_{nk}} - \widetilde{\mathbf{W}}_\mathbf{k} \langle \mathbf{x_{nk}} \rangle)}{\sum_n R_{nk}}, \\ \widetilde{\mathbf{W}}_\mathbf{k} &= \left[\sum_n R_{nk} (\mathbf{t_n} - \tilde{\mu}_\mathbf{k}) \langle \mathbf{x_{nk}} \rangle^T \right] \left[\sum_n R_{nk} \langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle \right]^{-1}, \\ \tilde{\sigma}_k^2 &= \frac{1}{d \sum_n R_{nk}} \left\{ \sum_n R_{nk} \| \mathbf{t_n} - \tilde{\mu}_\mathbf{k} \|^2 \right. \\ &\quad - 2 \sum_n R_{nk} \langle \mathbf{x_{nk}} \rangle^T \widetilde{\mathbf{W}}_\mathbf{k}^\mathbf{T} (\mathbf{t_n} - \tilde{\mu}_\mathbf{k}) \\ &\quad \left. + \sum_n R_{nk} tr(\langle \mathbf{x_{nk}} \mathbf{x_{nk}^T} \rangle \widetilde{\mathbf{W}}_\mathbf{k}^T \widetilde{\mathbf{W}}_\mathbf{k}) \right. \end{aligned}$$

(4.13)

If a covariance matrix values drop below a threshold ϵ which is set to machine precision, the matrix is reset to its initial status. This is because if a covariance matrix collapses, the next iteration of the EM algorithm fails and the model parameters are no longer floating point numbers [Nab01]. Poorly initialised parameters can result in a local maxima problem as there are generally multiple local maxima of the log likelihood function. To reduce the chances of that happening the K-medoids and K-means methods are used to perform initial clustering to provide the mixture model with initial settings (see Section 4.2.1.1). This also reduces the number of iteration needed when fitting the mixture model.

4.2.1.4 Computing probabilities

The marginal likelihood for each cluster i is calculated using Cholesky decomposition to speed the computation [Nab01]. The decomposition is applied on the covariance matrix $\mathbf{C_k}$ to find the upper triangular matrix $\mathbf{U}$ which satisfies

$$\mathbf{U}^T\mathbf{U} = \mathbf{C_k}. \tag{4.14}$$

The marginal likelihood for an arbitrary cluster derived from equation 4.4 is then given by

$$p(t) = \exp\left\{\frac{1}{2}\sum^{*}\left[\left(\frac{\mathbf{x_n} - \mu_\mathbf{k}}{\mathbf{U}}\right) \otimes \left(\frac{\mathbf{x_n} - \mu_\mathbf{k}}{\mathbf{U}}\right)\right] \oslash \left[(2\pi)^{\frac{-D}{2}} \prod_{j=1}^{D} \mathbf{U_{jj}}\right]\right\}, \tag{4.15}$$

where $\sum^{*}$ calculates the row sums, $\otimes$ denotes element-by-element multiplication, and $\oslash$ denotes element-by-element division.

Finding the posterior responsibility R_{nk} for a data point $\mathbf{t_n}$ is straightforward using the marginal likelihood for every cluster $k = 1..K$, and applying equation 4.8. Once complete, an K-dimensional vector consists of the responsibility values which indicate the probabilities of a point t_n belonging to each of the clusters, where $\sum_{k=1}^{K} R_{nk} = 1$. Mapping a data point to the most likely cluster is then

simply finding the cluster with the highest posterior value in the vector.

For a data point $\mathbf{t_n}$, the probability ρ for cluster k is the marginal likelihood for the cluster, scaled with its mixture component prior

$$\rho_k = \mathbf{p}(\mathbf{t_n}|k)\pi_k. \tag{4.16}$$

This is useful when choosing between two or more competing data points, such as points in a sketched stroke, where the one with highest probability value is chosen.

4.2.1.5 Dual PPCA

The training set $\mathbf{t}$ consists of a $n \times d$ data vector containing n sample faces. When using PPCA and d is very large, the sample covariance matrix of size $d \times d$ becomes a performance hindrance. Finding the sample covariance requires $O(nd^2)$ operations, and finding its eigenvectors requires $O(d^3)$ operations. This can be calculated iteratively by applying the EM algorithm [TB97] which is an approach that was taken initially in this thesis. Alternatively, this can be achieved in closed form using the dual approach to PPCA, where instead of marginalising the latent variables $\mathbf{X}$ and optimising the parameters $\mathbf{W}$ via maximum likelihood, the parameters are marginalised and optimising with respect to the latent variables [Law05].

The mean adjusted vector Y is found, and then instead of calculating the sample covariance $\mathbf{S} = \mathbf{Y}^T\mathbf{Y}$, the following steps are taken:

$$\begin{aligned} \mathbf{Y} &= \mathbf{t} - \mu \\ \mathbf{K} &= \mathbf{Y}\mathbf{Y}^T/d. \end{aligned} \tag{4.17}$$

The eigen-decomposition $\mathbf{U}$ of $\mathbf{K}$ is found, and the eigenvectors for the sample covariance $\mathbf{S}$ is recovered with

$$\mathbf{U_S} = \mathbf{Y}^T\mathbf{U}\mathbf{V}, \tag{4.18}$$

where $\mathbf{V}$ is the diagonal of the eigenvalues λ.

Choosing q principal eigenvectors, the latent variables x for an arbitrary sample t_s are found in the same way as in traditional PPCA

$$\begin{aligned} &\mathbf{x} = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{t_s} - \mu), \\ &\mathbf{M}^{-1} = \sigma^2\mathbf{I} + \mathbf{W}^T\mathbf{W}, \\ &\mathbf{W} = \mathbf{U_S}(\Lambda - \sigma^2\mathbf{I})^{\frac{1}{2}}\mathbf{R}, \\ &\sigma^2 = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j, \end{aligned} \tag{4.19}$$

where Λ is a $q \times q$ diagonal matrix containing the q eigenvalues $\lambda_1, .., \lambda_q$, and $\mathbf{R}$ is an arbitrary $q \times q$ orthogonal rotation matrix [TB97]. The reconstructed sample η is found using

$$\eta = \mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{M}\mathbf{x} + \mu. \tag{4.20}$$

4.2.1.6 Handling incomplete data

It was demonstrated in Figure 4.1 that sketched strokes map to contours which update the tracking model which consists of vertices with unknown coordinates. It is unrealistic to assume that the user sketches every aspect of every feature on the face model, and here the eyeballs themselves cannot be sketched (only the eye sockets). Furthermore, the strokes can only map to contours which at any time represent a small portion of the face model from a given viewpoint which means the observed vertex structure is always incomplete. The missing vertices can be found using known vertices obtained from the observed strokes using the method described here.

Let $\mathbf{t}$ be the sample vector for the whole vertex structure (or more generally any structure), made of up the observed and missing data which trivially form the observed (o) and missing (m) partitions. An example of this is visualised in Figure 4.5 where the sample vector is partitioned into observed (green) and missing (red) data. The same partitioning is applied to the mean vector $\mu_\mathbf{k}, k = 1..K$, and the covariance contains four partitions to account for the row and column

dimensions.

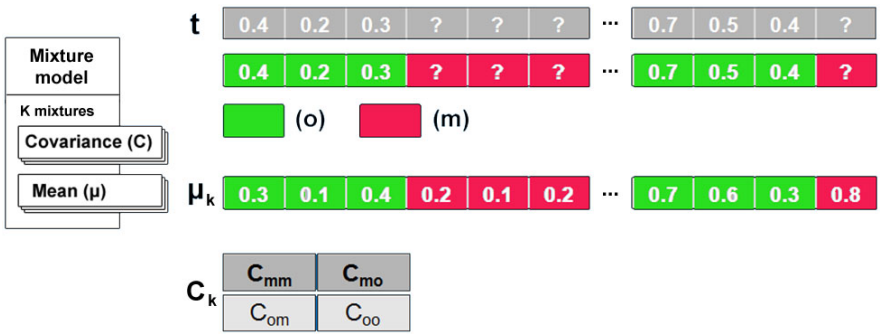


Figure 4.5: Partitioning the data into observed (conditional) and missing components: o=observed; m=missing;

The expected values for the missing data $\mathbf{t_m}$ are found using the conditional distribution $\mathbf{p}(\mathbf{t_o}|\mathbf{t_m})$ where

$$\mathbf{t} = \begin{pmatrix} \mathbf{t_m} \\ \mathbf{t_o} \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_\mathbf{m} \\ \mu_\mathbf{o} \end{pmatrix}, \quad \mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{mm} & \mathbf{\Lambda}_{mo} \\ \mathbf{\Lambda}_{om} & \mathbf{\Lambda}_{oo} \end{pmatrix},$$

$$\begin{aligned} \mathbf{\Lambda}_{mm} &= (\mathbf{C_{mm}} - \mathbf{C_{mo}}\mathbf{C_{oo}}^{-1}\mathbf{C_{om}})^{-1}, \\ \mathbf{\Lambda}_{mo} &= -(\mathbf{C_{mm}} - \mathbf{C_{mo}}\mathbf{C_{oo}}^{-1}\mathbf{C_{om}})^{-1}\mathbf{C_{mo}}\mathbf{C_{oo}}^{-1}, \end{aligned} \tag{4.21}$$

$$\mathbf{t_m} = \mu_{\mathbf{m}|\mathbf{o}} = \mu_\mathbf{m} - \Lambda_{mm}^{-1}\Lambda_{mo}(\mathbf{t_o} - \mu_\mathbf{o}),$$

and $\mathbf{\Lambda} \equiv \mathbf{C^{-1}}$ is known as the precision matrix. oo, om, mo, and mm correspond to the combinations of partitioning the observed and missing [row][column] entries in the matrices $\mathbf{\Lambda}$ and $\mathbf{C}$.

This is done for every mixture component k using the corresponding mean and covariance. The complete sample for k is found by concatenating the observed data with the expected data, and is referred to as a reconstructed sample (or reconstruction). The probability ρ_k is then calculated using equation 4.16. The reconstruction tied with $max(\{\rho_1, .., \rho_K\})$ is selected and used to update the face model. It is worth noting that the expected values get skewed towards the mean if the amount of observed data is very limited.

4.2.2 Segmenting the face

It was shown in Section 4.1 that contours are used to map sketched strokes to vertices. Faces in the training set might possess different facial details that are not shared in others, and every facial feature looks different depending on which viewpoint it is seen from (view dependent). This means that the range of potential contours can map to any vertex given different contexts. Therefore to reserve the possibility of sketching any feature and possible detail the contours represent, no vertex can be discarded. This leads to the use of the entire high-dimensional vertex structure.

Fitting a mixture model on n, d-dimensional training samples can cause singularity errors when calculating the covariance matrix if $d \gg n$. If d is very large then adding more samples to overcome the singularity problem can lead to memory issues when attempting to store and manipulate a full covariance matrix of size $d \times d$ for each mixture component.

182 low resolution training faces were used to fit a mixture model for the tests performed in this section. Each face consists of 863 3-dimensional vertices (face mask and eyes), making each sample vector 2589 dimensional. Since $2589 \gg 182$, singularity errors occur in the fitting process. Furthermore, m mixture components are needed to fit the data to a degree where it can be accurately generated from incomplete data, requiring storage for at least $m \times 2589 \times 2589$ single or double values which is more than the hardware used here can handle. Equivalent high resolution models contain 6174 vertices which means each sample vector has 18522 dimensions.

The dimensionality of the training samples could be reduced using methods such as PCA, but it remains unsolved how to relate, in a principled way, sketched 3-dimensional coordinates to a set of q-dimensional principal components generated from a set of n, d-dimensional samples of face models.

Another possibility would be to use a subset of vertices that relate to the main features. But that means facial details that might be a part of some train-

ing samples are not picked up. The ability to detect person-specific features and sketch them is one of the objectives for using the contour data insted of the FPs which only targets predefined features.

It is argued that the most practical solution for this problem is to segment the face model into k vertex groups where each group contains $d_i, i = 1..k$ vertices which is below a threshold value which offers a low risk of singularity occurring. Instead of fitting a mixture model on the whole data set, a separate mixture model is fitted on each group where the data set is restructured to form the subset of vertices from every face in the training set as specified by the group's indexing.

Figure 4.6 shows how the low resolution face has been segmented into seven groups and colour coded. This is done by applying k-means as explained in Section 4.2.1.1 where the tip of the nose is assigned as the centre for the first cluster. The clusters are then segmented again using k-means restricting the clustering along the vertical axis creating horizontal slices. Here, all clusters that lie on the same vertical level are merged except the ones at eye level. Clusters sharing a border and contain vertices below the threshold value when combined are also merged. This was done so the left and right side of the face correlate as they belong to the same segment avoiding having to sketch both sides every time. This makes the sketching process quicker and more natural since faces are generally nearly symmetrical.

Some issues arise from this horizontal slice arrangement. A particular nose shape can affect the silhouette or width of the face which can make sketching a particular combination of features cumbersome. It could be argued that a better arrangement isolates the main features even further. In both cases, only the segments corresponding to the sketched strokes are updated, leaving the others unchanged. This can create visible border issues between segments where the shared vertices and their surrounding polygons are considerably different. This is particularly apparent when using a large number of segments for isolated features which is the case when fitting and reconstructing the high resolution models. For

example, if a specific portion of the lips is sketched and reconstructed, the surrounding segments have to take into account how the connecting vertices resonate the new configuration.

A simple way to adapt the segments and correct the borders is to project the reconstructed segments onto the statistical segment space which uses the training set to create the closest approximation to the reconstructed segment (Section 4.2.4).

The entire face structure can then be projected onto the face space which smooths out the borders. However, this risks compromising the structure of individual features if the training set is limited in size. Another approach is to correlate any unaffected segments based on the reconstructed segments to make sure they adapt to the new structure (Section 4.2.5). This will also correlate any unsketched features, e.g. if the nose is sketched then the eyes will too if they have not yet been sketched.

The correlation can fail on two occasions. Correlated features are not necessarily what the user wants although this could be used to lookup faces in a database. If two neighbouring segments contain sketched information then they would have to iteratively correlate to each other to ensure smooth borders. Instead, the borders can be smoothed by applying a multi-resolution decomposition and blending technique [BA83] as suggested by Albrecht et al [ABHS06]. This approach is not implemented here, and instead is left as potential future work.

4.2.3 Fitting segments

The previous section showed how a d-dimensional vertex structure for the whole face is segmented into k groups and reformed to form the vertex structures $n \times d_i, d_i \epsilon \mathbb{R}^3$.

A mixture model is fitted on each vertex structure creating k mixture models corresponding to each group. The number of mixture components used for individual groups is decided by fitting the model using different number of components, and then picking the one with the highest complete likelihood.

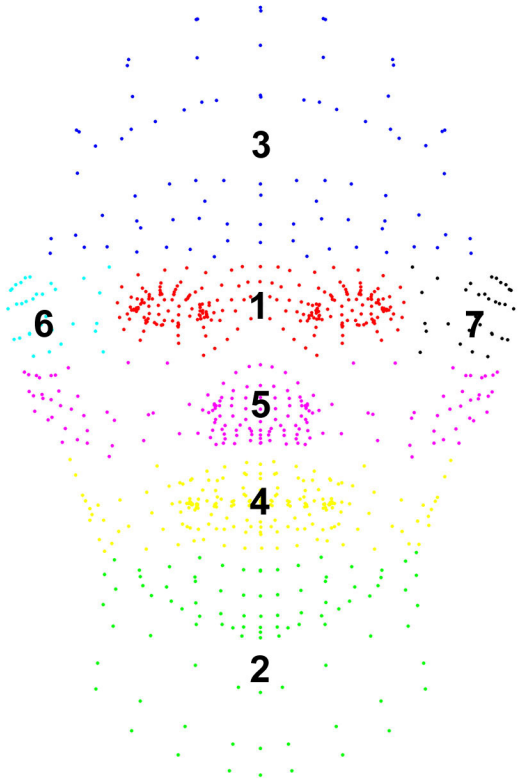


Figure 4.6: Segmented groups for the low resolution models.

The low resolution models use $k = 7$ with 182 training samples. The training parameters are displayed in Table 4.2.3 with the number of mixture components, and dimensionality (d) of each training sample vector. Once the training is complete, the data for the generative model is stored in a file which is loaded into memory at startup. The file size for the low resolution model is 338MB.

The high resolution models use $k = 41$ with 160 training samples. The number of mixture components range from 3 to 6, and d ranging from 195 to 774. The file size for the high resolution generative model is 667MB.

The low resolution models are used throughout this chapter to test the sketching interface as they require less memory, and the performance is significantly better at this stage.

Group	No. mixtures	d
1	17	735
2	15	273
3	19	249
4	11	639
5	17	501
6	6	96
7	6	96

Table 4.1: Segmented groups for the low resolution models and their training parameters

4.2.4 Segment clean-up

Irregularities in the vertex structure can occur during the mapping process due to the following reasons:

- The sketched points map to sequence of contours whose vertices violate the mesh structure, or a sequence containing unexpected sharp changes (See Figure 4.7). Improving the accuracy in the mapping approach reduces these unwanted artefacts. Furthermore, this can cause the expected data found using equation 4.21 to contain unwanted noise and unrealistic features.

- Features can be sketched repeatedly from different angles which can cause different stroke points to compete for the same vertices which can cause conflicting coordinates.

This is corrected by fitting a Dual PPCA model on the restructured training set from each of the k groups found in Section 4.2.2. Each group i contains $n \times d_i$ vertices where n is the number of training samples. This creates k Dual PPCA models where each contains p_i latent variables.

If a user sketches a stroke that maps to vertices in one or more groups, the mapped vertices are used to find the remaining unknown vertices in the affected groups. Once found, the complete set of vertices in each group are projected onto the corresponding Dual PPCA model, and then reconstructed using equation

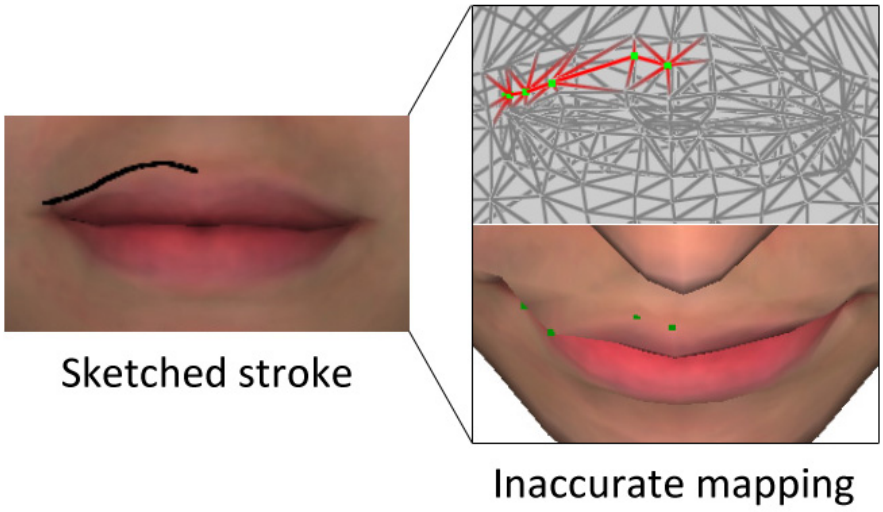


Figure 4.7: Mapped vertices containing unwanted sharp changes.

4.20. This projection onto the latent space generates a smooth surface which is the closest approximation to the original surface based on the observed faces in the training set.

4.2.5 Correlating segmented groups

Based on the training set, changes made in one group can indicate how the other groups should correlate. This is done by employing a hierarchical approach where a mixture model is fitted on latent variables generated by the k Dual PPCA models described in the previous section. Figure 4.8 shows how the latent variables are joined together to create the training set for the mixture model. The latent variables for each training sample are calculated and form a $n \times p_i, i = 1..k$ matrix. They are then concatenated to form a $n \times \sum_{i=1}^{k} p_i$ matrix which will be referred to as the correlation matrix. Each row in this matrix is therefore the complete set of latent variables, forming a feature vector for each training sample. The correlation matrix, containing the feature vectors for every sample in the original training set, makes up a training set for a mixture model which is responsible for correlating the missing parts of an incomplete feature vector.

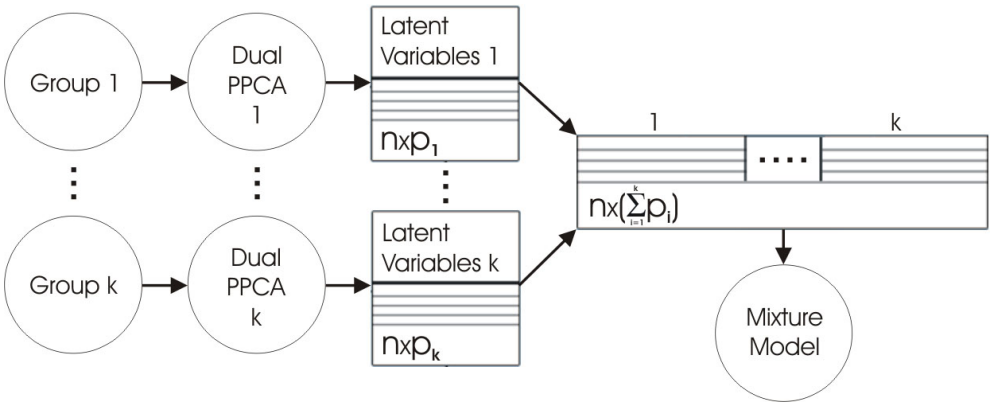


Figure 4.8: A statistical mapping technique performs segment correlation

Given that a group g is affected by a set of sketched strokes where the vertex structure is updated accordingly, the online process is as follows:

1. Calculate the latent variables for the new structure using Dual PPCA g.
2. Form a vector which corresponds to a single row in the correlation matrix where the known latent variables for group g are filled in and the remaining left blank.
3. Use the correlation mixture model to find the missing latent variables accounting for groups $i = 1..k, i \neq g$, using equation 4.21.
4. Calculate the vertex coordinates based on the expected latent variables using the corresponding Dual PPCA models using equation 4.20.

4.3 Calculating contour points

Contours are curves that represent the boundaries of objects and some of their internal features. The aim here is to calculate the contours of an arbitrary face model in the training set. However, the models consist of discrete vertices so the contours here will be made up of discrete contour points. The contours are independent, i.e. they do not form polylines using a secondary data set such as a connectivity chain. However, the contours map to the face vertex structure which can be used to examine paths between contours (see Section 4.8.4). A contour

and a set of contour points will be used interchangeable for the purposes of this thesis.

A set of contour points represent a low-dimensional vertex version of the full face mesh. Which contours are used is based on the current viewpoint, creating a view-dependent sketch-like representation of the face. Because the contours know which vertex they belong to, they serve the purpose of acting as a bridge between a mesh vertex and a sketched point. This way they interpret the sketched strokes and supply the mapped vertices with their coordinates to create new features. Three types of contours are used (see Figure 4.9):

- Silhouette contours (b)
- Suggestive contours (c)
- Manually labelled contours (d)

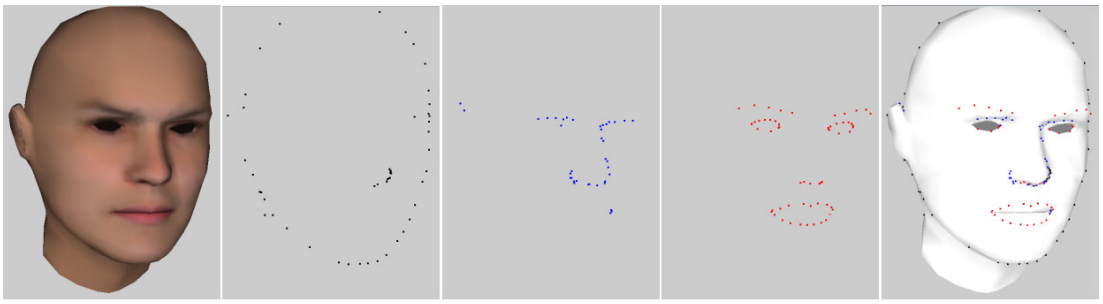


Figure 4.9: a) Low resolution model b) Silhouette contours c) Suggestive contours d) Manually labelled contours e) Combined contours

4.3.1 Silhouette contours

Finding the vertices that represent the silhouette contours for a surface S is straightforward. A vertex $\mathbf{p} \epsilon S$ is a contour point if its normal $\mathbf{n}(\mathbf{p})$ is perpendicular to the view vector $\mathbf{v}(\mathbf{p})$, or more formally if

$$\mathbf{n}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{p}) = 0 \tag{4.22}$$

and the view vector is defined as $\mathbf{v}(\mathbf{p}) = \mathbf{c} - \mathbf{p}$, where $\mathbf{c}$ is the vector coordinates for the camera centre. In practice, a small threshold value close to zero is used because the vertex structure is not continuous. The selected threshold value is based on the number of vertices in the face models. A low-dimensional model requires a higher threshold than a high-dimensional model. The value 0.14 is used here for the low-dimensional models.

4.3.2 Suggestive contours

The suggestive contours are defined as curves along which the radial curvature κ_r is zero, and the surface bends away from the viewer [DFRS03]. They can also be described as vertices which are almost silhouette contours, and will become true silhouette contours in near viewpoints.

The radial curvature κ_r is the normal curvature κ_n on the surface in the direction of the vector $\mathbf{w}$. Figure 4.10 visualises the vector $\mathbf{w}$ where it is the projection of the view vector $\mathbf{v}$ onto the tangent plane t at the point $\mathbf{p}$.

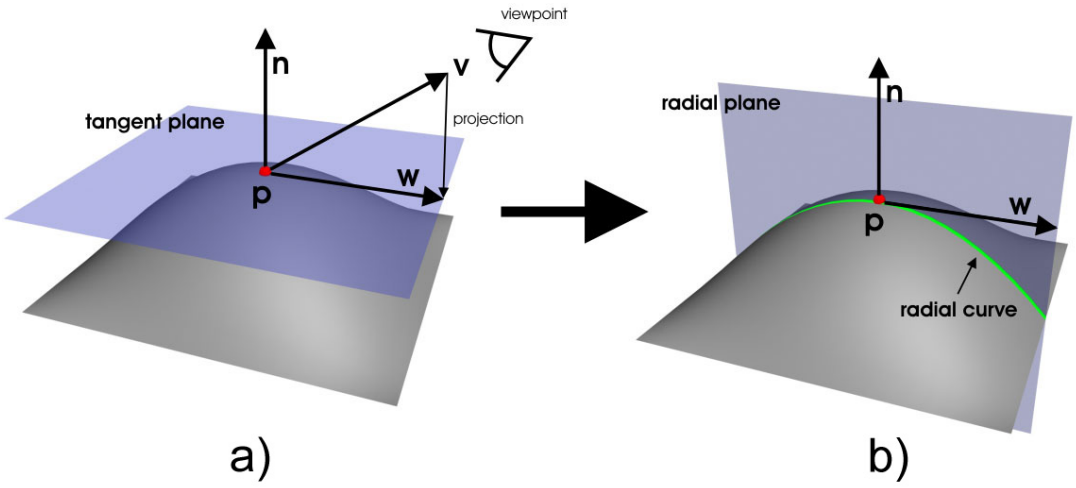


Figure 4.10: (a) The view vector $\mathbf{v}$ is projected onto the tangent plane to obtain $\mathbf{w}$. (b) The radial plane is formed by $\mathbf{p}$, $\mathbf{n}$ and $\mathbf{w}$ and slices the surface along the radial curve, the curvature of which is $\kappa_r(\mathbf{p})$ (after [DFRS03])

The maximum normal curvature κ_1 and the minimum normal curvature κ_2 are called the principal curvature at $\mathbf{p}$. The unit directions $\mathbf{e}_1$ and $\mathbf{e}_2$ for which these

maximum and minimum values are reached are called the principal directions, where they form an orthonormal basis in the tangent plane for $\mathbf{p}$ (Figure 4.10) [DC76].

The normal curvature κ_n along $\mathbf{w}$ where φ is the angle from the principal curvature direction $\mathbf{e}_1$ and $\mathbf{w}$ in the orientation of the tangent plane t to S at $\mathbf{p}$ gives the radial curvature κ_r, and is calculated using the Euler formula

$$\kappa_r(\mathbf{p}) = \kappa_1 \cos^2 \varphi + \kappa_2 \sin^2 \varphi. \tag{4.23}$$

This requires the principal curvatures and directions which are calculated using a method proposed by Taubin [Tau95] where he shows they can be obtained by computing the eigenvalues and eigenvectors of a matrix M_p defined by integral formulas. Furthermore, he provides an algorithm of linear complexity to perform this estimation. The symmetric 3×3 matrix is defined as

$$M_p = \frac{1}{2\pi} \int_{\pi}^{-\pi} \kappa_r(\mathbf{t}_\varphi) \mathbf{t}_\varphi \mathbf{t}_\varphi^T d\varphi \tag{4.24}$$

where $\mathbf{t}_\varphi = \cos\varphi \mathbf{e_1} + \sin\varphi \mathbf{e_2}$. Taubin shows that M_p can be factorised as follows:

$$M_p = t_{12}^T \begin{pmatrix} m_p^{11} & m_p^{12} \\ m_p^{21} & m_p^{22} \end{pmatrix} t_{12}, \tag{4.25}$$

where $t_{12} = \mathbf{e_1}, \mathbf{e_2}$. He finds that $m_p^{12} = m_p^{21} = 0$ so the eigenvalues for M_p are 0, m^{11}, and m^{22} corresponding to the eigenvectors $\mathbf{n}$, $\mathbf{e_1}$, and $\mathbf{e_2}$. The principal curvatures can be obtained by calculating

$$\kappa_1 = 3m_p^{11} - m_p^{22}$$
$$\kappa_2 = 3m_p^{22} - m_p^{11}.$$

The directional derivative of κ_r in the direction of $\mathbf{w}$ can be approximated as follows:

$$\kappa_r(\mathbf{w}) \approx \frac{2\mathbf{n}^T(\mathbf{q} - \mathbf{p})}{\|\mathbf{q} - \mathbf{p}\|^2}. \tag{4.26}$$

Now, a vertex is a suggestive contour if its radial curvature κ_r is 0, and the directional derivative of κ_r in the direction of $\mathbf{w}$ is positive [DFRS03]

$$D_{\mathbf{w}}\kappa_r > 0. \tag{4.27}$$

Filtering is applied to remove unwanted noise such as zero crossings which can be seen when $\mathbf{w}$ looks down its corresponding principal direction and the minimum principal curvature is close to zero [DFRS03].

4.3.3 Manually labelled contours

Some features are not picked up by the automatically calculated contours and need to be added manually to ensure they can be sketched. The models in the training set share a common vertex structure so any labelled vertex indices on the standard face model are directly transferable to every training model. The models used in the training have no eyebrows as a part of the mesh surface, and instead include them as a part of the texture map. Manually labelling the eyebrows is therefore the only way to allow the user to sketch them, and can be easily defined by sketching the expected shape for the manual contours.

Figure 4.11 shows the sketched eyebrows (left) which will be used to locate the vertices associated with the manually defined features, and the individual sketched points on top of the vertex structure. The point density of the strokes is considerably higher than the vertices, particularly in the eyebrow region.

The next step is to map the sketched stroke points (black dots) to the nearest vertex (red cross), where the green lines indicate which vertex each point maps to (see Figure 4.12). Each vertex is mapped from more than one stroke point because of the stroke's high density compared to the vertices.

Duplicate labels are removed where vertices map to more than one stroke point (see Figure 4.13, left), as well as any excessive labelled vertices which are misleading, unnecessary, or disruptive. This is not a big issue here where the vertices in the eyebrow region are sparse, although one redundant vertex is removed

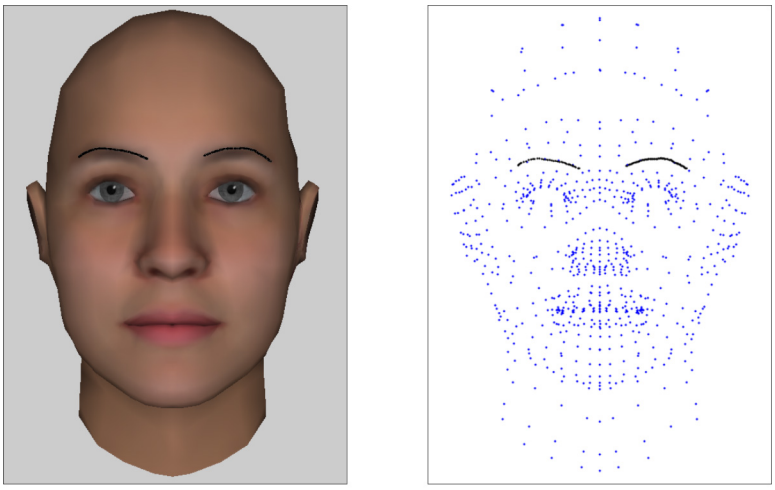


Figure 4.11: Creating manual contours for eyebrows using sketched strokes

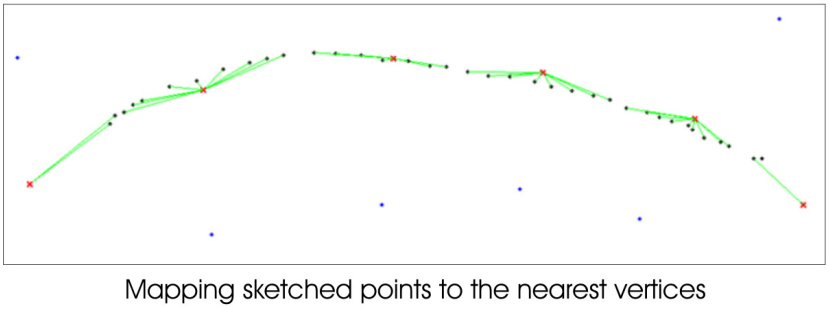


Figure 4.12: Mapping sketched contour strokes to nearby vertices

(green circle). The result is five labelled vertices for each eyebrow (right). The process of manually labelling new features is quick and straightforward, where the main issue is detecting and removing the unwanted labellings. Performing the same process in an area with a more dense vertex structure, such as the high dimensional models, can produce a bigger number of unwanted labelled vertices.

Another example of features that only exist as a part of the texture are the nostrils. The vertex density is also very low in that region making it tricky to label descriptive vertices. Only three vertices per nostril can be used to visualise their shape.

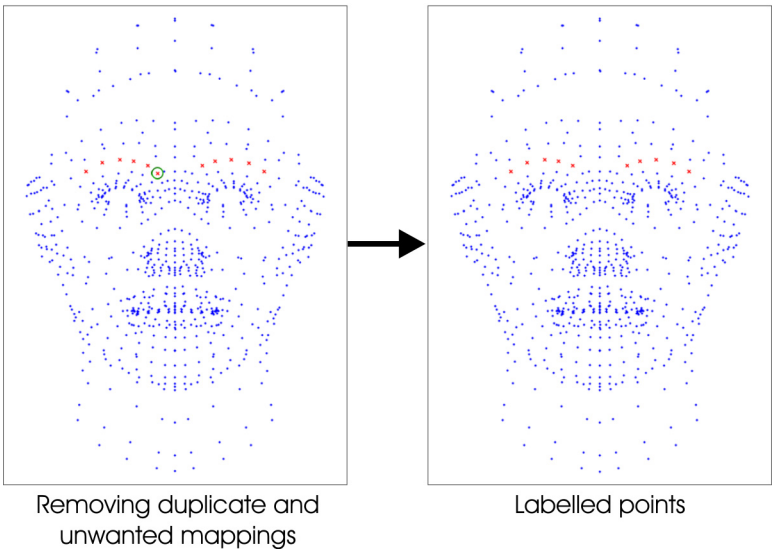


Figure 4.13: Cleaning up manual contours

The lips are not easily detectable from the front view, and they are rarely detected from any viewpoint in the low resolution models. For this reason the lip contours are sketched manually and the sketched points are mapped to the nearest vertices. The mapped vertex sequence is kept clean by removing any redundant vertices that could contribute to ambiguous or irrelevant contours. A total of 19 points are used for the upper and lower lips.

The eye sockets are detected by recognising that they form a connected edge where an edge made by two connecting vertices has only one adjacent triangle. Other edges such as the bottom of the neck are detected, but to avoid repeatedly culling unwanted edges when sketching, the eye socket edges are collected offline and included with the set of manually labelled contours.

Some contour points can cause problems when mapping to sketched strokes. This is most noticeable in the eye and lip regions where some vertices are occluded but are still detected as contours. This could be solved by excluding any occluded contours but it would add to the complexity of the online detection process.

Instead a quick yet acceptable solution for a low resolution model is to record the indices for these vertices and ignore them when generating the list of contours.

4.3.4 Caching curvature files

Figures 4.14 and 4.15 show the results of combining all the different sources together to form a complete set of contours for a single low resolution face.

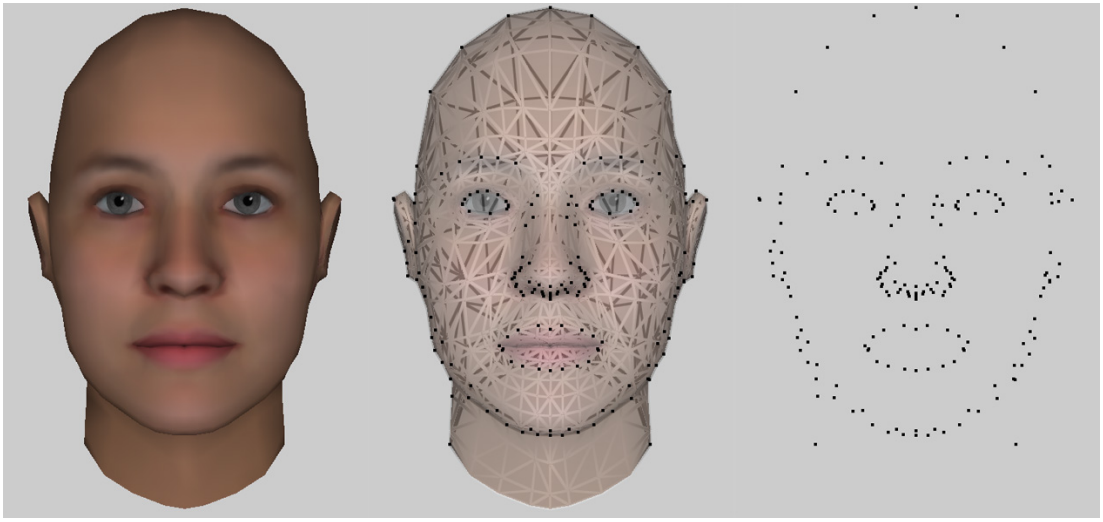


Figure 4.14: The average face model seen from the front viewpoint (left). The underlying wireframe visualises the vertex density and relevant contours (middle). The complete set of contours (right).

Let $\mathbf{M} = \{m_1, .., m_k\}$ be a collection of face models, where the curvature information and normals needed to calculate the contours for every $m \, \epsilon \, \mathbf{M}$ are calculated and stored offline. This enables calculating the complete set of contours in near real time in the sketching interface. This is discussed further in the next section.

4.4 Preparing contour candidates

Contours act as a bridge between stroke points and vertices on the face model as each contour represents a sketchable feature, and is associated with a particular vertex index.

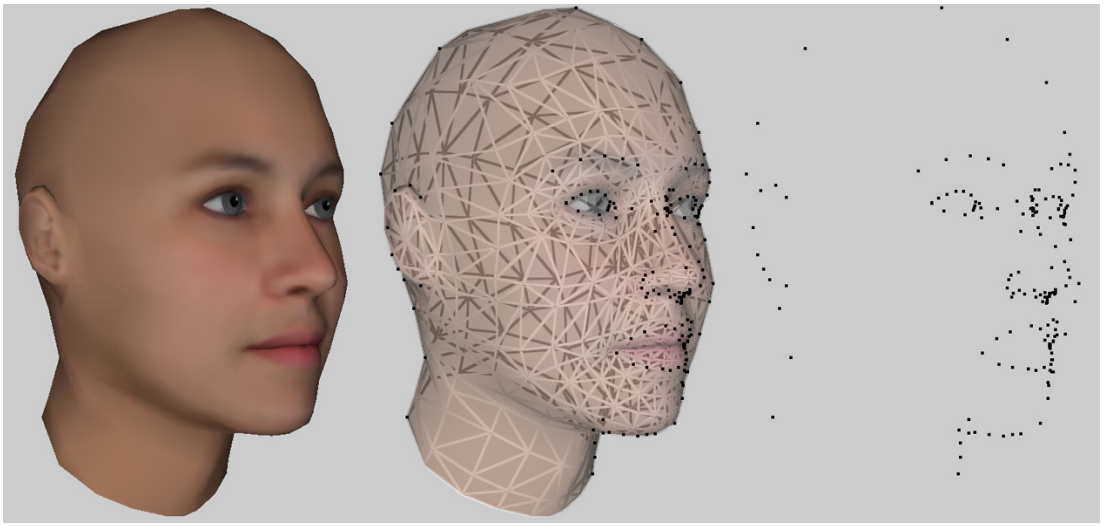


Figure 4.15: The average face model seen from an angle (left). The underlying wireframe visualises the vertex density and relevant contours (middle). The complete set of contours (right).

In order to map the stroke points to the appropriate contours, the contours are generated for every $m \, \epsilon \, \mathbf{M}$ from the current viewpoint using the curvature information and normals stored offline, and culled if they lie outside the stroke's bounding box. This forms a contour cloud which covers the range of possible ways for a stroke to be mapped, and in turn determines the sketchability boundaries (strokes outside the cloud are not registered). This cloud is denoted as contour candidates.

Figures 4.16 and 4.17 show the complete set of contour candidates for $\mathbf{M}$ from two different viewpoints, where the candidates are shown in green (middle). A number of candidates can occupy the same pixel space depending on the screen resolution and camera angle. The illustration on the right in the figures visualises this density where it ranges from blue (lowest) to red (highest). The mapping process is executed from the perspective of the stroke points where each point collects candidates lying nearby, which are then subject to further processing detailed in later sections.

Each $m \, \epsilon \, \mathbf{M}$ contains a set of t vertices $\mathbf{V} = \{\mathbf{v_1}, .., \mathbf{v_t}\} \, \epsilon \, \mathbb{R}^3$. A single stroke

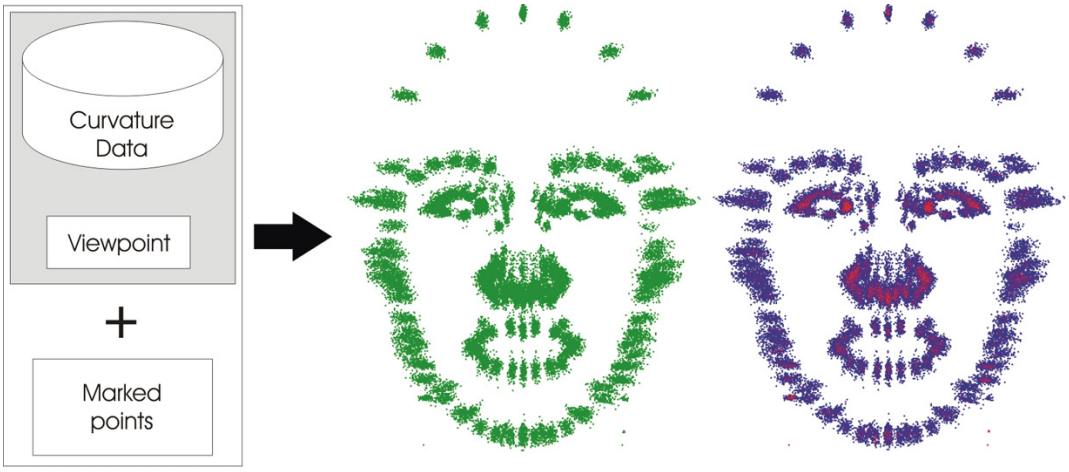


Figure 4.16: The contours are generated for every known model and combined to create a contour cloud which defines the range of sketchable features from the front viewpoint. The density is visualised on the right where it ranges from high density (red) to low (blue).

is made up of n points, $\mathbf{S} = \{\mathbf{p_1}, .., \mathbf{p_n}\} \epsilon \mathbb{R}^3$. The sketch detection often processes more than one stroke, but references to stroke numbers are omitted in the following equations for simplicity.

Each stroke point $s \epsilon \mathbf{S}$ is assigned m_i contour candidates which are chosen based on whether they lie within a variable threshold euclidean screen distance and a fixed world distance from s.

The assigned candidates form a collection $\mathbf{C_j} = (x_j, y_j, z_j, \eta_j)$, $j = 0..m_i$, where η_j specifies the vertex index the candidate maps to.

4.5 Reconstructing faces from extracted contours

It was argued in Section 4.1 that contours, as a product of NPR, depict a sketch-like representation of a 3D mesh. The inverse process of mapping strokes to contours is used to produce accurate 3D facial features through sketching. The contours are based on the mesh vertices and so can be seen as a low-dimensional version of the original mesh. It is important to verify that given an arbitrary

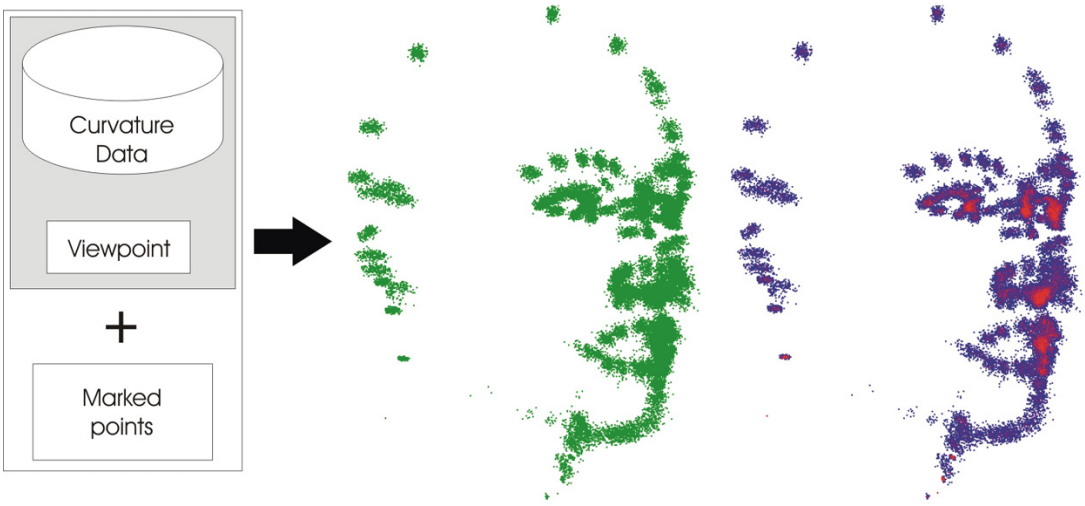


Figure 4.17: The contours are generated for every known model and combined to create a contour cloud which defines the range of sketchable features from the an arbitrary viewpoint. The density is visualised on the right where it ranges from high density (red) to low (blue).

sample mesh, that its corresponding contours can be used to faithfully reconstruct the original mesh with all features intact. By treating the contours as a sketched strokes will at the same time simulate precise sketching.

4.5.1 Creating existing faces

4.5.1.1 Reconstruction assuming a known vertex structure

The first step is to test the statistical model by generating contours using an arbitrary mesh from the statistical training set, or a mesh that shares the same polygon structure. A simulated stroke is created by assigning the generated contours as a set of discrete stroke points. Because the vertex structure is known, it is already known which vertices on the mesh the contours map to so the coordinates are easily extracted and assigned to the corresponding vertices.

The mesh now has some known vertices and the remaining vertices can be found using the method described in Section 4.2.1.6. Figure 4.18 displays three examples of this method where the row on the left shows the original source

meshes, the middle row shows the reconstructed target meshes using the contours, and the row on the right shows the difference between the source and target where the vertex distances are colour coded. The distance ranges from blue (minimum distance) to red (maximum distance).

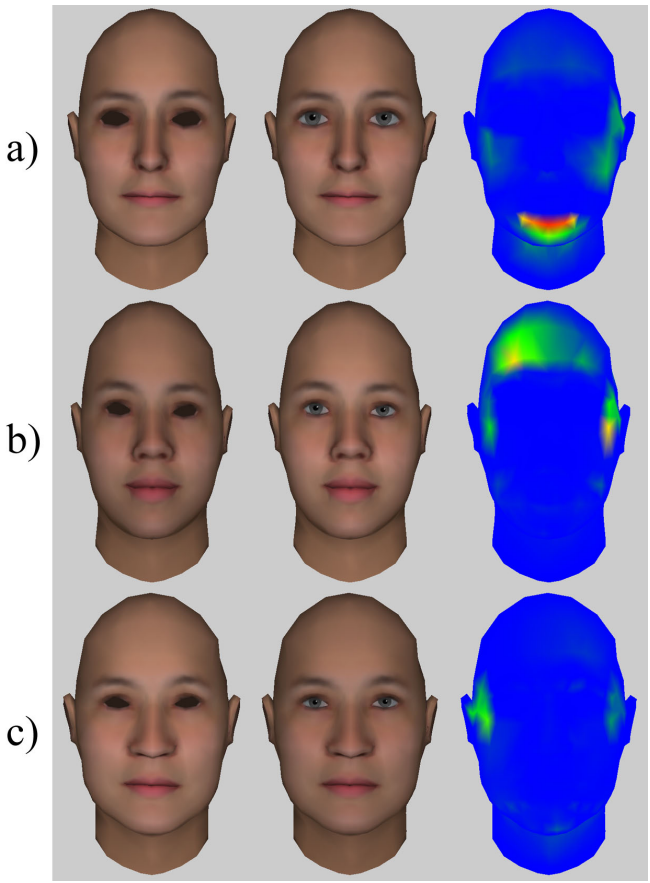


Figure 4.18: Reconstruction using a known vertex structure. The sample meshes contain 863 vertices. a) Sample F01, 180 contour points used b) sample F22, 159 contour points used c) sample F55, 160 contour points used

The results indicate that it is reasonable to assume that the partial input based on contour data can be used to accurately reconstruct a complete mesh with the intended features intact. The system is less confident in areas where the contours are sparse, as more than one shape is plausible given the surrounding contours based on the faces in the training set. This test does not indicate how well the system will fair when dealing with sketched strokes as unlike the contours,

the points making up the strokes have no prior knowledge of which vertex they should map to.

4.5.1.2 Assuming an unknown vertex structure

As before, the aim is to reconstruct an arbitrary face model using its contours. However, when creating the simulated stroke, the contours will now ignore the vertex index they belong to, and instead map the generated stroke points to the most appropriate index based on specific criteria. This is done to give a better simulation of using sketched strokes, as they do not know beforehand which vertex indices in the polygon structure their points should map to. Additionally, this allows using a model whose vertex topology does not necessarily match that of the training set.

The first step creates all the contour candidates as explained in Section 4.4. The second step iterates through stroke points created from the contours, and maps them to the closest candidate in screen space by calculating the euclidean distance. The mapping involves assigning the candidate 3D coordinate to the mesh vertex corresponding to the candidate vertex index η_j. The last stage is as before to reconstruct a full mesh given the partial input provided by the mapped candidates. Figure 4.19 shows the reconstruction results for the same sample meshes as in Figure 4.18 for comparison. As expected, the results are not as accurate as the earlier test where the index values were known, and the ears can cause unexpected problems. Importantly though, it shows that the system is capable of producing a visually accurate likeness of an sample mesh without knowing in advance what each contour point represents in relation to the sample.

4.5.1.3 Using 2D contour points

Since the mapping process is working primarily in screen space, the depth can be discarded entirely with very little effect on the reconstruction error. Figure 4.20 shows where the contours for the sample mesh F01 have been projected onto a 2D plane and used to reconstruct a complete mesh without compromising the

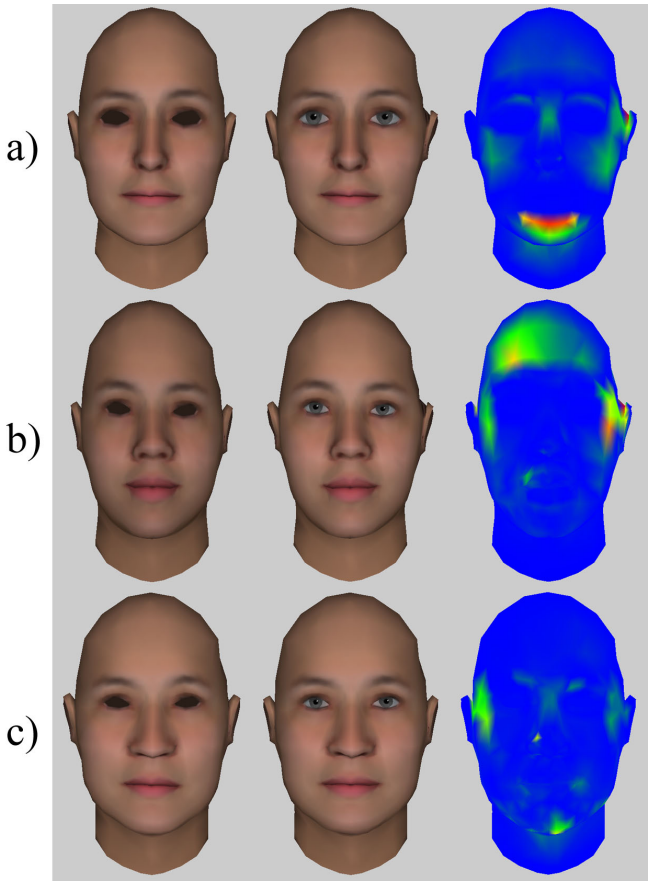


Figure 4.19: Reconstruction using an unknown vertex structure. The sample meshes contain 863 vertices. a) Sample F01, 180 contour points mapped to candidates b) sample F22, 159 contour points mapped c) sample F55, 160 contour points mapped

reconstruction accuracy. This feature can be used to simulate a natural sketch-on-paper experience where the user sketches the initial 3D shape on a blank canvas, and later tweaks it in 3D by sketching from different angles as needed. This is touched on briefly in the next section but further improvements and testing are left as possible future work.

Figure 4.20: Reconstruction using an unknown 2-dimensional vertex structure. a) Contour points for sample have been projected onto a 2D plane b) Original sample mesh c) Reconstructed mesh using 176 mapped contour points d) Colour coded distance to the original mesh

4.5.2 Creating novel faces

It has been shown here that a face in the training set can be reproduced using the methods described above, but using an existing face is not a realistic scenario as an arbitrary sample mesh will not have a direct correspondence to a face in the training set.

4.5.2.1 Measuring the generative properties

The smaller the training set of observed faces, the more it is up to chance whether a novel face fits well enough within the range of prior knowledge in order to create a decent approximation. It is of interest to see how many faces a generative model needs to observe in order to recreate a number of novel faces accurately.

This is touched on here by recreating three target face models from their contours using several generative models where each has been trained using a different number of observed faces. To clarify, if the training sets are denoted as $G_i, i = 1..n$, where G_1 is the smallest set and the G_n is the largest, then $G_{i-1} \subseteq G_i$. None of the target models are part of any of the training sets. Figure 4.21 shows the reconstructions (right) of each target (left) where each column is generated by a single generative model. The number at the bottom of each column represents the number of training samples used. Figure 4.22 displays

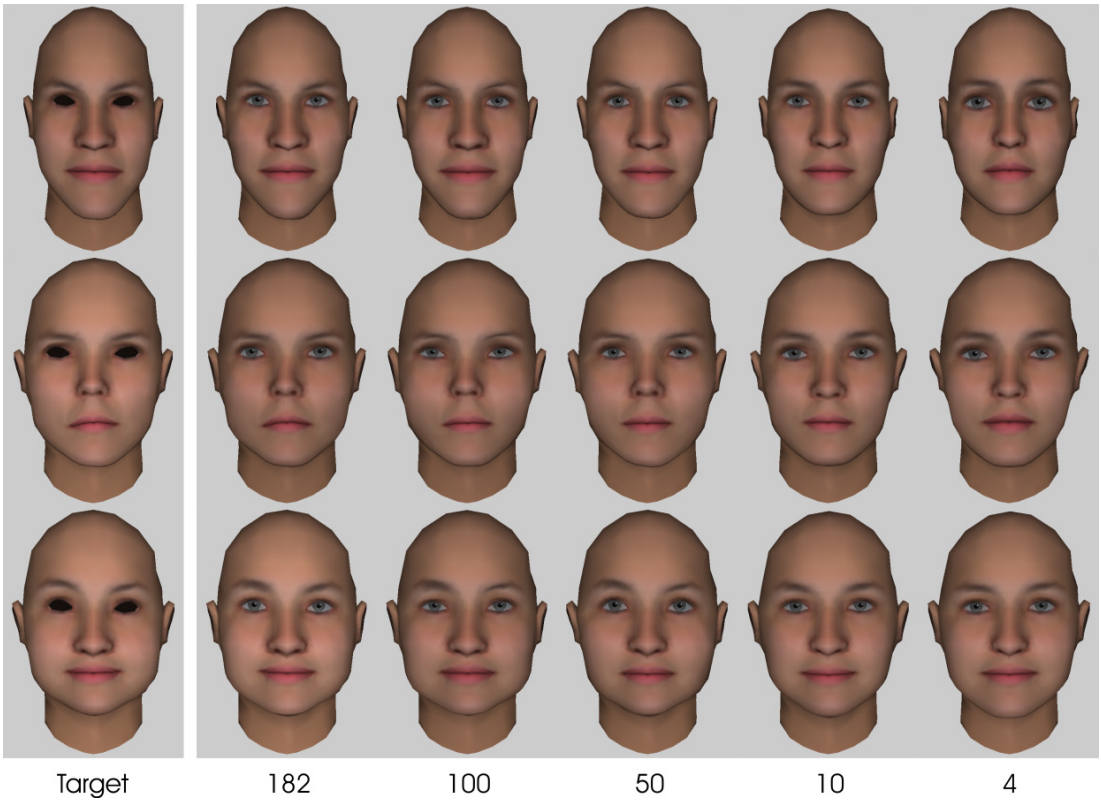


Figure 4.21: Reconstructing models from contours using different number of training samples. Each column visualises 3 face models at different stages where the target models are located on the left. The numbers below the columns specify the number of training samples used to train the generative models responsible for reconstructing the face models in the corresponding columns.

the euclidean distance (error) from the target for each reconstruction. Although a low euclidean error does not guarantee a subjective likeness, it provides an indication of whether there has been an overall improvement. As expected, the reconstruction based on only 4 faces is coincidental which results in a greater error variance. They quickly start to converge from only 10 training samples, and already produce a moderately accurate likeness based on 50 samples which is reflected in Figure 4.21. From then on a stable linear improvement is seen when observing more faces which enhances the details to give a more convincing reconstruction.

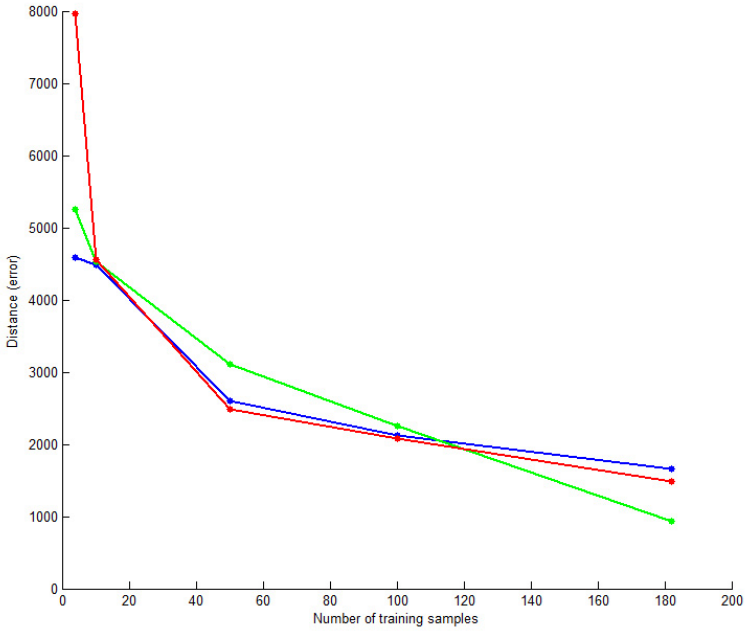


Figure 4.22: The reconstruction errors based on generative models trained on different number of samples. The error is the total size of the translations needed to move every vertex to the correct target location in the XYZ world space.

4.5.3 Depth ambiguity

The contours are made to simulate sketched data by discarding their depth values and performing measurements in screen space. This causes the depth ambiguity issue that plagues sketching interfaces to become apparent. Sketched strokes are 2-dimensional so when sketching from a particular viewpoint, the depth has to be inferred through maximum likelihood. However, what the generative model classifies as the most likely outcome is not necessarily what the user is after. The same applies when contours extracted from complete samples are used unless the sample is in the training set for the generative model where its values will arguably be considered the most likely outcome. Figure 4.23 demonstrates this by choosing a model that has been left out of the training set, and reconstructing it from different viewpoints. First the mesh is reconstructed using the front view where 156 points are mapped to vertices. The reconstruction looks convincing when comparing it from the front view, but some differences are noticeable when the

comparing the profile to the original. This is again applicable in the second test where a model is reconstructed using the profile view. The reconstructed profile is more accurate than the one generated using the front view, but the system is unaware of the width of the facial features. This causes it to underestimate the width of the cheeks and nose as expected because the majority of the faces in the training set do not possess wide cheekbones.

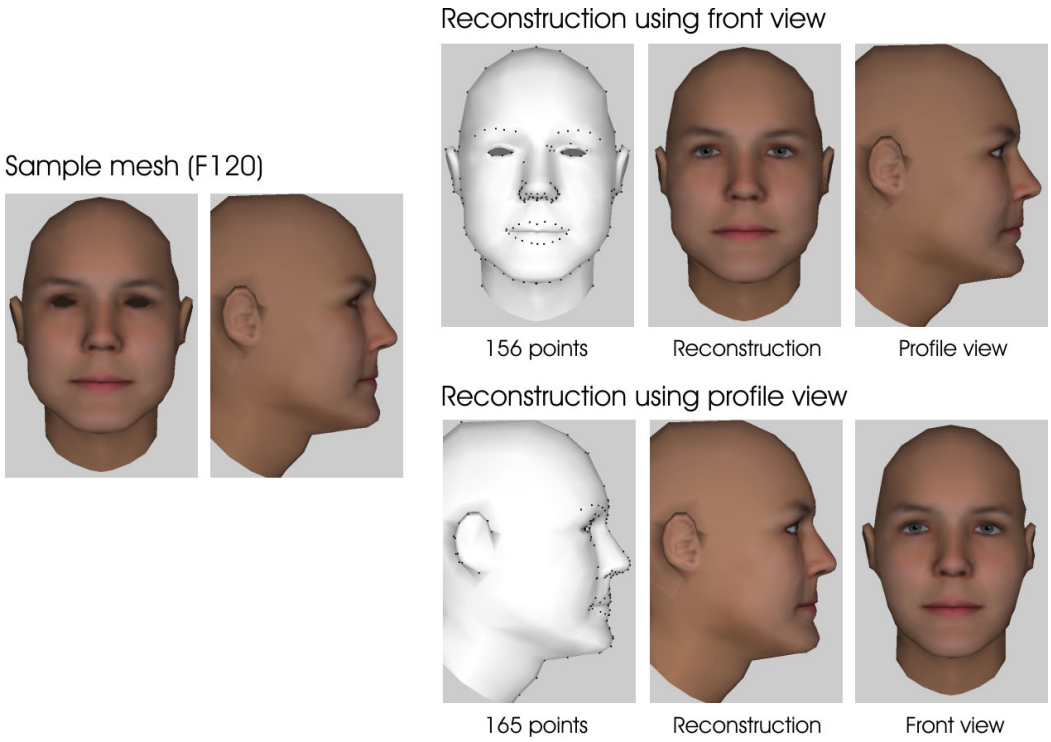


Figure 4.23: The missing depth is inferred using the most likely values based on the training set, but may not reflect the user's expectations

4.5.4 Subjective test

So far the reconstruction error has been measured objectively, and shown visually by colour coding the model vertices. However, a low overall reconstruction error is not a guarantee for an accurate likeness from a human perspective [SJ99]. This can be measured subjectively with human subjects by asking them to identify a

range of target faces from a list of possible faces, where one of the choices is the reconstructed model, and the other choices are faces that resemble the target face. If the users consistently pick the reconstructed face out of the pool of choices, it will indicate that the reconstructions are recognisable and therefore produce an adequate likeness of the original target faces.

4.5.4.1 Test setup

An online test was devised where users completed a series of recognition tasks. They were asked to identify a series of 10 target face models. The eyes were omitted, and all the models had the same texture to focus primarily on the differences in shape. Figure 4.24 presents 9 faces, where the target face is in the centre surrounded by 8 randomly ordered candidate faces. One of the candidates

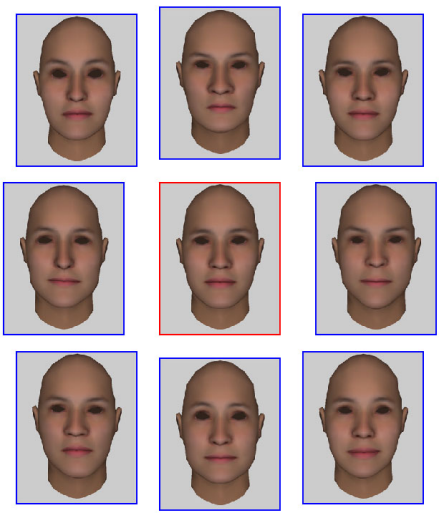


Figure 4.24: The user looks at the target face (centre), and picks the surrounding face he thinks most resembles the target.

was a reconstructed face using the contours generated from the target model, while the other seven were faces from the training set that most resemble the target face. The closest matches to the target face are determined using the following process:

1. Fit a Dual PPCA model on the contour data acquired from the set of n meshes in the training set.

2. Calculate the latent variables $x_i, i = 1..n$, for each sample t_i in the set using $\mathbf{x_i} = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{t_i} - \mu)$ (Equation 4.20).

3. Calculate the latent variables x_t for the target face.

4. Calculate the errors $\varepsilon_i = \|x_t - x_i\|$.

5. The 7 lowest errors are chosen as the candidate faces from the training set.

The contour coordinates are used instead of the vertex coordinates because the focus is on the main facial features from a particular angle. If all the vertices were used, the large number vertices irrelevant to those features, could sway the error value to be smaller than a face that has very similar features, but differs in redundant areas.

4.5.4.2 Test results

122 subjects took part in the test ($n = 122$) where the set of scores are referred to as X and $x_i \, \epsilon \, X, i = 1..n$. Some noise is expected when analysing the results due to the relatively uncontrolled nature of an online test. Users are often not as engaged and prepared when compared to a controlled environment, where the tester can interact with the subject. Some users scored 1/10 (10%) where it is likely they either misunderstood the test, or did not give their full attention since the chance of selecting a single reconstructed face randomly is 1/8 (12.5%).

The mean score was $\mu_X = 0.6721$ with a standard deviation $\sigma = 0.2117$ found using

$$\sigma = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu_X)^2 \right)^{\frac{1}{2}}. \tag{4.28}$$

Figure 4.25 shows how many times each reconstructed model was identified as the target model (a), and how many times another candidate was chosen instead of the reconstructed model (b). The reconstruction error for each model is measured as the sum of differences between the vertices of the reconstructed model and the target model (c). It is apparent that this objective euclidean distance error is not

a good indicator of how well the model scored in the subjective recognition test, although two factors affect the credibility of direct comparison.

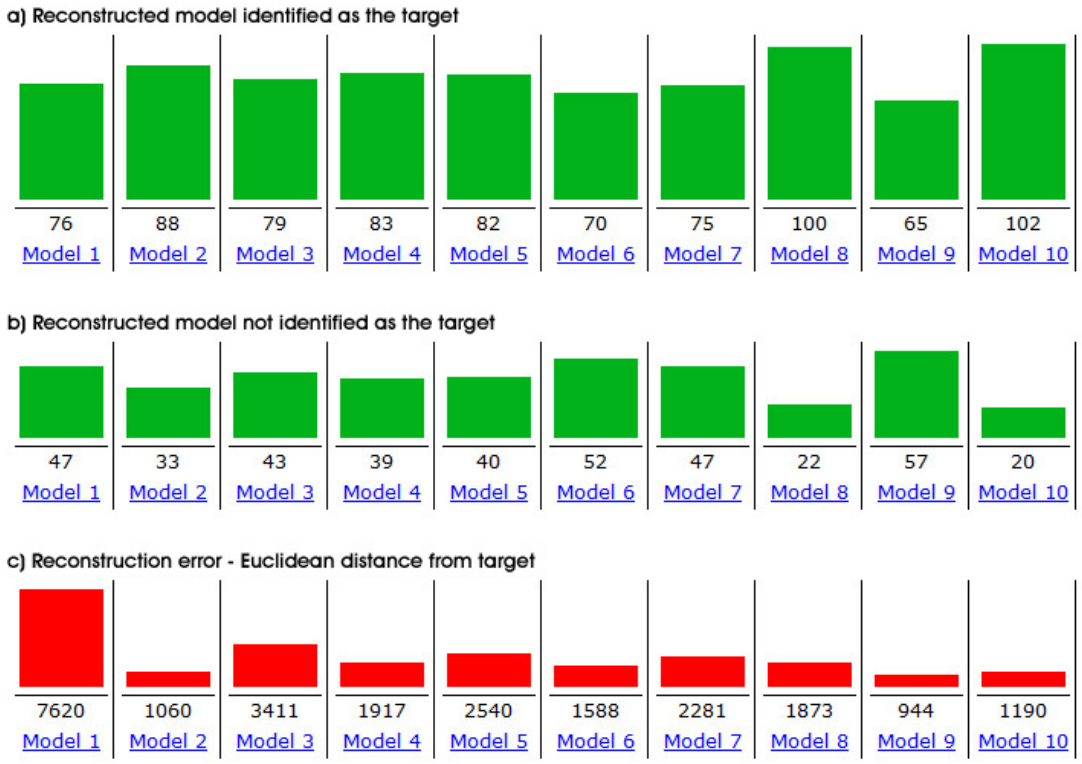


Figure 4.25: Model statistics for an online user test measuring the subjective likeness of the reconstruction process

The first factor influences the reconstruction error. As discussed in Section 4.5.3, the face is reconstructed using contours from a specific angle. This means that even though the reconstructed model looks similar from that particular viewpoint, the depth is ambiguous and is not necessarily accurate as it depends on what the statistical model finds likely given its prior knowledge (see Figure 4.23). This can contribute to the an overall higher reconstruction error yet retaining a similar overall appearance from the fixed viewpoint.

The second factor concerns the recognition score. The ability to correctly classify the reconstructed face from the other seven candidates is affected by how similar the faces in the training set are to the target face, which is coincidental.

Evidence of this can be found by observing that target model 9 has the lowest reconstruction error, yet has the highest number of failed recognitions. Figures 4.26 and 4.27 decompose the results for target models 4 and 8, respectively. The reconstructed models are shown in the top row along with the number times they were classified as the target model (number of hits). The remaining rows show the candidate models from the training set that were also classified as the target model, ordered based on the number of hits they received.

The following list compares the training model candidates displayed in Figure 4.26 with the target model in order to speculate why the users picked them:

- a) The nose and lips are quite similar to the target, giving the central part of the face an overall likeness despite the target being considerably wider with smaller eyes. The reconstructed model is arguably a better likeness apart from the ears which is potentially confusing the users, and swaying them to pick this model.
- b) Most features on this model differ from the target. The face has larger eyes, is rounder and shorter, and the nose is narrower. The shape of the lips is similar but relatively smaller. It is difficult to pick a feature which convinced the users to pick this face.
- c) This model has similar eyes, chin, and lower cheeks as the target model, but the lips are quite different.
- d) Here the eyes are of similar size and shape, and the lips are of similar thickness albeit not as wide as on the target model. The nose is completely different in almost every aspect.
- e) This model has almost identical lips to the target model which is a distinguishing feature, but it is not as elongated and is clearly a female. Speculatively, the lips must have been the deciding factor for the single case.

The most distinguishing features shared between these models and the target are the lips and nose which seems to be enough to prompt the users to pick

them over the reconstructed face. It is possible the poorly generated ears on the reconstructed model convinces some users that it is not the best likeness.

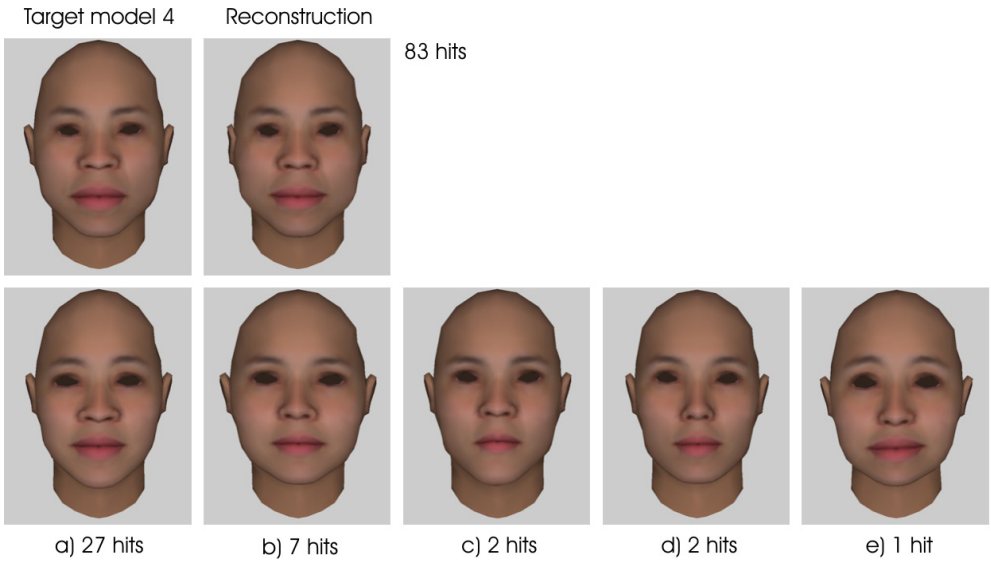


Figure 4.26: Identifying similarities between target model 4 and the candidate models classified as the target during the online test

Likewise, the following list compares and discusses the training model candidates displayed in Figure 4.27:

- a) Similar overall elongated shape, width and height. Lips of similar length but slightly thinner. The shape of the eyebrows resemble the target, the nose width is similar but the nose tip is too high.
- b) The chin curves are very similar although the overall face is slightly less elongated. Nose and lips pair up quite well, both in terms of shape and the distance between them.
- c) Elongated face and similar eyes.
- d)-g) All these faces have a similar slim elongated facial structure, but none of them have any particular features that compare well with the target apart from similar eyebrows in the case of e and f to some extent although its jaw structure is considerably wider.

None of the faces in the set of training candidates stands out as being an accurate likeness of the target, but they do share a common characteristic, the slim elongated overall shape. Face b has the most convincing central region with a similar nose, lips, and chin, while faces a and e possess a better eye region with lowered eyebrows.

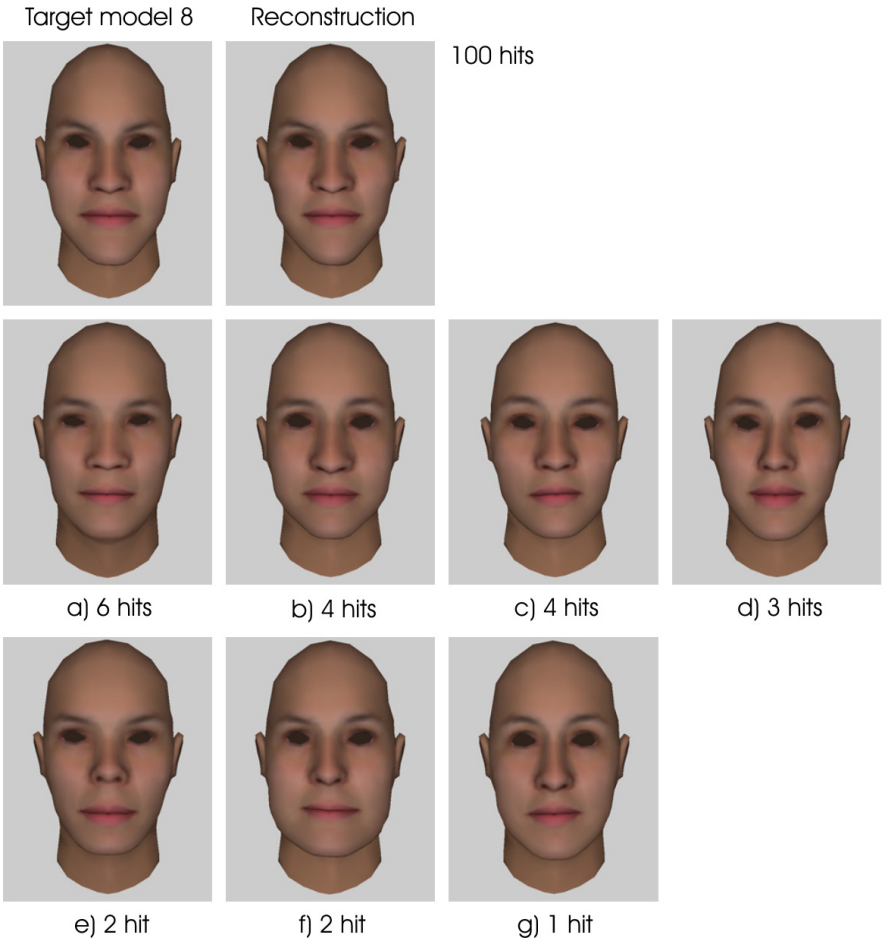


Figure 4.27: Identifying similarities between target model 8 and the candidate models classified as the target during the online test

Both target examples introduced in these figures have one distinguishing factor which seems to continuously influence the users' choices. This trait is exploited when creating caricatures of individuals which has been shown to be as recognisable or better than the original counterpart [MK92; SBOR07].

The second factor affecting the recognition score is observed by comparing the results from the two target models in Figures 4.26 and 4.27. For model 4, model a) that is repeatedly chosen (27 hits) over the reconstructed one. That model arguably bears resemblance to the original face. In contrast, no candidate model is picked noticeably more often than the others for model 8 which is arguably because none of them are overwhelmingly similar to the target model, giving the corresponding reconstructed model the second highest hit count of the test.

4.5.5 Limitations

Mapping sketched points to the contour candidates is only based on the minimum euclidean distance, not taking into account the context, continuity, and structural integrity of the vertex structure affected by the mapped contours. This is tackled in the next section when mapping manually sketched strokes, but is left here as possible future work.

Another limiting factor is the lack of training data. If the generative model has never encountered certain features it will struggle to reconstruct them. This also affects the mapping process since it generates the contour candidates from the training data which the sketched strokes map to. It is worth considering whether the generative model or the mapping process is more affected by the lack of observed data. The probabilistic properties of the generative model suggests it can cope better than the heuristic mapping process and this is found to be the case. This is demonstrated in Figure 4.28 where a target model (left) is reconstructed from its contours. Here, the curvature data for this model is added to the offline data used in the contour mapping process, while leaving the generative model unchanged. A comparison of the reconstructed models (right) yields an improvement when using more curvature data where it is particularly noticeable in the nose region. This indicates the generative model is able to reconstruct this target mesh faithfully without having observed it, but the mapping process fails to supply it with accurate data based on the nearest candidate criteria.

It raises a question whether it is possible to fit a generative model to the curvature/contour data to create a probabilistic mapping process. The contours

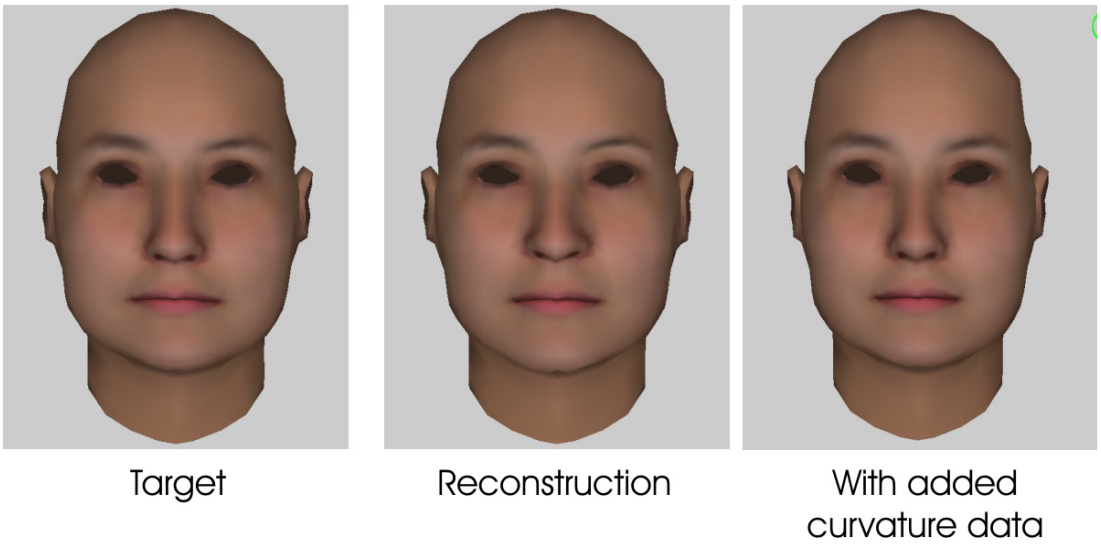


Figure 4.28: Adding curvature data without changing the generative model can produce better reconstructions

are view-dependent and therefore the set of candidates changes based on the viewpoint, both in terms of quantity, and which vertices they correspond to. The solution to that problem lies outside the scope of this thesis.

4.6 Reconstructing faces from strokes: A heuristic approach

The previous section showed how automatically generated 3D contour points, simulating sketched strokes, can be used to reconstruct a novel 3D face model from any viewpoint. The contour points represent an accurate, and non-redundant description of the facial features for a particular face model which corresponds to the existing training data. Sketched strokes however are made up of an arbitrary number of points where it has to be determined which combination of points, sampled from the strokes, is the best to describe the intended facial features. The sketched points are typically more dense than the vertex structure which means that several points compete to represent a single vertex, whereas before the contours had a direct correspondence to the vertex structure. Picking the

right set of points ensures a good fit while mapping an inconsistent set of points can give unsatisfactory and unexpected results.

Sketching takes place in the 2D screen space where the stroke points can be projected onto the surface of the current face model or axis planes to get a depth estimate. The intended feature may not have the same depth as the surface the stroke is projected on so this does not give an accurate estimate. However, the estimate is still useful by culling unlikely candidates as a sketched feature tends to lie close to the original feature on the underlying mesh.

These issues make it more difficult to interpret sketched strokes. They can be summarised into the following questions:

1. What is the most suitable 3D vertex for a given 2D stroke point?

2. Given that the points have been assigned to vertices through contours, which set of points best conveys the intentions for a particular facial feature embedded in the sketched stroke.

These questions are coupled where the second one relies on the first one to find an appropriate candidate for any sketched point, but in order to do so it has to take into account the context provided by the other points.

A stroke is a sequence of linked points portraying a facial feature. Therefore a contextual meaning is gained from taking into account its sequential properties as opposed to interpreting each point independently.

This section tackled this heuristically by iterating through each point on a stroke, and assigning a grade to every potential contour candidate based on how well it fits the point being examined, as well as considering how the previous point was assigned. An overview of the whole process for a single stroke is given in Figure 4.29. Section 4.4 describes stages 2 and 3, Sections 4.6.1 and 4.6.2 go through stage 4 which finds the best candidates based on the grading scheme and assigning them to vertices, and stage 5 is covered in Section 4.6.4. The optional step of correlating unsketched feature is discussed in Section 4.6.4.

The notion of utilising a sequential pattern when assigning a stroke to a vertex structure is naturally described using a Hidden Markov Model (HMM) where the

grades are replaced with probability distribution functions. This is covered in Section 4.8.

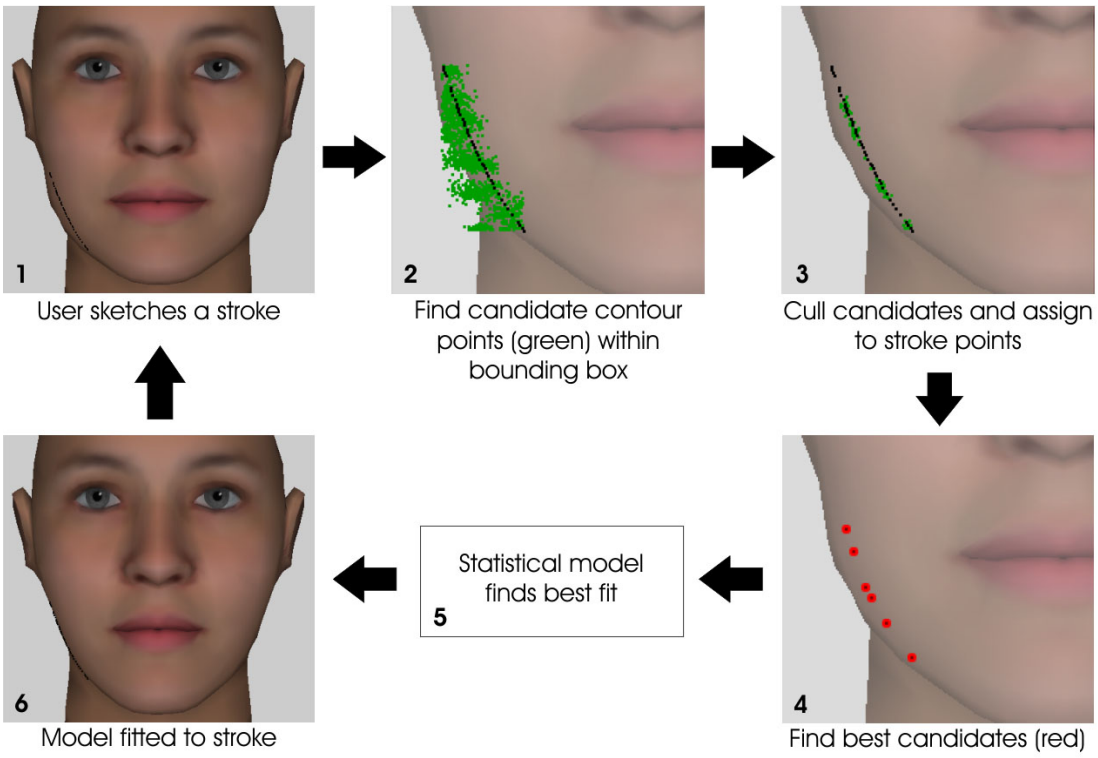


Figure 4.29: Overview of the mapping process

4.6.1 Calculating grades

The grading process goes through every stroke point that has assigned candidates and calculates the grade for each candidate j using

$$\varepsilon_j = \psi(\lambda(r_x\varphi_x + r_y\varphi_y)) + \alpha s_1 + \beta s_2, \tag{4.29}$$

whose terms are explained in Sections 4.6.1.1 and 4.6.1.2. It consists of two main factors. The first factor, $\psi(\lambda(r_x\varphi_x + r_y\varphi_y))$ is sensitive to the stroke's endpoints which generally define the boundaries of the sketched feature, while the second one $\alpha s_1 + \beta s_2$, ensures the chosen candidates form a structure that is faithful to

the stroke's directional changes.

Two functions are used throughout this process:

- The function $f : \mathbb{R}^3 \rightarrow \mathbb{Z}_+^2$ maps a 3-dimensional world coordinate to its projected 2-dimensional screen coordinate. For example, $f(v)$, $v \, \epsilon \, V$ projects a 3D vertex v to its corresponding 2D coordinate.
- The function $g : \mathbb{Z}_+ \rightarrow \mathbb{R}^3$ looks up a vertex coordinate with a given index from a predetermined mesh $\mathbf{M_k}$.

4.6.1.1 Feature boundaries

It is very important to map the first and last point on the stroke to the most plausible vertex indices, as they carry important descriptive information about the stroke and its intentions. The first point will also set the context to define how the remaining points will be determined. Choosing the wrong one may give results that look nothing like what the user had intended. Figure 4.30 visualises this problem where it shows a stroke drawn top-down (a) intended to specify the right wing of the nose. The first step (b) finds candidates that lie within a threshold distance from the stroke points. Nose shapes vary among the training samples and differ from that of the current face model on the screen, e.g. lie higher or lower in screen space. This is demonstrated in c) where candidates that lie close to the first point have different contexts in the sample models. The candidate in the top row describes the top of the nasal wing, while the candidate in the last row refers to the bottom of the wing. The range of potential candidates can be examined by mapping them to the template model (d) and comparing their absolute values. Strokes are generally intended to define the start and end of a feature, so in this case it is assumed that the top point is contextually most relevant.

The first part of equation 4.29 focuses on this problem and is activated using the switch ψ, where

$$\psi = \begin{cases} 1 & \text{if } i = min(i) \text{ or } i = max(i) \\ 0 & \text{otherwise.} \end{cases}$$

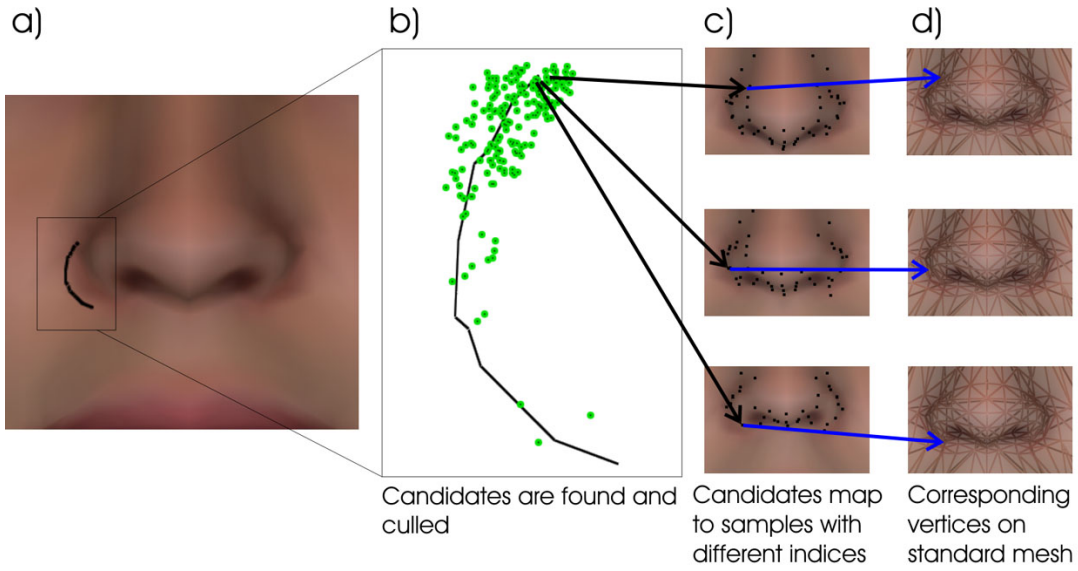


Figure 4.30: Setting the context for the first point in a stroke from a set of contour candidates

Every candidate for the first and last stroke point is mapped to the template model followed by calculating their mapped screen coordinates. This collection of screen coordinates forms a bounding box which is used to determine the most suitable candidate. φ_x and φ_y define the distance from the edge of the x and y axes. The best candidate is in principle the one with the lowest error for $\varphi_x + \varphi_y$, where φ_x and φ_y define the proportional distance from the width and height edge respectively, of the bounding box, or

$$\varphi_x = \frac{f(g(c))_x - min(f(g(\mathbf{C}_i))_x)}{max(f(g(\mathbf{C}_i))_x)}, \tag{4.30}$$

$$\varphi_y = \frac{f(g(c))_y - min(f(g(\mathbf{C}_i))_y)}{max(f(g(\mathbf{C}_i))_y)}, \tag{4.31}$$

where $c \, \epsilon \, \mathbf{C_i}$.

According to equation 4.30, the proportional distance is always measured from the left edge of the bounding box, and similarly equation 4.31 measures from the

top edge. This is only applicable for the first point if the stroke is sketched top-down and left-to-right. To take stroke orientation into account, φ_x and φ_y are subject to the following criteria:

$$\varphi_x = \begin{cases} \varphi_x = 1 - \varphi_x & \text{if } i = max(i) \text{ and } \omega_x = 0 \\ \varphi_x = 1 - \varphi_x & \text{if } i = min(i) \text{ and } \omega_x = 1 \\ \varphi_x & \text{otherwise} \end{cases} \tag{4.32}$$

$$\varphi_y = \begin{cases} \varphi_y = 1 - \varphi_y & \text{if } i = max(i) \text{ and } \omega_y = 0 \\ \varphi_y = 1 - \varphi_y & \text{if } i = min(i) \text{ and } \omega_y = 1 \\ \varphi_y & \text{otherwise} \end{cases} \tag{4.33}$$

$\omega_x \epsilon \{0, 1\}$ and $\omega_y \epsilon \{0, 1\}$ determine the orientation of the stroke in a very simple and straightforward way. A vector between the first point ($i = 1$) and the middle point ($i = \lceil \frac{n}{2} \rceil$) on the stroke is calculated, and the vector direction is used to assign the parameters ω_x and ω_y according to the axis chart shown in Figure 4.31.

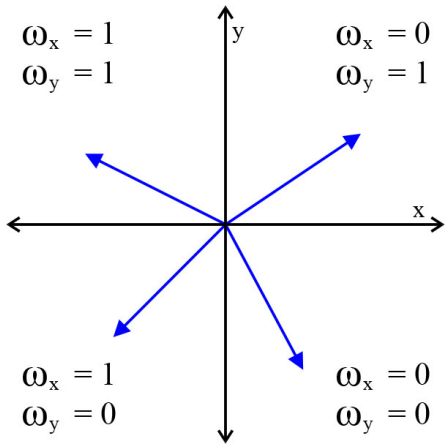


Figure 4.31: Stroke orientation parameters

Both axes (XY) govern how the first point is assigned, but they should carry different weights depending on the context as they can conflict otherwise resulting in inaccurate reconstructions. The ratio between the height and width of the stroke's bounding box is generally a good indicator of the relevant context. The stroke in Figure 4.30 is elongated along the Y-axis putting an emphasis on picking the best candidate along that axis. Therefore, to control how much effect each

screen axis has on the error, the ratios between the width (r_x) and height (r_y) of the stroke's bounding box are used to linearly blend φ_x and φ_y (see Figure 4.32).

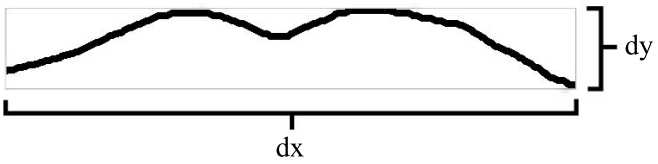


Figure 4.32: The stroke width to height ratio determines which axis has more influence when setting the stroke's context

The overall effect this has on the grade is scaled with the linear scaling factor λ. These assumptions have been found to improve how accurately reconstructed features fit the sketched stroke.

$$r_x = \frac{1}{dy/dx} \tag{4.34}$$

$$r_y = \frac{1}{dx/dy} \tag{4.35}$$

4.6.1.2 Mapping the structure to the stroke

The context of the stroke has been set by applying the methods described above to find the most suitable candidate for the first point in the stroke. The second part of equation 4.29 controls how the remaining stroke points are assigned to vertices based on the previously assigned vertex.

There are two aspects that affect this decision, the metrics s_1 and s_2 and are subject to the weighting coefficients α and β respectively, and $\alpha, \beta \, \epsilon \, \mathbb{R}, \; \alpha + \beta = 1$.

The premise for the metric s_1 is visualised in Figure 4.33 where its aim is to find the candidates that best match the angle and distance of the stroke points. An arbitrary stroke is displayed (a) with the set of culled candidates (green) found for each point on the stroke. The vector between the last assigned stroke point

p_l and the current one p_i is found (b). s_1 then measures the vector difference between the vector in b) and the vectors in c) which is go from the previously assigned candidate (red), to the set of candidates c_j (green), where $j = 1..m_i$. A secondary objective is achieved with this process where it helps to choose between stroke points as some of them tend to map to the same indices due to their high density compared to the vertex structure (see Section 4.6.2).

$$s_1 = \|(f(p_i) - f(p_l)) - (f(c_j) - f(c_l))\| \tag{4.36}$$

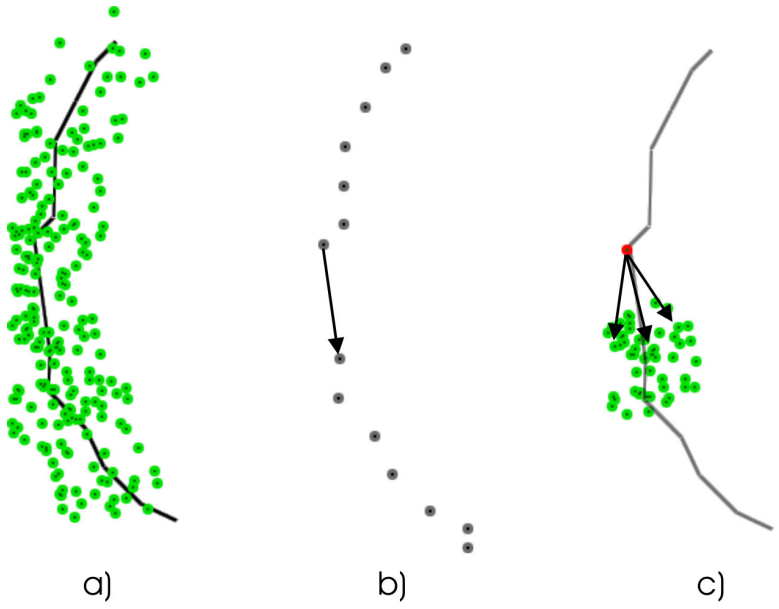


Figure 4.33: a) The contour candidates for the stroke The metric s_1 calculates the difference between the transition vectors in b) and c) (in screen space), where lower values give a higher grade

The second metric, s_2, helps to ensure the selection of a candidate that is faithful to the vertex topology based on the previously assigned candidate (see Figure 4.34). As with s_1, the first step is to find the vector between the last assigned stroke point p_l and the current one p_i (a). The second stage (b) maps the set of candidates c_j (green), $j = 1..m_i$, to the template mesh and finds the

vector between the mapped c_j and the previously assigned candidate (red). s_2 measures the difference between the vector in (a) and m_i vectors in (b).

$$s_2 = \|(f(p_i) - f(p_l)) - (f(g(c_j)) - f(g(c_l)))\|. \tag{4.37}$$

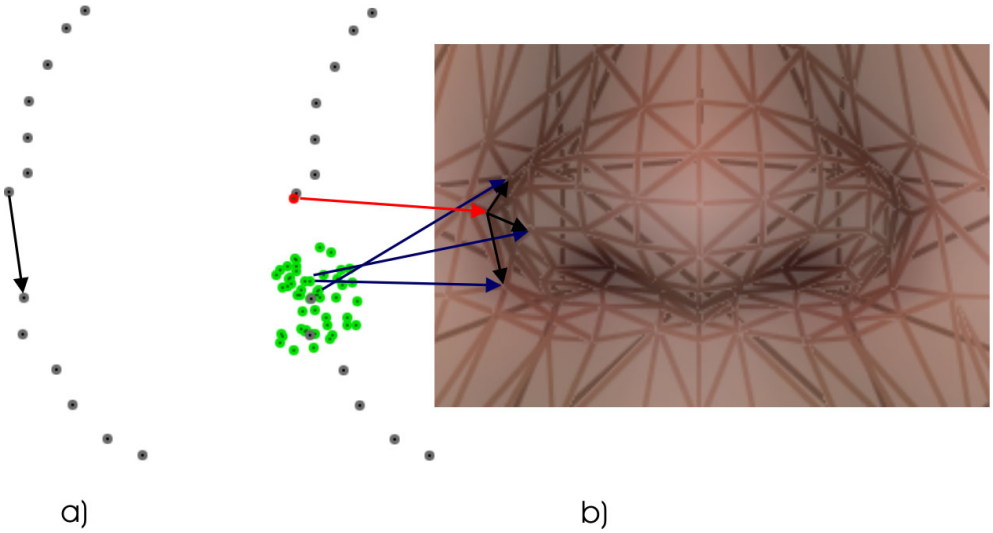


Figure 4.34: The metric s_2 calculates the difference between the transition vectors in a) and b) (in screen space), where lower values give a higher grade

4.6.2 Finding the best candidates based on grades

The previous section showed how every point in a sketched stroke is assigned with potential contour candidates, and the candidates were given grades based on the stroke's boundaries and relative position to previously assigned candidates. The candidate with the highest grade is chosen and mapped to vertex index η_j given that two conditions are met.

1. The candidate does not overwrite a previously assigned extreme point on the stroke (start, end, top, bottom, left, right).

2. If the vertex has been assigned before, the grade has to be higher than the existing grade, and satisfy the first condition.

The grading scheme along with these conditions solves two issues discussed at the beginning of Section 4.6. The candidates with the highest grade have the best chance of mapping to the most suitable sequence of vertices. The first condition ensures the reconstructed features preserve the stroke's boundaries, and the second condition chooses between points competing for the same vertex. Figure 4.35 shows two sets of sketched strokes from different viewpoints (left), and the reconstructed models (right), where the mapped vertices are highlighted in red.

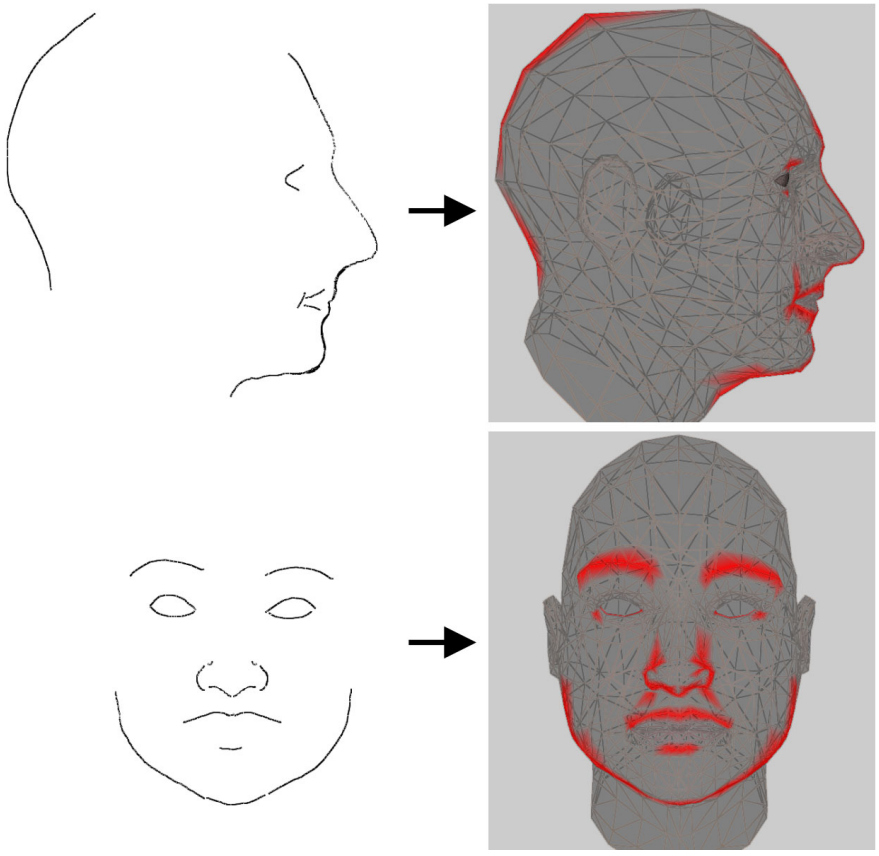


Figure 4.35: Visualising the mapped vertices on the face model (highlighted in red) can be used to verify that the strokes deformed the right set of vertices.

The selected candidate may not be the closest one to the sketched point in screen space. This is because there is limited amount of offline contour data, and therefore a candidate referring to a vertex index might be a better choice than a candidate positioned closer, based on preserving a particular vertex sequence. Therefore, although increasing the size of the threshold that determines the cut-off radius when assigning candidates to stroke points puts more workload on the mapping process, it can help to make better choices when using a limited training set. An additional step is then taken to adapt a selected candidate to the stroke. The translated coordinate is found by taking the screen coordinate for the candidate, and update the XY-coordinates with the one for the stroke point, while leaving the depth unchanged. This new screen coordinate is projected back into world space (using function f) to give the updated coordinate which is assigned to the vertex. This ensures mapping a stroke point to the most appropriate vertex index, while staying faithful to the stroke itself.

The importance of applying the improved grading scheme instead of simply relying on the nearest candidates is emphasised in Figures 4.36, 4.37, 4.39, and 4.40. In particular, mapping stroke points to the nose region is tricky. The region contains a high number of candidates due to its high density of vertices, and as shown in Figure 4.30 the vast range of possible shapes and their positions means the candidates map to a number of different indices.

Figures 4.36 and 4.37 show that by considering the context of the stroke's endpoints, and preserving the vertex structure when traversing through the stroke points, the feature fits the stroke more accurately than when using an euclidean distance metric only.

Figure 4.38 shows how a stroke (black) can often be interpreted either as a complete description of a feature (left) where the endpoints define its boundaries, or as a part of a feature (right) where the endpoints carry no significant meaning. This thesis considers the first choice as a more natural way of resolving the ambiguity by expecting the user to draw the entire feature as opposed to tweaking it which could lead to any number of different interpretations regarding the endpoints, and therefore adding uncertainty to what the user could expect from a stroke. This is reflected in Section 4.6.1.1 where the grading scheme treats the

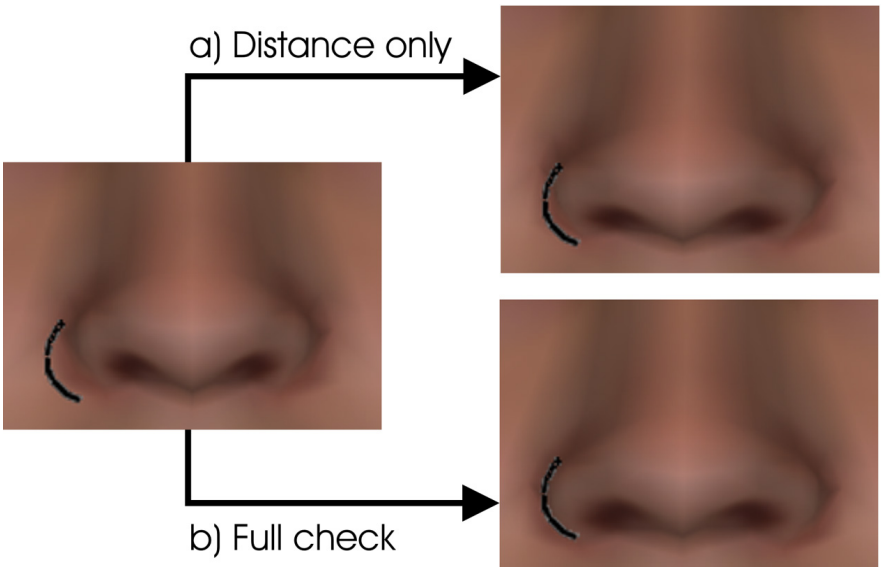


Figure 4.36: The grading scheme correctly identifies the context for the stroke which assures the reconstructed feature fits the sketched stroke accurately.

endpoints as feature boundaries, and is demonstrated in Figure 4.39.

However, the user interface could easily offer the user to choose between the two interpretation modes. Using the second mode could prove useful when a user sketches a stroke where he has already sketched the boundaries for a feature (such as the lip corners) and wants to preserve them while sketching a second stroke relating to the same feature.

Figure 4.40 visualises the importance of the second factor of equation 4.29 (see Section 4.6.1.2). Here it is ensured the shape is preserved by favouring candidates whose vertex is connected to the previously assigned vertex, iteratively adapting the vertex structure to the stroke.

4.6.3 Simple mapping for quick deformations

An alternative approach to using a collection of contour candidates and a grading scheme is to map the sketched strokes to the contours already present in the

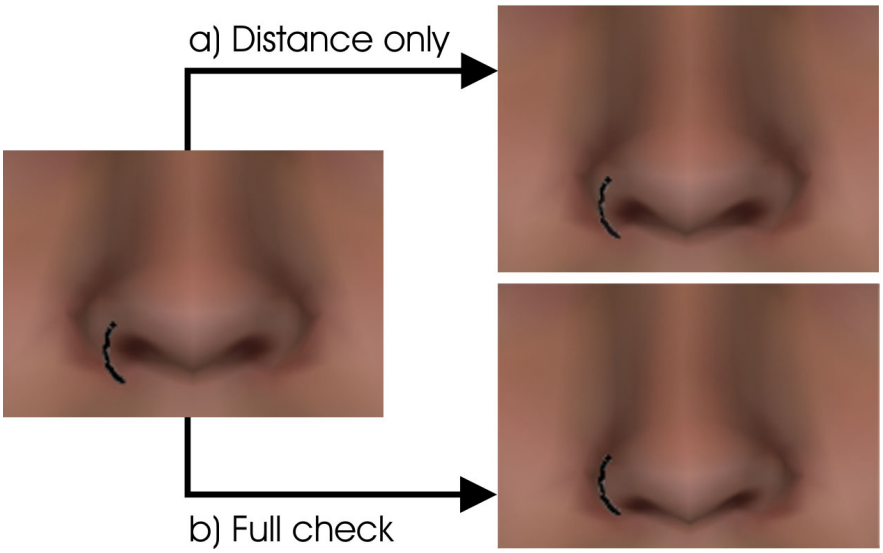


Figure 4.37: The grading scheme correctly identifies the context for the stroke which assures the reconstructed feature fits the sketched stroke accurately.

Figure 4.38: Different interpretations of the same stroke.

current face model. This avoids looking through a large number of possible candidates for each stroke, and instead focuses on mapping them to the most appropriate contours describing the existing model.

To perform a simple test using this method, the sketched points are mapped to the nearest contours in euclidean space. Before, the mapped candidates supplied the strokes with the depth values, but now the depth is has to be estimated through other means. A temporary solution is to create a new screen coordinate using the XY values from the stroke, and the depth from the mapped contour. This screen coordinate is projected into XYZ world space. This is equivalent to translating the mapped contour along the viewing plane until it intersects the stroke. More elaborate schemes could give improved results, but that will not be explored in this thesis.

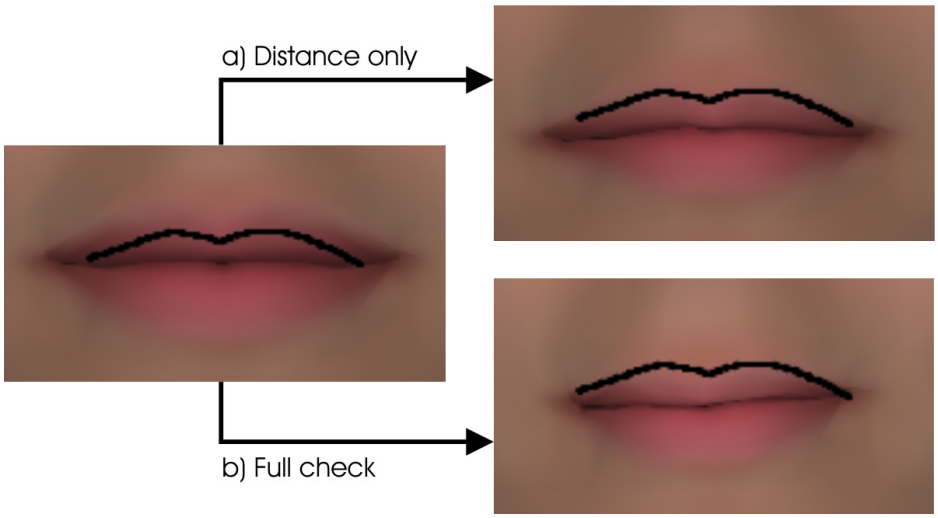


Figure 4.39: The grading scheme identifies the feature boundaries correctly

The mapped vertices go through the same pipeline as before, using the probability model to find the coordinates for the remaining vertices, along with the established options to clean up the vertex structure and correlate unsketched segments. This method cannot create new features, i.e. features not already described by the current set of contours, but it allows the user to quickly deform the model.

4.6.4 Reconstructing sketched features

The system keeps track of the sketching process using a vertex structure identical to the face mesh itself. Initially this structure is filled with -1, which represents missing values. Instead of updating the actual face mesh, this structure is updated every time a vertex is assigned or updated with a candidate chosen by the grading scheme.

The complete face mesh with the sketched features is then reconstructed by using the applying the method described in Section 4.2.1.6, where the assigned data in the vertex structure is used as conditional data to find the expected values for the missing vertices. The face mesh is updated with the complete structure, consisting of the mapped vertices, and the expected vertices.

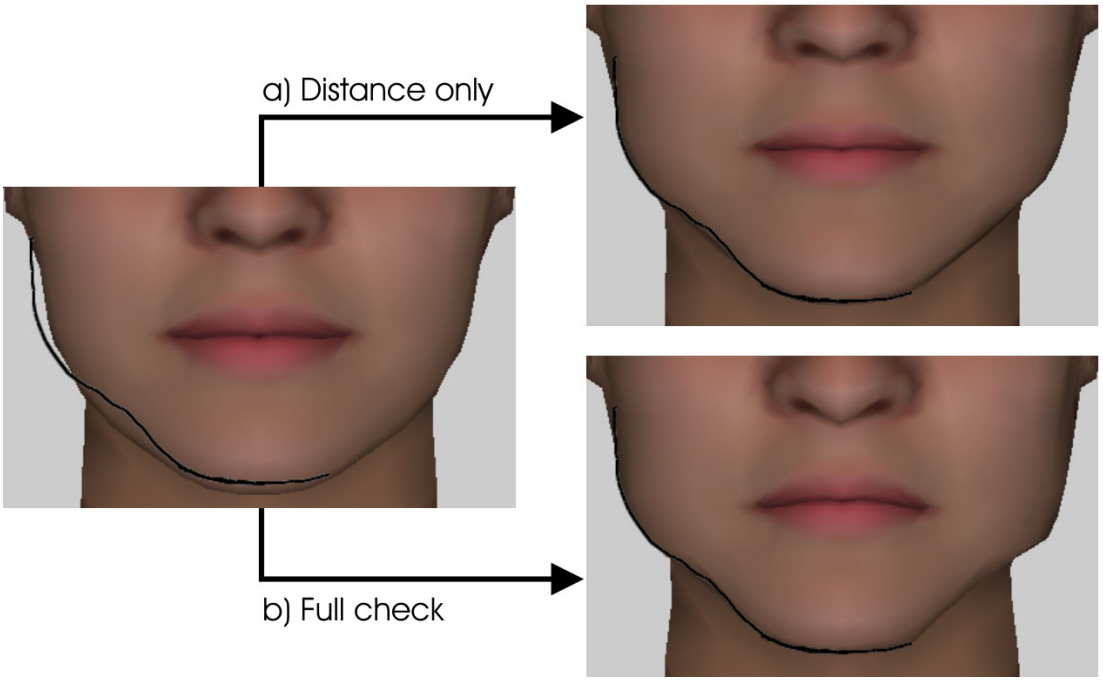


Figure 4.40: The transitional metrics make sure the reconstructed features follow the stroke's shape accurately

It was shown in Section 4.2.5 that one segment of the face can be used to correlate the features for the remaining segments. Figure 4.41 gives an example of this where a part of the lips are sketched and reconstructed (left), where the remaining segments are then correlated. A noticeable difference can be seen where the eyebrows are now raised and the eyes are wider. The chin is rounder and the nose is smaller and more delicate. The original reconstructed face is arguably male, but has been transformed into a female in the correlation process. In general, larger lips tend to be associated with females more often than males.

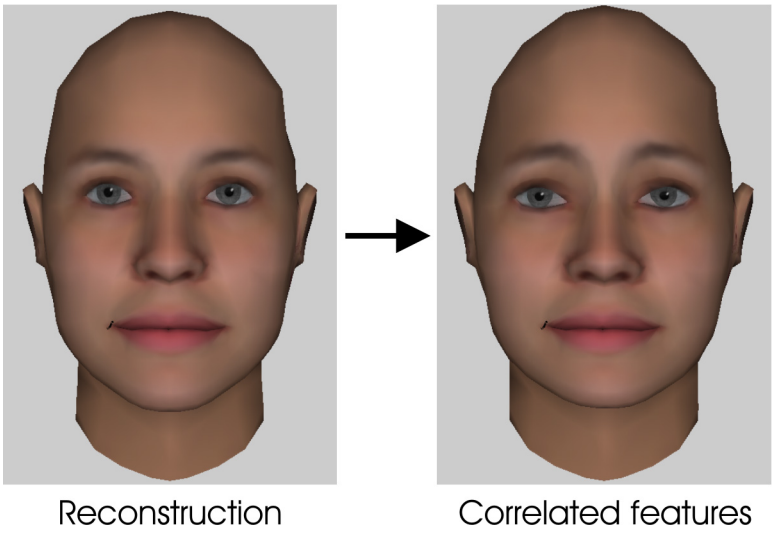


Figure 4.41: Correlating features

4.7 Reconstruction samples: Heuristic approach

4.7.1 Single viewpoint

Figure 4.42 shows how a set of simple lines are used to create a range of different novel face models from the front view. The sketched strokes are shown on a white background for clarity.

Figures 4.43 and 4.44 show examples of sketching an existing source model from different viewpoints, where the base mesh for the source models are displayed in the left column. The main feature outlines are sketched (middle column) and processed to produce a reconstructed model based on the strokes (right column). It is noticeable that the reconstruction may not be an accurate likeness of the original model regardless of whether the outlines for the features seem to fit the sketched strokes. This is due to subtle deviations in the surface curvature from the source model between the outlines because the same stroke can map to more than one possible shape in the higher dimensional feature space.

Figure 4.45 further demonstrates the effectiveness of sketching from the profile view. The user starts with the standard template (left), sketched the desired outline (middle), and the system adapts the model to fit the sketched lines. Achieving this shape is not straightforward using a parameter based system, but

is very intuitive and simple using sketched strokes.

An example of a simple high resolution reconstruction is shown in Figure 4.46. Due to high resolution's superior contour data shown earlier, it allows sketching features that the low resolution models cannot capture. This can also offer better precision, particularly in the nose region where vertices are sparse in the low resolution model. Currently though, the efficiency of the low resolution models has favoured them during the system's development and testing. Additionally, blending the 7 segments of the low resolution model generally does not produce any unwanted artefacts, but the large number of segments needed for the high resolution models causes problems. A possible solution is to apply a multi-resolution decomposition and blending as suggested in Section 4.2.2.

4.7.2 Multiple viewpoints

It was shown in Section 4.5.3 that when reconstructing models from one viewpoint, the probability model has to estimate the depth for unknown vertices. The result depends on what input data was mapped from the sketched strokes, and observed faces in the training set. The generated features may therefore not be what the user is after when they are viewed from another angle.

In the sketching interface, the model can be rotated and sketched from any angle to ensure the features are correct from every viewpoint. Figure 4.47 shows an example of this where features (red strokes) have been sketched and fitted from seven different viewpoints. Strokes are only relevant from the viewpoint they were sketched in (red strokes), and will not make sense when seen from other viewpoints (black strokes). The last two columns in the bottom row show the final model.

4.7.3 Sketching issues and future work

Currently the system only grades candidates based on the structural integrity of vertices for individual strokes which can cause gradual degradation in the reconstructed model due to conflicting observed data in the tracking model. Consider the following scenario: The lips are sketched from the front viewpoint, and then again from the profile view. In both cases, a set of vertices in the tracking model

are given new coordinates based on the mapped contours and then used to find the remaining vertices using the mixture model (face generator) to form a complete structure. However, the vertices affected by the second stroke are likely to overlap or lie close to vertices modified by the first stroke. Updating the tracking coordinates blindly without considering previous strokes can move the vertices in such a way that they violate the vertex structural integrity. If this happens, the corrupted conditional data based on the tracking model will generate skewed expected values when reconstructing the complete model (Section 4.2.1.6), giving unexpected results.

This is reduced by updating the observed coordinates in the tracking model after projecting the reconstructed model onto the face space which can only generate a valid vertex structure (Section 4.2.4). This makes sure that the structure is corrected before sketching the next stroke which helps to prevent the gradual corruption.

Alternatively, this could be rectified by taking into account every nearby stroke when mapping contour candidates to the tracking model where the newest stroke carries the highest weight. This would provide a controlled compromise for the mapping scheme as opposed to the currently unsupervised process where each stroke updates the vertex coordinates independently.

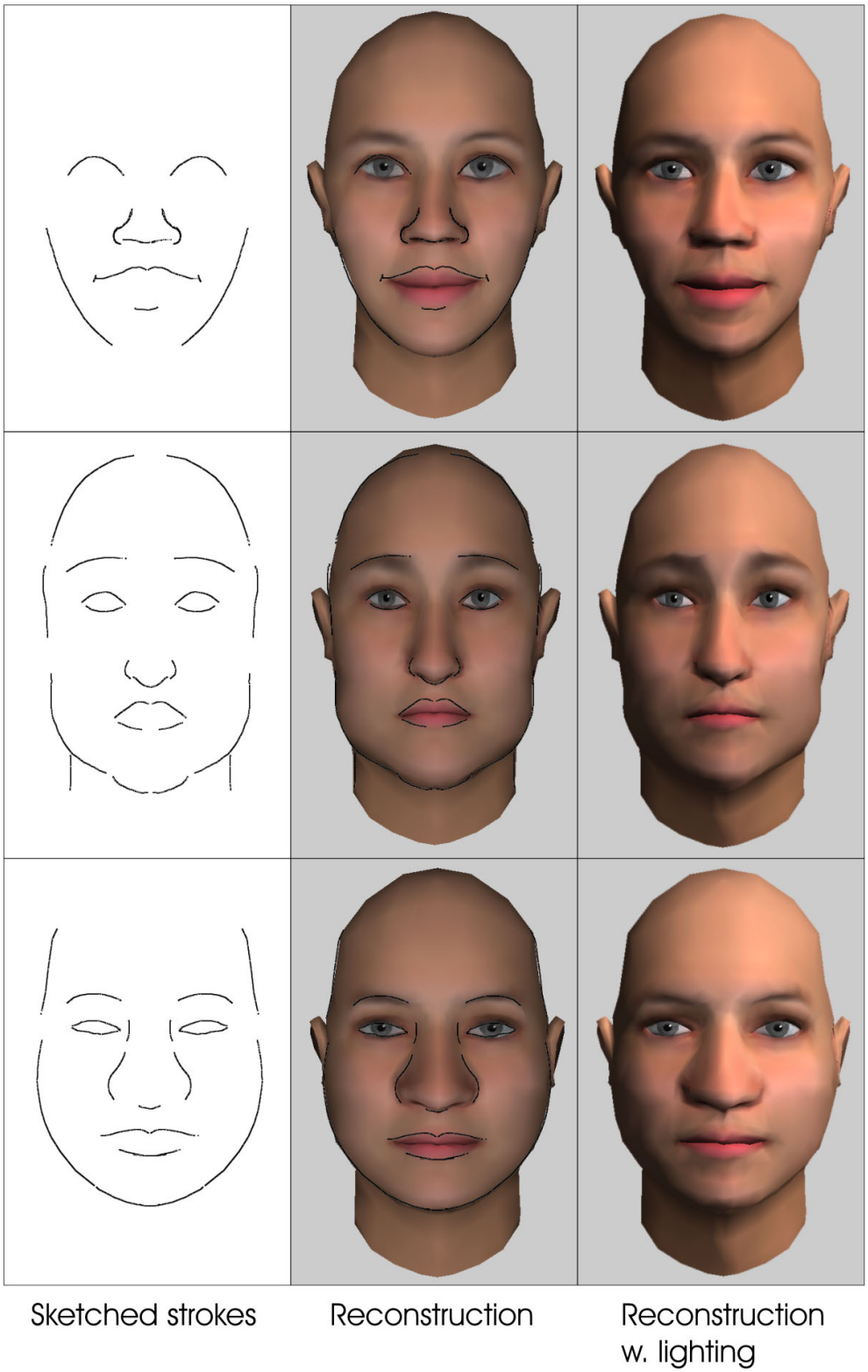


Figure 4.42: Novel models created using few simple strokes from the front viewpoint (the strokes are shown on a white background for clarity)

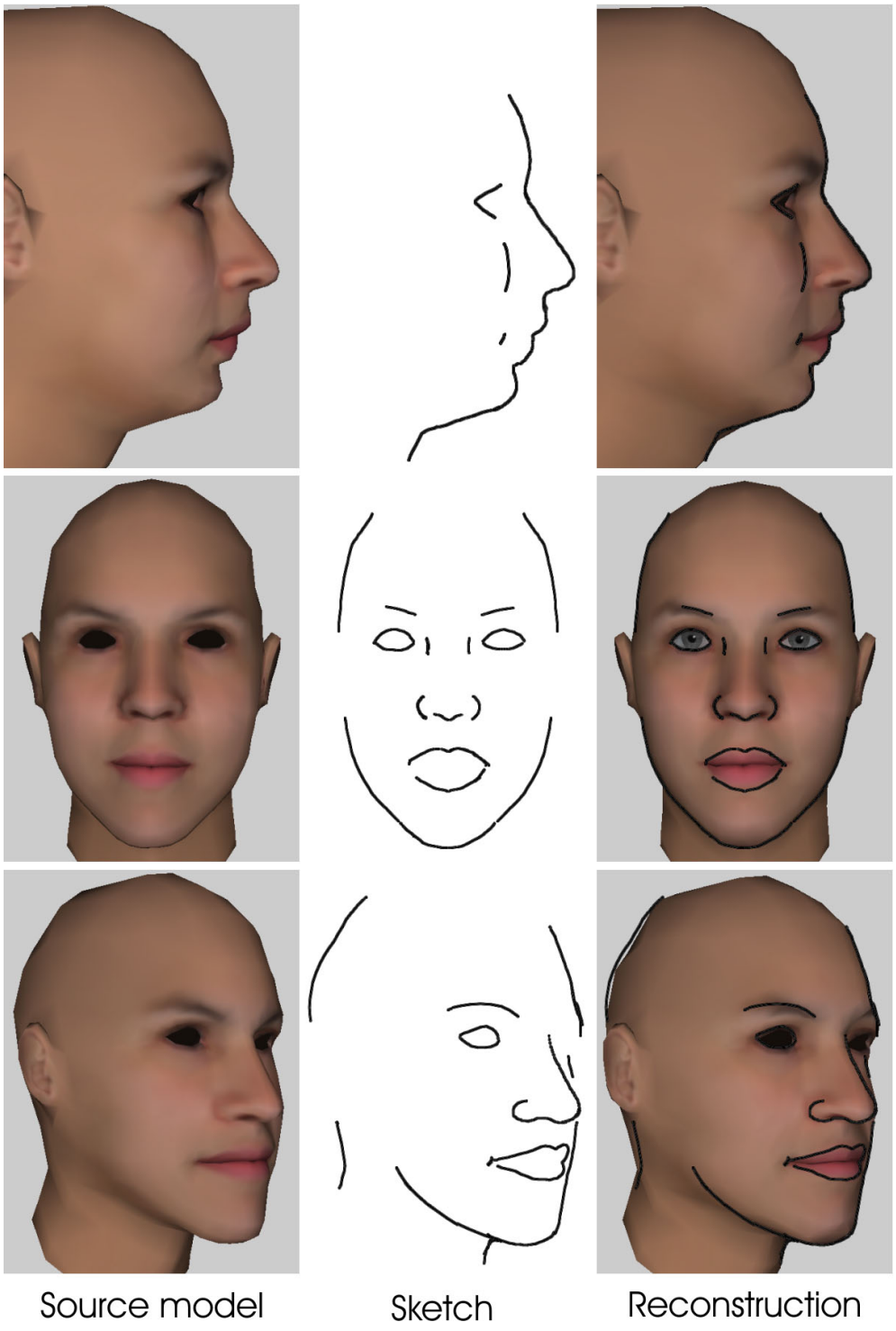


Figure 4.43: Sketching existing faces from different viewpoints

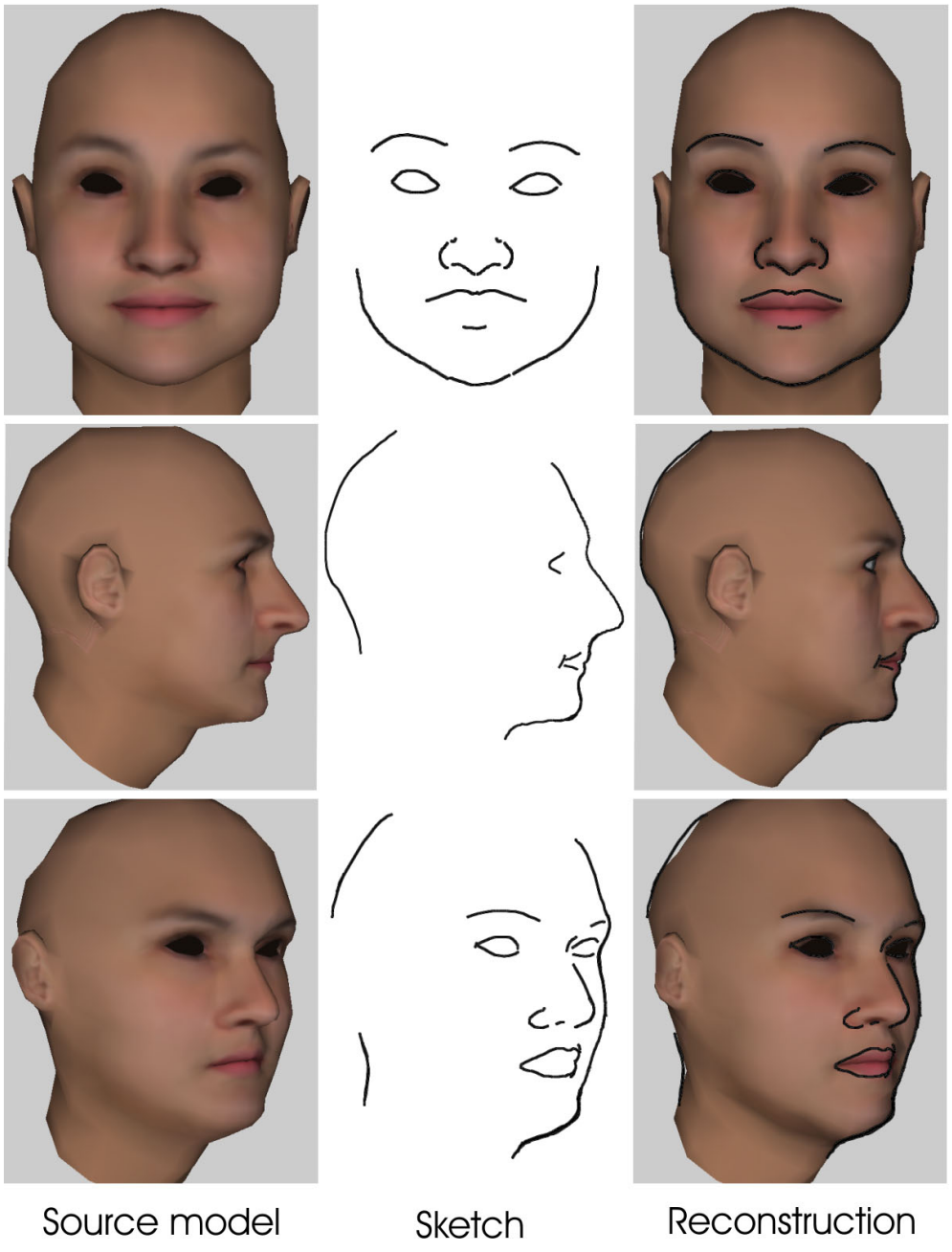


Figure 4.44: Sketching existing faces from different viewpoints (continued)

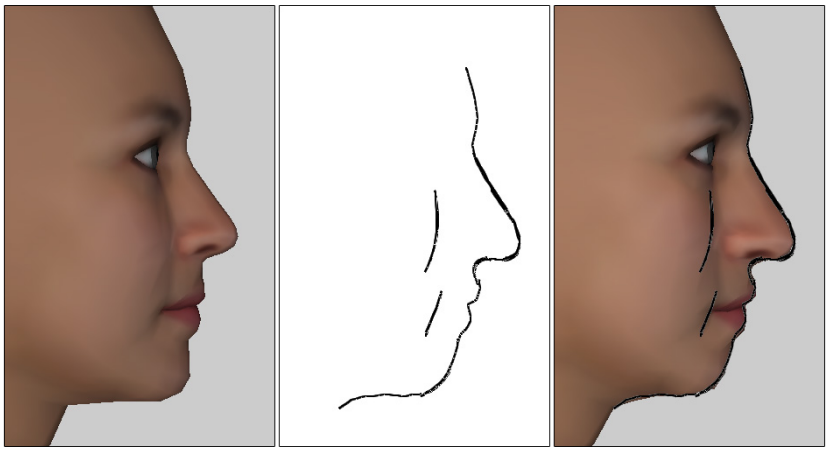


Figure 4.45: Profile sketching is a powerful way to create distinctive models using very few and simple strokes

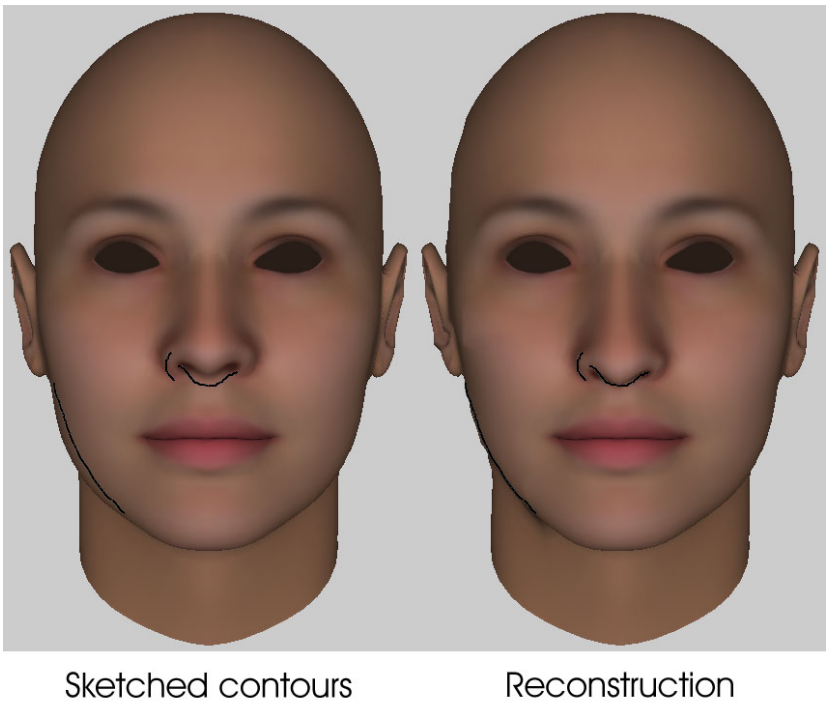


Figure 4.46: Sketching simple features on a high resolution model

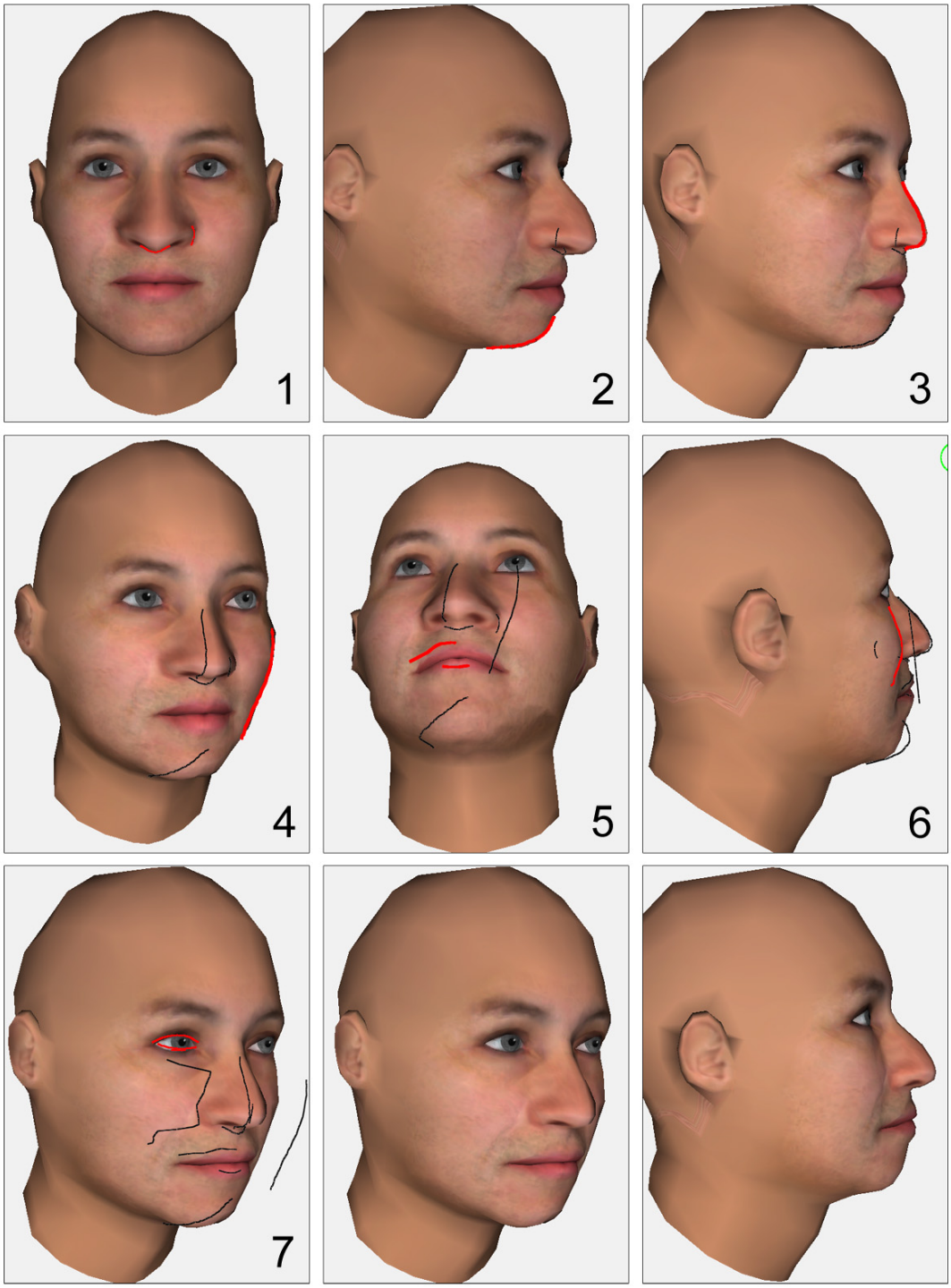


Figure 4.47: Creating a novel face model by sketching from multiple viewpoints

4.8 Reconstructing faces from strokes: Hidden Markov Models

Here, a statistical approach is used to interpret the sketched strokes as an alternative to the heuristic method proposed in Section 4.6. A sketched stroke consists of a sequence of points where the assumption is they map to a corresponding sequence of contour candidates which again map to vertices on the face mesh. The problem is finding this unknown sequence of optimal vertices describing the sketched feature based on an observed stroke. This is naturally expressed as a Hidden Markov Model (HMM) where the goal is to find the most probable sequence of hidden states (contours/vertices) for a given observation sequence (stroke points). This is explained in detail in [Bis07] but is summarised here for clarity with regards to the construction of the probability matrices in Section 4.8.5. The results using this approach is compared in Section 4.9 with the heuristic approach described in Section 4.6.

4.8.1 Hidden Markov Models

A Markov model expresses the joint distribution for a sequence of observations in the form

$$p(\mathbf{x}_1, ..., \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, ..., \mathbf{x}_{n-1}), \tag{4.38}$$

where each of the conditional distributions is independent of all previous observations except N most recent [Bis07].

A first order Markov chain only takes into account the most recent observation $\mathbf{x}_{n-1}$, a second order chain takes the two previous observations into account and so on. Lower order chains put restrictions on the model, while larger ones require a large number of parameters rendering the system impractical.

Using latent variables overcomes this problem as it does not put a limit on the order, and requires only a limited number of free parameters. For each observation $\mathbf{x}_n$ there exists a latent variable $\mathbf{z}_n$, where the latent variables now form

the Markov chain. This forms a state space model whose joint distribution is

$$p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{z}_1, ..., \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{z}_n). \tag{4.39}$$

A Hidden Markov Model (HMM) is a state based model where the latent variables are hidden (or discrete). Three probability distributions are required forming three probability matrices: transition probabilities, initial probabilities, and emission probabilities.

The probability distribution of state $\mathbf{z}_n$ is based on the previous latent variable $\mathbf{z}_{n-1}$ through the conditional distribution $p(\mathbf{z}_n|\mathbf{z}_{n-1})$. Based on K-dimensional latent variables, this forms a matrix $\mathbf{A}$, denoted as transition matrix, where it describes the probability of moving from each state to every other state

$$p(\mathbf{z_n}|\mathbf{z_{n-1}}, \mathbf{A}) = \prod_{k=1}^{K} \prod_{j=1}^{K} \mathbf{A}_{jk}^{\mathbf{z}_{n-1,j}\mathbf{z}_{nk}}. \tag{4.40}$$

Unfolding the transition states over time gives a trellis diagram. The initial state is described using a marginal distribution $p(z_1)$ which is a vector of the initial probabilities π

$$p(\mathbf{z}_1|\pi) = \prod_{k=1}^{K} \pi_{k1k}^{z}. \tag{4.41}$$

The conditional distribution of the observed variables $\mathbf{x}_n$ mapping to a latent variable $\mathbf{z}_n$ is $p(\mathbf{x}_n|\mathbf{z}_n, \phi)$, where ϕ is the distribution parameters. This forms the emission probabilities of the form

$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) \prod_{k=1}^{K} p(\mathbf{x}_n|\phi_k)^{z_n k}. \tag{4.42}$$

Putting this all together gives the joint probability distribution over both the latent and observed variables

$$p(\mathbf{X}, \mathbf{Z}|\theta) = p(\mathbf{z}_1|\pi) \left[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}) \right] \prod_{m=1}^{N} p(\mathbf{x}_m|\mathbf{z}_m, \phi), \tag{4.43}$$

where $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_N\}$, and $\theta = \{\pi, \mathbf{A}, \phi\}$.

4.8.2 Mapping strokes to contours

In order to find the most probable sequence of latent states (contours) for a given set of observed data (strokes), the Viterbi algorithm [Vit67] is employed which is a max-sum algorithm whose complexity grows linearly with the length of the trellis chain. The algorithm traverses through the chain to find the optimal path through the trellis.

Once a stroke points has been mapped to a contour, the vertex indexed by the contour adopts the contour coordinate. The stroke point density is generally higher than the contour density which causes more than one point to be assigned to the same latent state. The contour (latent state) closest to the stroke point is chosen. The following section shows that it is equal to picking the latent state that has the highest emission probability.

4.8.3 Mapping criteria

Finding the optimal sequence of contours (latent states) based on sketched stroke points (observed data) requires three probability matrices as explained above.

The initial probability matrix is responsible for mapping the first stroke point to an initial contour candidate (or state) setting the context for the rest of the stroke as described in Section 4.6.1.1. Therefore the same equations are used here where they form a metric used to create the marginal likelihood distribution.

The emission probability matrix specifies the independent conditional likelihood of the stroke points mapping to a particular contour (state). Here it is simply based on the euclidean distance between the two.

Once the initial contour has been assigned it is very important to select the subsequent contours carefully in order to reconstruct a plausible and noise free facial feature. The transition probability matrix considers three factors when determining how to move from one contour (state) to another. The first two are based on s_1 and s_2 (Equations 4.36 and 4.37 respectively), discussed in Section 4.6.1.2, where s_1 favours contours whose distance and direction matches that of the sketched stroke points, while s_2 makes sure two consecutive contours do not breach the integrity of the mesh vertex structure. Once a contour has been assigned, the sequence of contours to follow should ideally map to connected vertices to form an unbroken chain of updated coordinates, thus avoiding sudden unexpected noise. The third factor favours transitions between contours whose vertices are connected with the shortest possible path, found by applying Dijkstra's algorithm [Dij59] on the polygon structure.

4.8.4 Computing a vertex connectivity map

The transition matrix requires the shortest path between any two vertices on the mesh. For a mesh consisting of n vertices, the first step is to explore the polygon structure and create a $n \times n$ map $\mathbf{S}$ where

$$\mathbf{S}(i,j) = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ share a triangle} \\ 0 & \text{otherwise.} \end{cases} \tag{4.44}$$

A $n \times n$ vertex connectivity map $\mathbf{C}$ specifying the path distance between any two vertices is found by applying Dijkstra's algorithm [Dij59] on every pair (i,j) and then mirroring the results to (j,i). This connectivity matrix is then used as a lookup map at runtime. However, calculating the shortest path using Dijkstra is intractable for a large number of vertices as it goes through every possible path combination through the polygon structure, even if the vertices are directly linked to each other with zero distance. This is resolved by specifying a search scope for each vertex with k number of vertex connectivity rings around it. Figure 4.48 shows $k = 1$ (left) and $k = 2$ (right), where the vertices in the first ring are

labelled with blue circles, and the second ring with red circles.

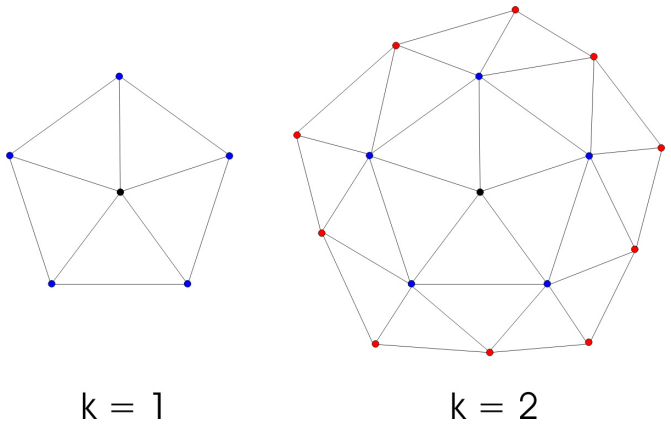


Figure 4.48: Defining a limited search scope for each vertex (black) based on the number of connectivity rings

Now instead of exploring the entire polygon structure to find the shortest path between vertices i and j, the search is limited to the scope of the k-ring for vertex i, iff vertex j exists within that scope. Figure 4.49 shows the connectivity maps for $k = 1..4$ where the axes represent the 775 vertices and each cell (i, j) has a colour coded path length between vertices i and j. The colours range from blue indicating there is no known connection (infinite path length) between the vertices within the given scope, and red which means the vertices are connected with zero path length.

Figure 4.50 gives a clearer picture of individual vertex connections and their corresponding path length by zooming in on the first 100x100 cells in the top left corners of the four connectivity maps in Figure 4.49.

It is clear that using $k = 4$ does not find the path distance between every vertex, but is still capable of finding paths up to the length of 25. Path lengths exceeding the scope of $k = 4$ are not considered feasible when mapping sketched points to contours. Consequently there is no need to process the connectivity map any deeper since the likelihood drops below 0.1 where the path length exceeds 6 (see Equation 4.48).

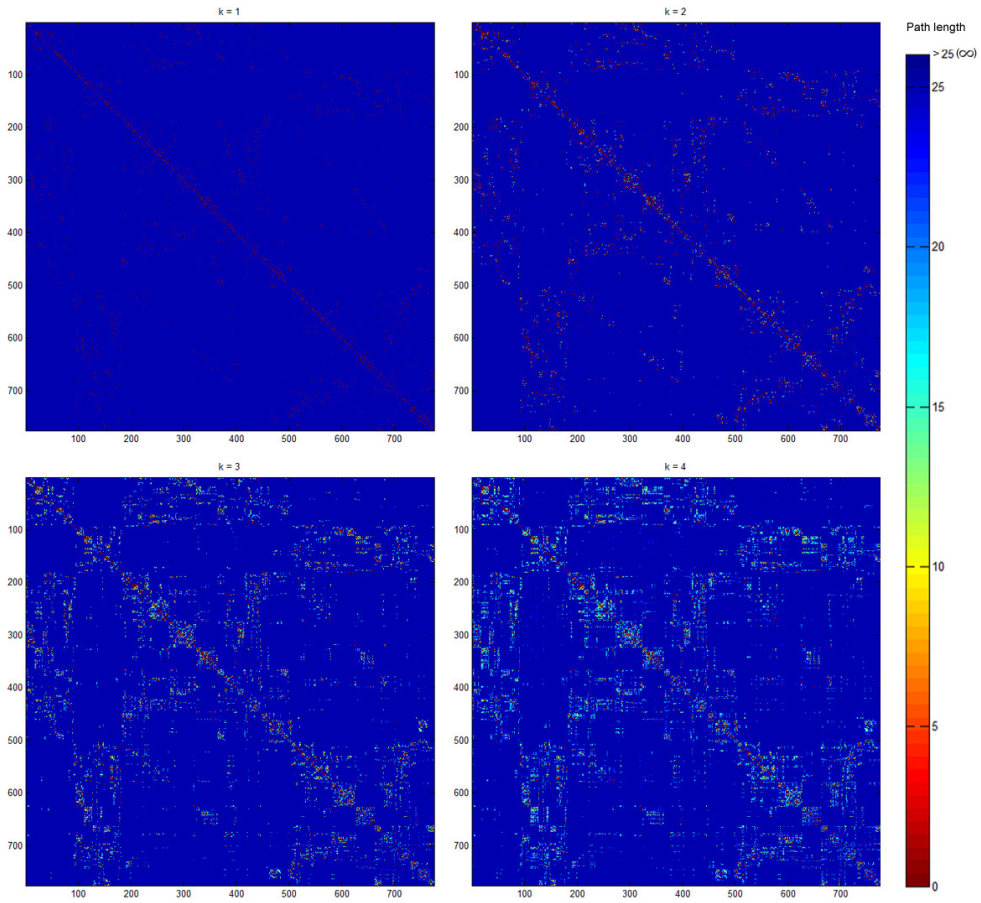


Figure 4.49: Connectivity maps (path length) for different search scopes. Blue indicates infinite path length while red represents zero path length

4.8.5 Constructing probability matrices

The probability density functions are defined based on the criteria given in Section 4.8.3 to form the probability matrices needed to find the optimal contours from the observed stroke points using the Viterbi algorithm.

The set of observed stroke points are defined as P, and $p_i \, \epsilon \, P, i = 1..n$. Similarly the set of contour candidates for the current viewpoint and bounding box is called C, and $c_j \, \epsilon \, C, j = 1..m$.

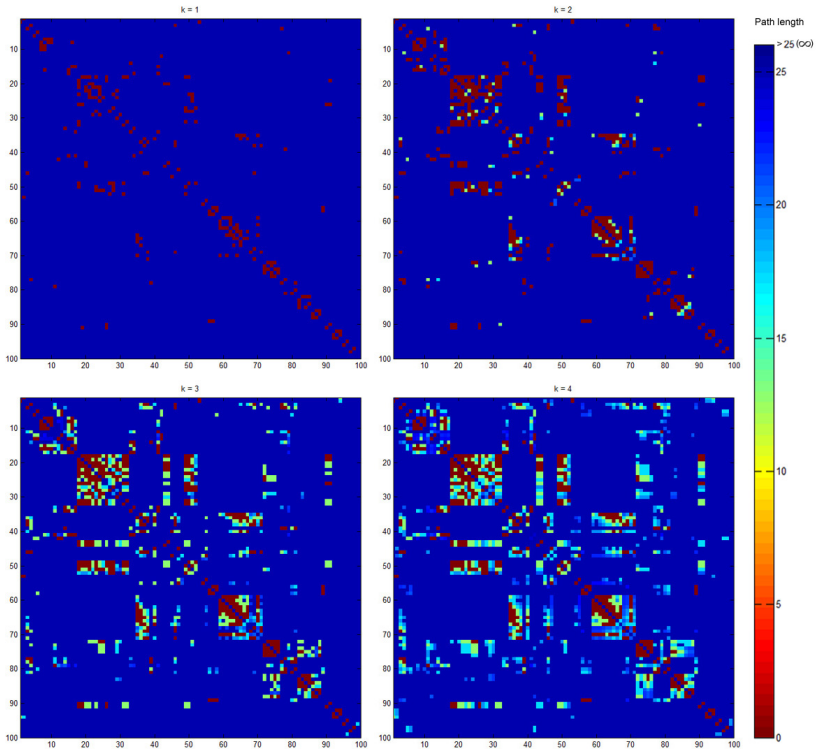


Figure 4.50: A closer look at the first 100x100 cells in Figure 4.49

4.8.5.1 Initial probability matrix

Using Equations 4.30, 4.31, 4.34, and 4.35, the initial metrics $\psi_{sx} = r_x\varphi_x$ and $\psi_{sy} = r_y\varphi_y$ are formed. The $m \times 1$ initial probability matrix is found using

$$\rho(p_1|c_m) \propto \exp\left\{-\frac{1}{2}\left(\frac{\psi_{sx}}{\sigma_s}\right)^2\right\} \exp\left\{-\frac{1}{2}\left(\frac{\psi_{sy}}{\sigma_s}\right)^2\right\}, \tag{4.45}$$

where $\sigma_s = 0.4$.

4.8.5.2 Emission probability matrix

The $n \times m$ emission probability matrix is calculated based on an euclidean distance metric ψ_d between p_i and c_j in screen space

$$\psi_d = \|f(p_i) - f(c_j)\|. \tag{4.46}$$

The emission probability is then estimated as follows:

$$\rho(p_i|c_j) \propto \exp\left\{-\frac{1}{2}\left(\frac{\psi_d}{\sigma_d}\right)^2\right\}, \tag{4.47}$$

where $\sigma_d = 2\tau$, and τ is a screen space coefficient found by measuring the screen distance between two fixed vertices in the original face mesh. This coefficient grows when the user zooms in to compensate for increased distances between the contour candidates in screen space.

4.8.5.3 Transition probability matrix

The transitional probability is a $m \times m$ matrix which defines the probability whose jq-th cell defines the probability of moving from contour c_j to contour c_q. Computing the probabilities is based on three factors as discussed in Section 4.8.3, consisting of three corresponding metrics ψ_c, ψ_{stroke}, and ψ_{skel}.

ψ_c defines the vertex distance between the vertices to which c_j and c_q map to and is found by looking up $\mathbf{C}_{jq}$. The probability distribution function for the connectivity metric is defined as

$$\gamma_1 \propto \exp\left\{-\frac{1}{2}\left(\frac{\psi_c}{\sigma_c}\right)^2\right\}, \tag{4.48}$$

where $\sigma_c = 3.0$.

ψ_{stroke} is based on s_1 (Equation 4.36) and takes the stroke direction and distance into account by comparing the transition between two contour points and corresponding stroke points. The nearest stroke point is assigned to each contour since it is not known beforehand which point maps to them. This gives an estimate which is good enough since only the 2-dimensional screen coordinates are used for the points. ψ_{stroke} is found by calculating the ratio between the distance between the contours c_j and c_q, and the distance between their corresponding

stroke points p_j and p_q

$$\psi_{stroke} = \frac{\|c'_q - c'_j\|}{\|f(p_q) - f(p_j)\|} - 1, \tag{4.49}$$

where c' is made up of the XY-screen coordinates found using f, and the Z-world coordinate. A ratio of 1 is found if the two vectors have the same length with no change in depth. Since equal displacement with minimum change in depth makes for an ideal candidate, one is subtracted from the ratio as 0 will return a likelihood of 1.0 using the probability distribution function

$$\gamma_2 \propto \exp\left\{-\frac{1}{2}\left(\frac{\psi_{stroke}}{\sigma_{stroke}}\right)^2\right\}, \tag{4.50}$$

where $\sigma_{stroke} = 0.5$.

ψ_{skel} is based on s_2 (Equation 4.37) and is responsible for maintaining the vertex structure integrity by comparing the screen displacement between the contours c_j and c_q to the screen displacement of the vertices they map to on the face mesh. A large distance between the two vectors indicates moving from c_j to c_q does not follow the same path on the vertex skeletal structure. It is found by measuring the euclidean distance

$$\psi_{skel} = \|(f(c_q) - f(c_j)) - (f(g(c_q)) - f(g(c_j)))\|. \tag{4.51}$$

The probability distribution function for this metric is calculated using

$$\gamma_3 \propto \exp\left\{-\frac{1}{2}\left(\frac{\psi_{skel}}{\sigma_{skel}}\right)^2\right\}, \tag{4.52}$$

where $\sigma_{skel} = \tau/6.5$. The associated transitional probability matrix is then found by combining the three metrics in the conditional distribution

$$\rho(c_q|c_j) = \prod_{m=1}^{3} \gamma_m. \tag{4.53}$$

4.9 Comparing stroke interpretations

This section contrasts the two methods covered in this thesis that interpret sketched strokes by mapping them to contours, the heuristic approach (see Section 4.6), and the HMM approach (see Section 4.8.1). Additionally, it will show the validity of employing each transition metric in Section 4.8.5.

Figure 4.51 starts by showing a weakness in the heuristic approach. Sketching internal features such as lips from this viewpoint often proves very tricky to interpret and reconstruct as the picture on the left shows. Adding the vertex connectivity functionality to the heuristic approach by multiplying the grade with the path distance between assigned vertices (Section 4.8.4) greatly improves the identified contours and the reconstructed mesh (middle). However, the HMM model gives a better overall fit to the sketched strokes.

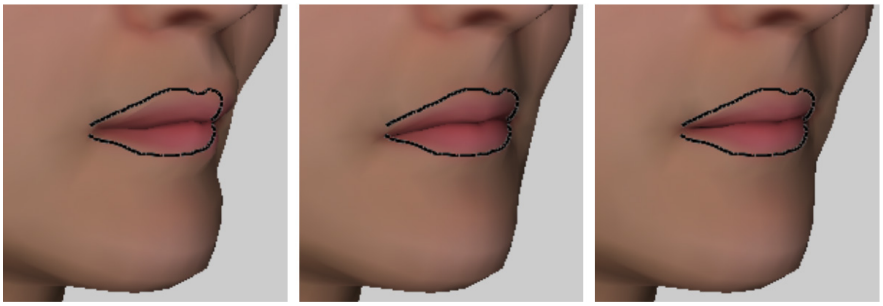


Figure 4.51: Demonstrating the effectiveness of incorporating the connectivity metric to fit a feature accurately to the sketched strokes from difficult viewpoints

Another common problem when sketching from this angle is demonstrated in Figure 4.52. Here the system assigns stroke points to contour points whose vertices are associated with two different features, the cheek and the nose, giving unexpected results. The HMM model using the connectivity metric explained in Section 4.8.5.3 (right) correctly stays faithful to the stroke's intention and correlates the nose based on the new cheek profile, while omitting the connectivity metric (middle) and using the heuristic approach incorporating the connectivity

matrix (left) fails to do so. Each of the three reconstruction samples in Figure 4.52 shows the model reconstruction before (left) and after (right) projecting it onto the face space to correct structural errors (see Section 4.2.4). Inaccurate, unrealistic reconstructions will result in a bigger distance between the reconstruction and the projected version where the projected features will differ from those of the reconstruction. Improved reconstructions are therefore important to maintain stable interpretations to the sketched strokes.

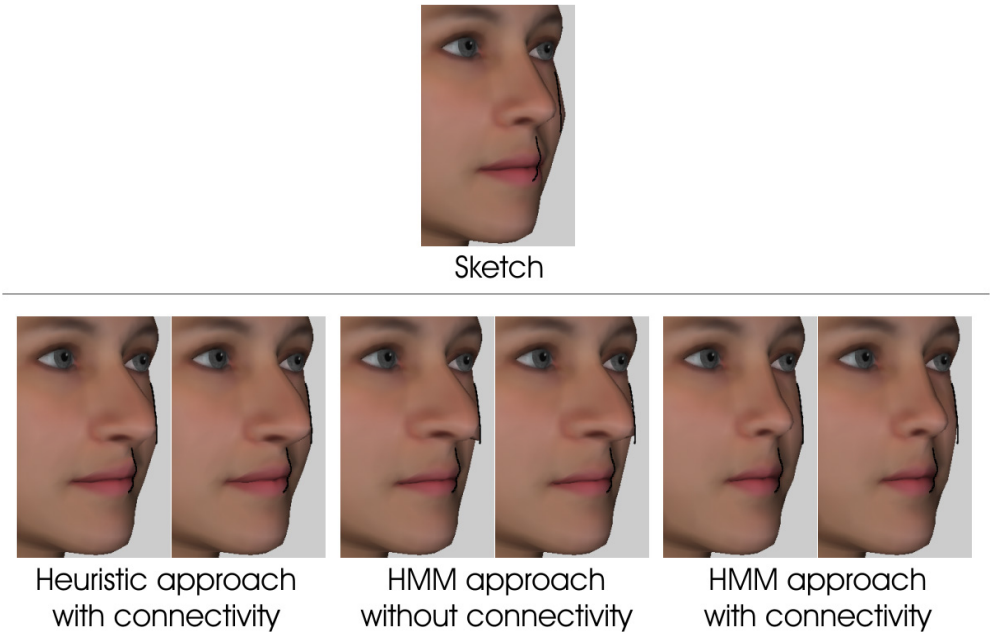


Figure 4.52: Demonstrating the effectiveness of incorporating the connectivity metric to avoid mapping the same stroke to two unrelated features

Figure 4.53 shows an example of comparing the interpretation and reconstruction based on a simple sketched stroke. All the models are shown without projecting them onto the face space for clean-up. The top row shows the textured reconstruction of each method (from left to right); the heuristic approach, the HMM approach using only the ψ_{stroke} metric (s_1) to calculate the transition probability, the HMM approach using both ψ_{stroke} and ψ_c, and the HMM approach using all the metrics. The results from each method seems to match the stroke well, but examining the vertex structure in the middle row shows a clear difference

in quality. The affected vertices are highlighted in red, and the selected contour candidates are shown as green points. Here distortions in the mesh structure are visible where the mapped points (green) are inaccurately placed causing unrealistic vertex distribution. This is further emphasised by looking at the mapped contours from above. Adding more transition factors puts the contours in place by restricting the selection to more suitable candidates.

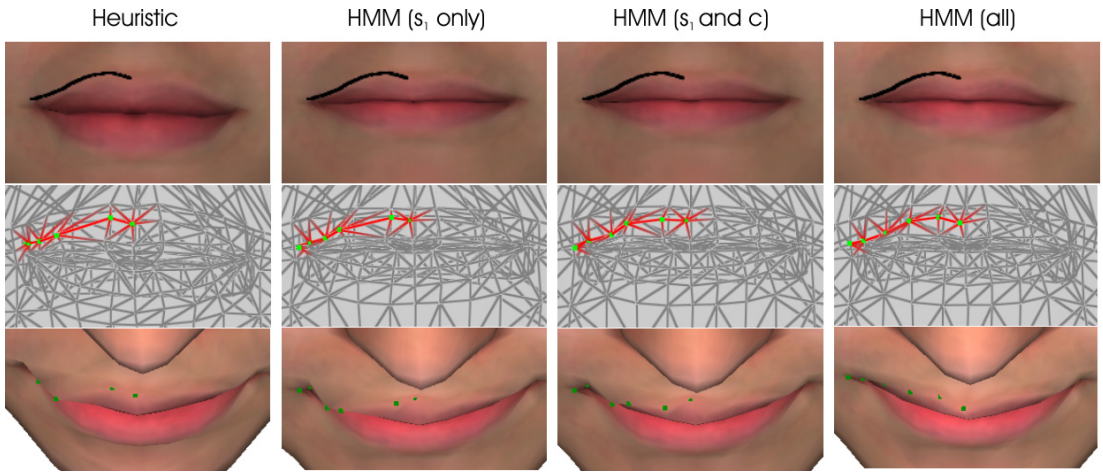


Figure 4.53: The effect of applying different metrics to the mapping process and hence reconstruction accuracy

The advantage of using the vertex connectivity metric is clearly illustrated in Figure 4.54. Again, the reconstructions have not been projected onto the face space in order to visualise every unwanted artefact. The results from both HMM models seem nearly identical when seen from the front, but other viewpoints immediately demonstrate the disadvantage of omitting the connectivity factor.

4.10 HMM sketch examples

Figure 4.55 shows a reconstructed face model with a number of sketched strokes defining the model's features. As noted earlier, the strokes are view-dependent and will only make sense when seen from the viewpoint it was created in. Strokes

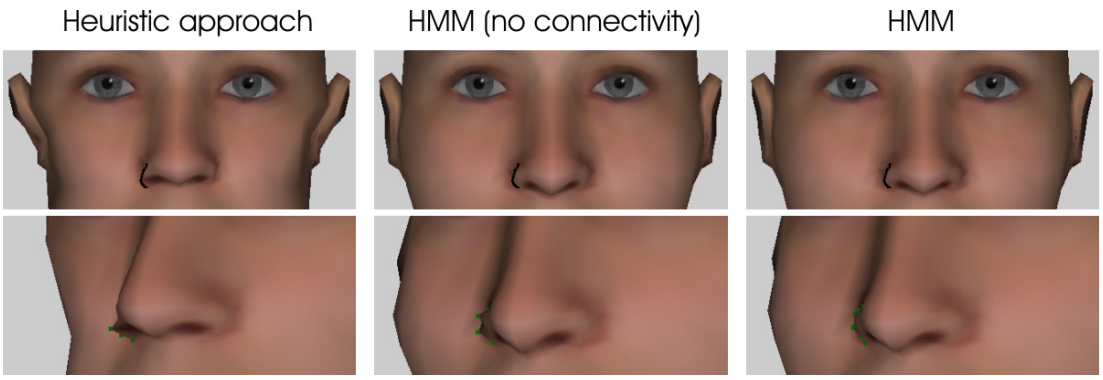


Figure 4.54: The HMM approach gives consistently better results in trouble areas making it less dependent on segment clean-up (Section 4.2.4) to produce smooth and natural looking faces.

matching the viewpoint are highlighted in red while the remaining strokes are black. Figure 4.56 provides an example of using feature correlation to either speed up the modelling process, or to look up individuals in a database by sketching few descriptive features where the system quickly converges onto a specific subject.

Sketching issues were discussed in Section 4.7.3 wher they were caused by the fact that each stroke is processed independently and therefore no attempt is made to treat conflicting strokes using a controlled procedure. The same problems apply here since no such procedure is in place as a part of the HMM approach. The main issue is caused by strokes lying near each other, often intersecting due to oversketching or multiple representations from different viewpoints. When this happens, the strokes map to the same vertices or different vertices where the path length between them is small.

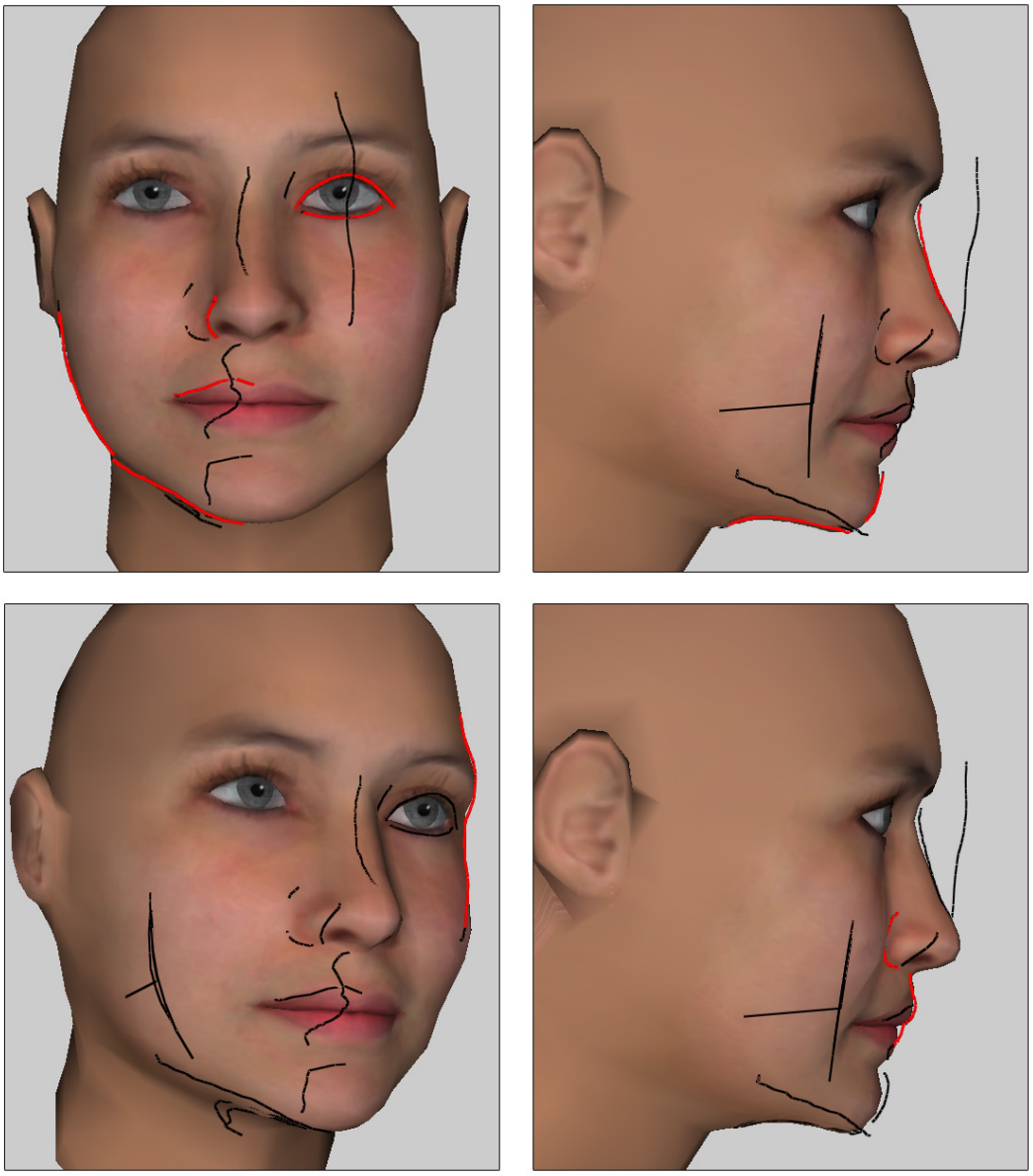


Figure 4.55: Sketching a novel face from multiple viewpoints using the HMMs approach

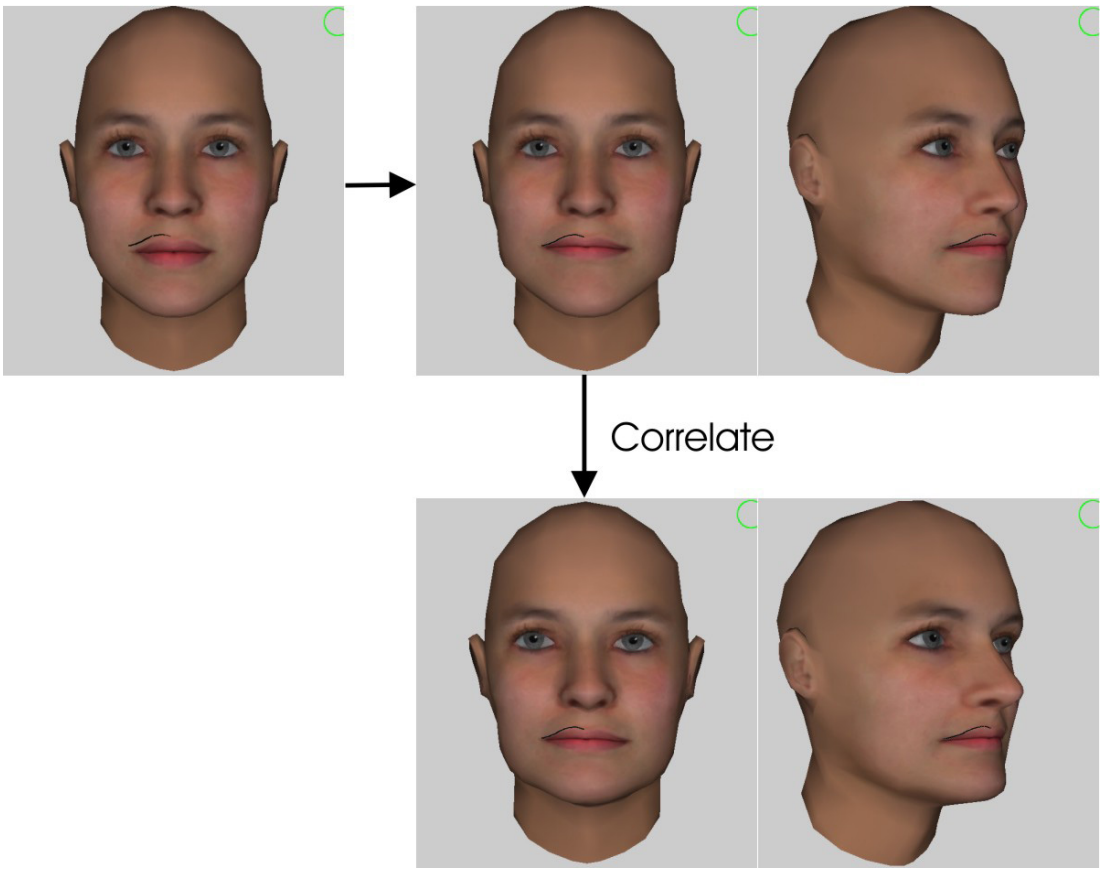


Figure 4.56: Correlating features for more robust modelling or face look up

4.11 User based system evaluation

Section 4.6 showed how a 3D face model can be reconstructed from one or more sketched strokes, where a number of sketch-oriented metrics facilitated a heuristic function which determined the mapping of sketched points to a series of contours describing particular facial features. Section 4.9 demonstrated a few examples which show how the accuracy of the reconstructed features is improved by defining the metrics as simple probability distribution functions and employing them in a HMM framework. This approach was laid out in Section 4.8 which concluded by giving an example of using sketched strokes to create a new face model.

However, all the examples given in this thesis are created by the author and while the system may respond well when used by a person that knows the system's capabilities and limitations, it does not give a clear indication of how an arbitrary user may fare when interacting with the system. A user evaluation was performed which serves as a valuable resource in determining how the system could be improved further to make it a more useful and practical tool for the everyday user. Section 4.11.1 describes the evaluation set up and parameters, and section 4.11.2 discusses the evaluation results and draws conclusions from it.

4.11.1 Set up

The evaluation was conducted with 9 volunteers where all of them were computer literate but none of them has used a sketch-based interface before, and none of them were experienced sketch artists. The volunteers took on the role of a user trying to reconstruct a particular face model using the sketch-based interface. Each user was shown a tutorial video demonstrating the system's main functions. The users were then asked to complete the following 2 tasks:

1. Sketch a face model based on the following description: The face has a slim jawline and thin lips. The eyes are quite larger and closer together than most peoples. The nose has wide nasal wings and it has a slightly protruding bit running from the eyes to approximately half way to the nose tip. But the nose tip still protrudes a little after this.

2. Sketch a face model based on two pictures of an existing model. The model was generated using the system and showed the front and profile view of the model (see Figure 4.57).

The verbal description in task 1 described the target model in task 2, unbeknownst to the users. This allows a comparison of the two modalities where the first one is an abstract description which is likely to manifest differently in the mind of each user, while the latter one is unquestionable physical evidence. During the evaluation however, it was apparent that the users interpreted the physical evidence differently which sometimes led them to make unnecessary or incorrect changes to the facial features. The test was performed on a Toshiba M700 TabletPC using a digital pen (digitizer). After completing each task the users were asked to fill

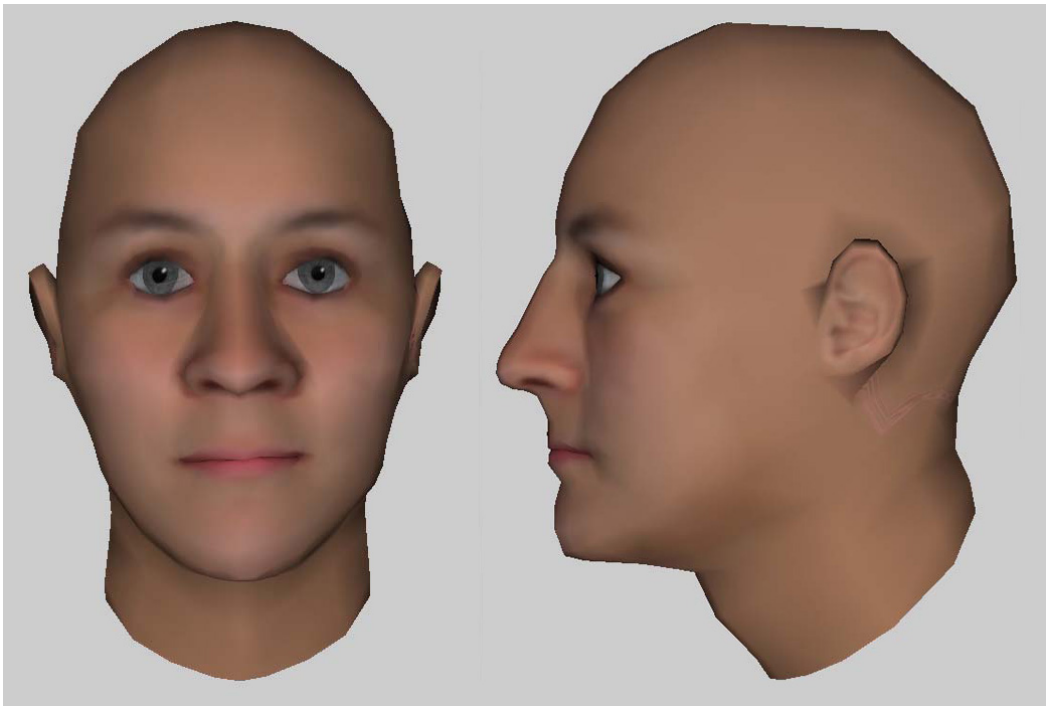


Figure 4.57: The target face users were asked to reconstruct

out a questionnaire about their experience using the sketching interface for the particular task. The questionnaire consisted of 6 questions where the volunteers scored each question using a five point scale (1-5). This is summarised in table 4.11.1. Furthermore, the users had the opportunity to write their own comments

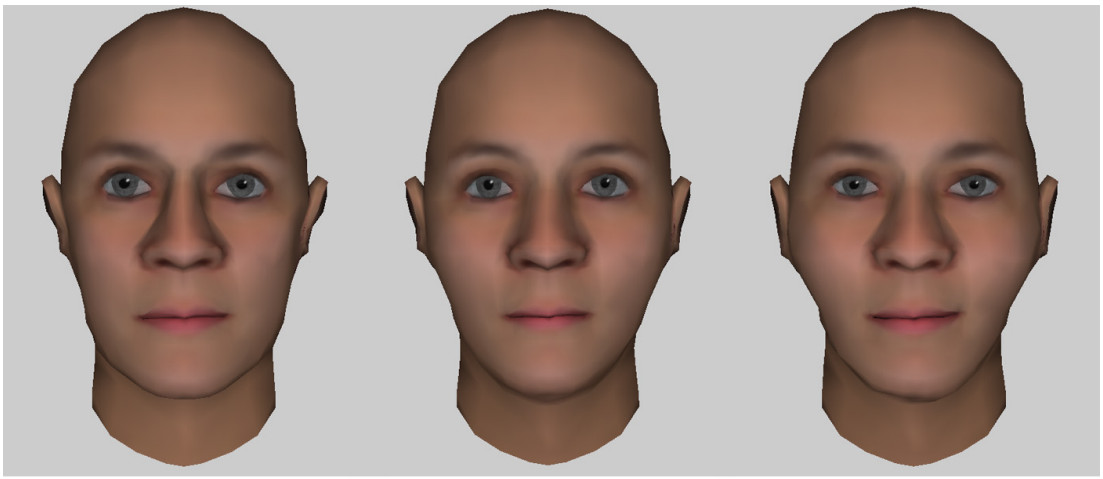
No.	Question	Score = 1	Score = 5
Q1	How much experience do you have of sketching?	No experience	Very experienced
Q2	How easy is it to use the sketching interface?	Not easy at all	Very easy
Q3	How easy is it to sketch strokes?	Not easy at all	Very easy
Q4	How easy is it to create new facial features?	Not easy at all	Very easy
Q5	Did the strokes generate the facial features you expected?	Never	Always
Q6	Are you pleased with how well your sketched face resembles the original description?	Not pleased at all	Very pleased

Table 4.2: Evalution questionnaire

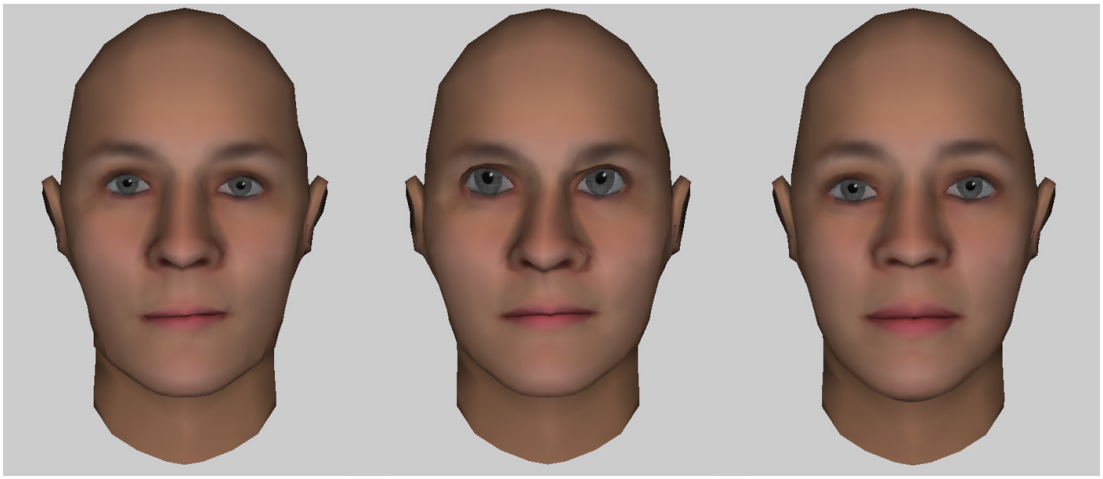
at the bottom of each questionnaire to provide additional feedback.

4.11.2 Results and discussion

The average time taken to complete each task was 10 minutes. The models from each task were stored and compared to the original target by calculating the model error ξ which is defined as the combined vertex difference between a sketched model and the target model. Figure 4.58 visualises the three models with the smallest ξ from task 1 (top) and task 2 (bottom). Appendix C provides a complete list of the models generated for each task along with a difference map. Tables 4.2 and 4.3 show the scores as rated by the users for task 1 and 2 respectively, as well as the model errors ξ. The bottom two rows show the mean values and standard deviations (Std) for the users' answers and model errors. Figures 4.59 and 4.60 visualise the score distribution for each question in task 1 and 2 respectively, by summarising the choices from every user where the scores represented by a distinct colour. None of the questions were rated with the lowest score (blue). The mean value for question 2 (Q2) is higher in task 2 which is to be expected since the users have become more familiar with the system



Task 1



Task 2

Figure 4.58: Sketched models which have the lowest model error ξ.

User	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	ξ
1	1	5	5	4	4	4	8995.4
2	4	3	4	3	3	4	9701.2
3	4	4	4	2	3	4	2543.2
4	2	3	4	3	3	3	9515.4
5	3	4	4	4	4	2	6121.5
6	2	4	5	5	4	4	8658.8
7	1	4	5	4	4	4	8445.6
8	3	3	5	5	4	5	4259.5
9	3	5	3	4	4	4	8898.0
Mean	2.56	3.89	4.33	3.78	3.67	3.78	7459.8
Std	1.13	0.78	0.71	0.97	0.5	0.83	2556.6

Table 4.3: User scores collected from the questionnaires for task 1

User	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	ξ
1	1	5	3	4	4	4	6545.4
2	4	4	4	3	4	4	2987.7
3	4	4	4	4	3	4	8283.0
4	2	3	3	3	4	4	8625.4
5	3	2	2	3	3	2	7013.0
6	2	4	4	5	4	4	5612.8
7	1	5	4	4	4	4	6403.4
8	3	5	5	4	3	4	6789.8
9	3	4	4	4	4	4	6230.7
Mean	2.56	4.00	3.67	3.78	3.67	3.78	6499.0
Std	1.13	1.00	0.87	0.67	0.5	0.67	1629.8

Table 4.4: User scores collected from the questionnaires for task 2

and its controls after completing task 1. The model error mean and standard deviation are lower for task 2 which suggests the natural assumption that the pictorial description provided with that task delivers a superior mental image of the target features than a verbal description. Furthermore, it was observed that the verbal description had further implications on users having little experience in sketching human faces. When asked to sketch larger than normal eyes (as stipulated in task 1), most users resorted to sketching eyes of unnaturally large proportions which the system could not recognise. These extreme strokes often extended to the forehead, cheeks, and the middle of the nose bridge. In contrast, when the same users were asked to use the pictures as a guide in the second task, their strokes were more moderate and realistic which gave more accurate and predictable results. The same problem was encountered while creating a

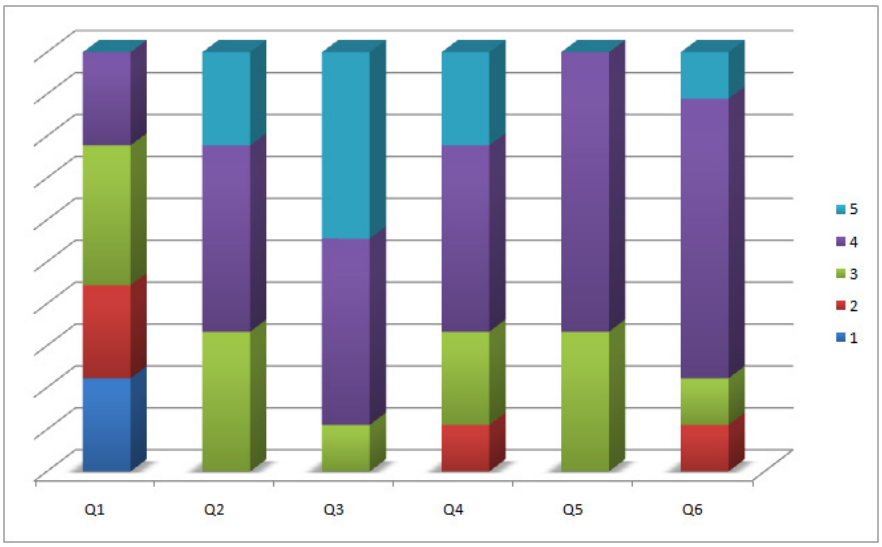


Figure 4.59: Score distribution for each question in task 1

variety of features. The nose was particularly troublesome for many users as they sketched caricature like features. The slim jawline was often portrayed through unnaturally extreme strokes and sometimes using straight lines. This observed behaviour indicates that the typical inexperienced user expects a line to represent a gesture like control which adapts the mesh as opposed to a direct correspondence to the shape and curvature of the reconstructed feature. Only two users sketched the strokes according to how they wanted the reconstructed feature to look.

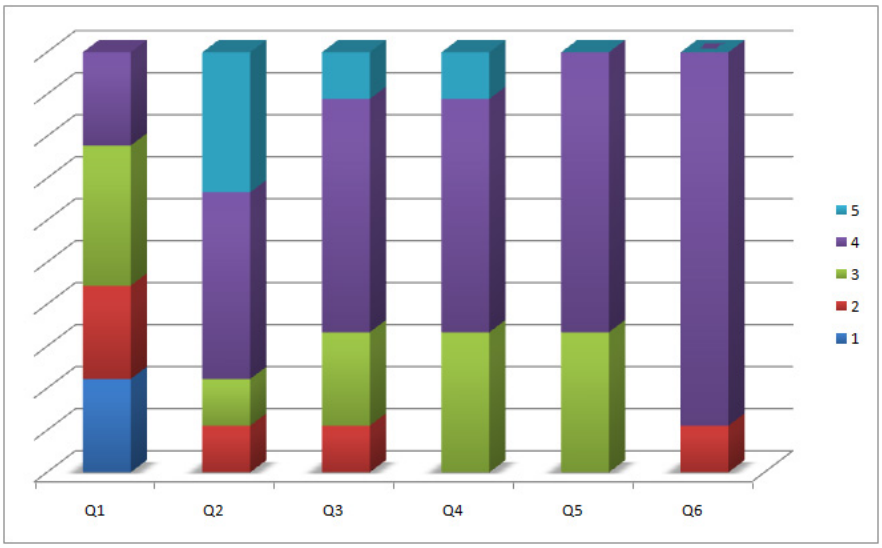


Figure 4.60: Score distribution for each question in task 2

Another common behaviour among the users was observed. If a user sketched an extreme stroke which was not recognised by the system, the users often sketched a more extreme stroke to convince the system of their intentions. This sometimes led to a stroke representing an upper eyelid to reach the forehead. Therefore, this is the opposite of how the system behaves whereby a more moderate stroke would be more effective.

There are two ways of tackling this issue. Firstly, the system could be altered to cater for very extreme strokes by attempting to find the closest match in the realm of known features. This could be fashioned in such a way that the system proposes an alternative stroke based on the extreme stroke where the user can either accept the system's interpretation or sketch a new stroke. This way the user gets an idea of how his stroke is interpreted before reconstructing the 3D feature which could potentially improve the likeability factor. Secondly, the system can guide the user better towards making a more realistic set of strokes by visualising the range of sketch-able features.

The lips were the only facial feature that was consistently accurate across all users. This is probably because they were asked to sketch thin lips which already assigns the current lips as boundaries, thus making sure the users sketch something plausible with regards to the system's prior knowledge.

The user satisfaction according to Q5 is not always a good indicator of the user's corresponding model error ξ because in some cases users either misunderstood the verbal description or sketched strokes that did not represent any target feature. However, the users often seemed satisfied with the change and moved on to the next feature if the system would fit an accurate feature to the stroke. A good example of the former scenario where a user misunderstands the verbal description is user 6 during task 1. The user sketched small eyes and big lips where the system accurately reconstructed the corresponding features. An example of the latter case is user 3 during task 2 where he sketched a line from the profile view which cut the nose in half, creating a flat nose bridge with a non-protruding tip.

The average satisfaction of the final model as measured by Q6 was the same for task 1 and 2, while the deviation is slightly lower for task 2 which indicates the users were marginally more pleased with their model reconstruction in task 2. This may seem counter-intuitive since the target is in the mind's eye in task 1. Therefore, it would presumably be easier for the users to reconcile their expectations irrespective of the accuracy contained in the sketched features. However, the image base description given in task 2 does outperform the verbal description both in terms of smaller average model error and variance, as well as providing a better guide to sketching more moderate and accurate strokes as pointed out earlier.

Future considerations and improvements touch on two different aspects. The first one involves the evaluation process and how things could be done differently in order to get better data for analysis, while the second one probes into technical improvements based on the observed behaviours and pitfalls from this evaluation. The following suggestions are aimed at technical changes based on observing the users in this evaluation process:

- Make the interface more commercially ready, e.g. better undo and history features, reset viewpoint.
- Interpret extreme and crude strokes to offer the user a more viable stroke based on the prior knowledge. The user can then accept or adapt the stroke

without having to reconstruct the 3D feature before experiencing how the stroke will be perceived by the system.

- Visualise sketching boundaries to guide the user while sketching features to avoid unnatural or unsupported strokes based on the viewpoint.

- Oversketching is a natural way of sketching repeatedly until the desired feature has been reconstructed. However, earlier strokes may have been assigned to neighbouring vertices and can therefore compromise a new stroke by providing conflicting information (this was mentioned in Section 4.7.3 and elaborated further in Section 4.10). Methods should be implemented to examine and deal with potential complications arising from this problem.

- Fix a problem which sometimes causes a stroke to be projected incorrectly onto the surface. When this happens, it looks normal from the current viewpoint but the projected depth is off target causing the stroke to lie far away from the model surface. The algorithm mapping strokes to vertices culls stroke points lying too far away to limit the search scope. Therefore, the system ignores any strokes that get incorrectly projected.

The following suggestions are pertinent to improving the evaluation process:

- Ask the users to start by sketching the target face on paper which serves as a way of measuring their sketching skills and sense of shapes. The quality of the paper sketch could be measured subjectively using a survey to find if there is a correlation between the quality of the drawing and the 3D model error. It is of interest to see if the sketching interface could augment the users' sketching abilities.

- Give the users small step by step tasks before the main tasks to familiarise them better with using the system efficiently.

- Recruit users with prior sketching experience, particularly faces, as the system is primarily aimed at that target group. It will help answer questions such as: Will users with a better perception of facial shapes run into the same problems as the inexperienced users in this evaluation? Will the experienced users produce 3D reconstructions with a lower model error.

- Collect all the strokes sketched by the users to analyse the sketching behaviours better. If the system were to be improved by adapting extreme and crude strokes instead of ignoring them, the stroke data could be used to simulate a second evaluation which could be used to see if the proposed improvements would produce lower model errors.
- Use a survey to measure the subjective quality of the sketched 3D reconstructions. The measured quality could be compared with the model errors to measure the correlation between these two measurement, and also to the results from the paper sketch subjective measurement discussed in the first point.

Chapter 5

Posing faces by sketching

Animating a 3D face model with realistic expressions and facial poses generally requires extensive and skillful manual labour. The aim of facial animation research is to make it quick and easy to accurately pose an arbitrary 3D face model by offering high-level tools. Facial animation has been an active research topic since the work of Parke [Par72; Par74], where he parameterises facial expressions on a specific mesh and is able to create a range of expressions by varying the parameters. Since then, parameterised methods have proved successful. In 1978, Ekman and Friesen [EF78] developed the Facial Action Coding System (FACS) which has been incorporated by numerous researchers [Wat87; KMMtT91; CBK+06].

Blendshapes is the most popular facial animation technique used today. An artist creates key poses which are used to linearly interpolate new poses, where the blend can consist of whole faces or regional blends [PHL+98; LCF00]. Creating the key poses, sometimes called morph targets, needs skillful manual work unless they can be generated using motion data from real people [CB08]. Performance-driven methods where an actor performs the facial actions provide a more automatic and accurate way of generating realistic animations, where the actor's face is generally labelled with a set of markers. Deng and Neumann [ZD07] provide further information by describing a range of different facial animation techniques developed in recent years.

The combination of fusing together the performance-driven approach into an example-based technique is a recent trend in facial animation. Fundamentally, it gathers prior knowledge of facial movements by appling statistical inference on the motion data to achieve accurate reconstructions through maximum likelihood. Example-based sketch interface methods build on the same idea, but introduce an intuitive, high-level approach of controlling the facial poses through sketched strokes. This was discussed in Chapter 2, where they were referred to as template-based systems because they target a particular class of objects, in this case the human face. Section 2.3 details a range of methods which pose a 3D face model using sketched strokes.

This Chapter proposes an animation system (see Figure 5.1), which applies a sketch-based approach to posing 3D face models, and is based on the modelling approach in Chapter 4, but uses a less complex mapping approach. Generated poses form a set of keyframes which are used to create an animation sequence.

In the modelling approach, each training sample was a different face, where here each sample is the same face, but posed to signify different expressions and phonemes. The following sections focus on the implementation details of this approach.

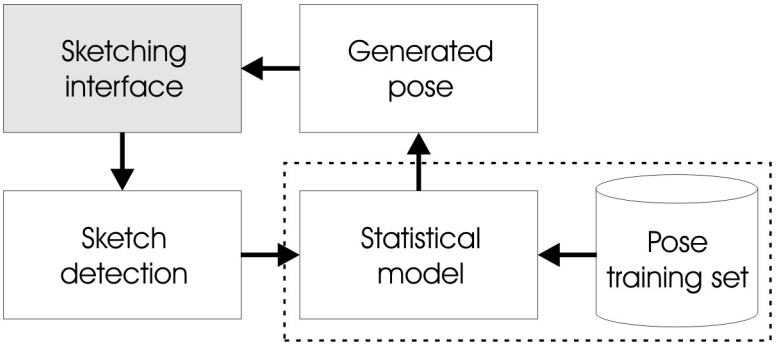


Figure 5.1: Proposed sketch-based animation system.

5.1 Posing models using simple strokes

As with the modelling approach in Chapter 4, the sketch-based approach is made up of an offline part and an online part. The offline part is where face data is collected and processed to form a knowledge-base in the form of a statistical model that can be accessed in real-time by the online part. The online part is an interactive sketching interface that can interact with the statistical model to provide intelligent feedback to any sketched strokes. The stages of the approach are as follows:

1. **Prepare facial poses -** The training data contains 36 poses of a 3D face mesh, each representing a different expression or viseme. Each pose is labelled with 46 FPs giving two sets of corresponding poses, 'mesh poses' and 'FP poses' (Section 5.2).

2. **Interpret sketched strokes from a user -** The user sketches directly on the 3D face as illustrated in Figure 5.2. A probability model is used to identify potential FPs from the sketched strokes (middle) (Section 5.3).

3. **Find best pose -** A generative model uses the the FPs identified in 2 to find the remaining, unidentified FPs in order to make up a complete FP pose (Section 5.4).

4. **Reconstruct mesh pose from FP pose -** The complete FP pose acts as a set of control points used to deform the face mesh into the desired facial pose through a statistical mapping (Figure 5.2, right).

5.2 Preparing training data of facial poses

36 face poses (neutral pose and 35 different expression and visemes poses) are created using FaceGen [1] where each mesh shares the same vertex topology. 46 FPs are placed manually on the neutral pose using a subset of the MPEG-4 facial animation standard [Pak02]. They are then mapped to the nearest vertex on

[1] FaceGen; Singular Inversion

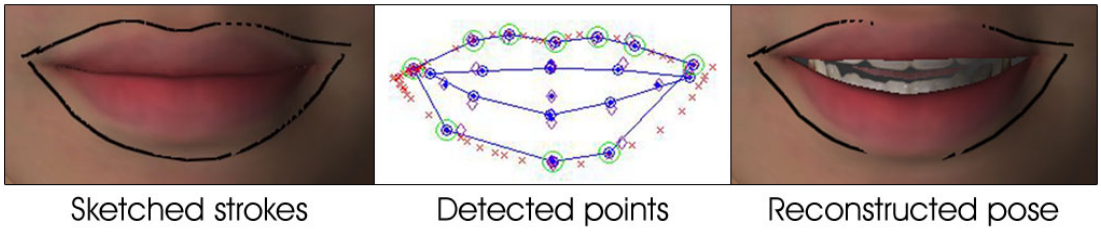


Figure 5.2: The sketching process. Red x's represent the sketched strokes. Green circles are the identified FPs from the stroke (A). Blue circles are the input and expected data making up the complete FP pose (B).

the neutral mesh which is used to automatically calculate the FP coordinates for the remaining poses based on the mapped vertex indices. The anger and open smile pose were used to examine if the FPs map to vertices that best capture the characteristics of the lip contours. Ideally, the FPs should be located at local extremas where the tangent of the contour curve is zero. If the indexed vertex location for an FP did not lie at a local extrema when adapted to a new pose, but had a vertex nearby that did, the FP was assigned to the more descriptive vertex. This was particularly important for the inner lips since they are hard to label using the neutral pose.

40 FPs are used for the main skin mesh shown in Figure 5.3 on the neutral and anger pose where the FPs are displayed as red dots. Figure 5.4 shows the range of possible locations for every FP on the main skin by plotting the FPs from every pose, where each FP forms a cluster which is distinguished with a unique colour tone.

The tongue is labelled using 4 FPs (shown in Figure 5.5 without an opacity map), and the lower teeth contain 2 FPs. The upper teeth are not labelled because they do not move in the training set samples. Figure 5.6 plots the range of motions for these 6 FPs using coloured clusters as before.

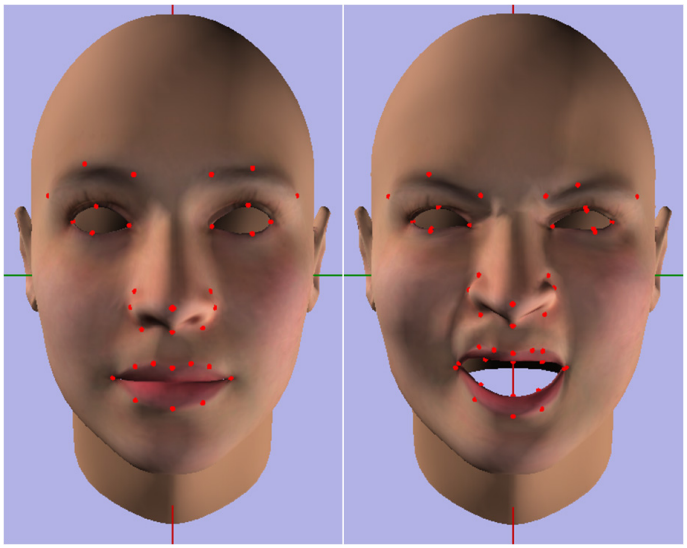


Figure 5.3: Labelled FPs (red) on the neutral pose (left), the anger pose (right).

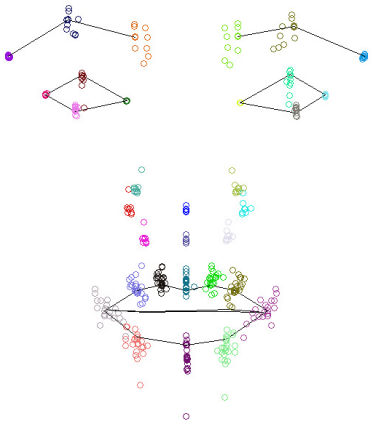


Figure 5.4: FP clusters for all 36 poses of the skin.

5.3 Finding FPs from sketched strokes

Every FP pose can be thought of as a low dimensional representation of its equivalent mesh pose. Figures 5.4 and 5.6 show the FPs for every pose in the training set where points labelling the same feature form a cluster. The clusters are plotted with different colours to visualise the range of motion for each facial feature.

In the sketching interface, the user sketches strokes representing the shape of facial features, e.g. whistling lips, sad eyebrows etc. A sketched stroke is made up of 2D discrete points which are projected onto the 3D mesh to get a depth estimate. The process of interpreting the sketched strokes involves mapping these points to the most suitable FPs. This can be done heuristically by measuring the shortest average euclidean distance from the sketched point to every FP in a particular cluster. FP clusters sometimes intersect because the facial features are moving. This is particularly true for the eyes and eyebrows, and some parts of the lips. A better solution is to define the FP clusters in a likelihood framework where the clusters naturally extend to the soft clustering approach embedded in

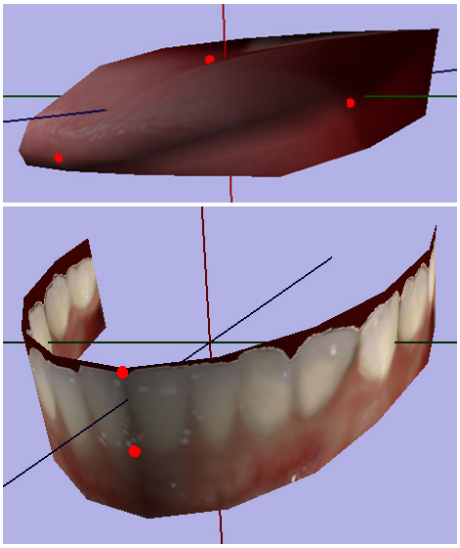


Figure 5.5: Labelled FPs for the tongue (top) and lower teeth (bottom).

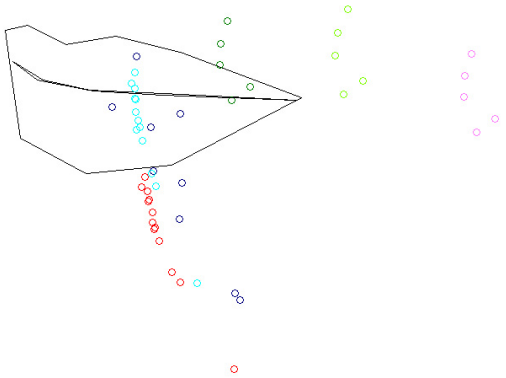


Figure 5.6: FP clusters for all 36 poses of the teeth and tongue (with lip outlines).

the mixture model, and calculate the probability that a sketched point belongs to a particular FP cluster.

This is done by fitting a mixture model on 32 FPs (a subset of the labelled FPs), where each sample is the XYZ-coordinate of a single FP. A subset is used to simplify the classification by omitting unnecessary FPs with regards to sketching. This includes the tongue, teeth and the FPs for the inner lip as they are not needed in most cases to distinguish between different lip poses. Each group of FPs is defined as a cluster which means there are a total of 32 clusters (or mixture components), where the centre for each cluster is initialised as the mean of the corresponding FP coordinates.

When the user sketches a stroke, the marginal likelihood $\mathbf{p}(\mathbf{t_n})$ and the posterior responsibility is calculated for each point in the stroke using Equation 4.8. The responsibilities form a vector with 32 values, where each index represents a single cluster (see Figure 5.7). The values determine the probability of a single stroke point belonging to a particular cluster, where the values range from 0 to 1,

and the sum of responsibilities is 1. The point is assigned to the cluster with the highest posterior probability if its likelihood is above a certain threshold. If more than one stroke point belongs to the same cluster, the point with the highest likelihood is used.

This makes sure only probable points are extracted from the strokes and mapped

Figure 5.7: Posterior responsibility for each cluster (total of 32 clusters) calculated for each sketched point.

to the most suitable FP. The observed FPs can now be used to find the unobserved FPs in order to make up the complete pose (see Section 5.4).

Some complications arise in a sketching interface that should be considered. As mentioned above, the strokes are sketched in 2D which means the values along the depth axis based on the current viewpoint are unknown. The stroke points were projected onto the 3D model to get an estimate for the depth values in order to classify the points as FPs, but these values might not represent a realistic depiction of any known pose. Figure 5.8 demonstrates this problem where a particular lip shape is sketched from the front view and then examined from a higher viewpoint. The sketched strokes lie on the xy-plane, and are projected onto the model to retrieve the z-depth (left). However, as seen by observing the optimal pose in relation to the strokes (right), the ambiguous depth values are not accurately depicting a realistic pose.

To overcome this, the values for the ambiguous depth axis (based on the current viewpoint) are removed, and replaced with the maximum likelihood values using the more reliable axes values representing the sketch plane as conditional data (Equation 4.21). Applying that to the scenario in Figure 5.8, the ambiguous axis is the z-axis, and the reliable axes are formed by the xy-plane since the strokes are sketched from the front viewpoint.

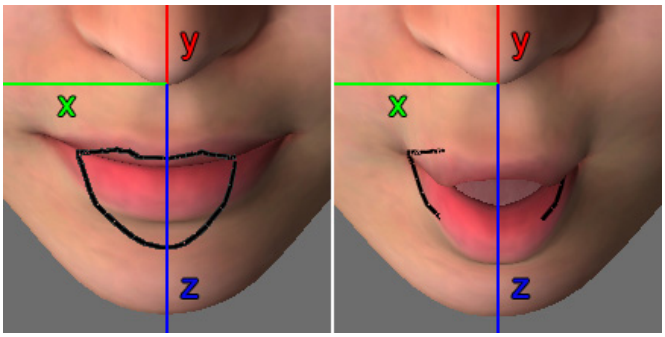


Figure 5.8: Projected depth values for sketched strokes may not represent accurate values of the optimal pose.

Entire strokes are sometimes ambiguous where they can be interpreted as more than one facial feature. An example of this is a sketched line that can either be an inner lower lip, or an outer lower lip. A sketched eyebrow can be classified as the upper eyelid.

The modelling approach in Chapter 4 uses contours which only display features relevant to the current viewpoint at any time, and the contours generate a considerably more dense representation of the facial features compared to the FPs. Mapping stroke points to a the dense contours therefore provides more redundant-free data to base the mapping on, and one incorrect assignment does not compromise the mapped feature. In contrast, incorrectly assigning a single FP fails to produce an accurate feature.

Instead, this is managed here by assigning the FPs to different groups such as left eye and upper lip. Strokes are assigned to a unique group by examining which group it is assigned to most often. Any point in the stroke can then only map to FPs within the assigned group.

Strokes containing points which do not conform to the training set can also cause problems. For instance when sketching an upper lip, the endpoints may not lie near a cluster likelihood range and will therefore be discarded. The endpoints carry important information as they define the boundaries of the desired feature. An example of this is if the stroke defining the upper lip is too short. If as a result

the endpoints are ignored, the width of the lips will remain unchanged, while it is fairly likely that the user intended for them to become shorter. It is also possible that an endpoint has a lower likelihood than a neighbouring point on the stroke which maps to the same FP (lip corner).

Two methods are used to increase the chances of correctly identifying the boundaries of a stroke:

- The stroke endpoints are given higher weights to make sure they have an advantage over other points in the stroke.
- If a stroke's endpoint does not map to any FP, a test is performed to see if they would be classified correctly if the user would have sketched them slightly differently. The endpoint is moved towards nearby FP clusters defining a feature boundary (e.g. a lip corner), and will be mapped to an FP if the endpoint's translated coordinates fall within the FP's likelihood range.

5.4 Reconstructing a complete pose

At this stage the system has identified sketched points mapping to a number of FPs from the set of 46 FPs describing a single facial pose. There are typically a large number of unidentified FPs as the user is not expected to draw every aspect of the pose. In addition to that there are 14 FPs that cannot be sketched here (teeth, tongue and inner lips) which means the FP pose is always incomplete. A generative probability model is needed to find the most likely pose given a partial pose defined by the FPs that were identified from the sketched strokes.

A Gaussian mixture model (Section 4.2.1.3) is fitted on a training set consisting of $n = 36$ different poses, where each pose contains 46 3-dimensional FPs. A single training sample is therefore a vector with $46 * 3$ dimensions, making up a training set of size 36×138. The FPs identified from a set of sketched strokes are used as observed data where the max-conditional distribution is calculated over the missing points to construct a complete FP pose (Section 4.2.1.6).

The problem is finding the number of mixture components that best fit the model for the purposes of sketching. Two mixture models with a similar likelihood, but different number of mixture components interpret the data differently when estimating missing data.

The criteria for finding the most suitable number of components is defined based on both the overall likelihood of the model, and what a user might expect the system to produce given some minimum number of strokes describing a particular facial feature.

The minimum number of FPs needed to characterise each target pose are specified and stored as incomplete pose vectors $p_i \epsilon P, i = 1..n$, where the known coordinates are extracted directly from the target pose. For instance, three FPs depicting a lowered eyebrow are expected to produce an angry pose.

Finding the best mixture model involves fitting m mixture models on the training data and examine how well it reconstructs full poses using the incomplete pose vectors P. The procedure is as follows:

1. Fit a mixture model M_j on the training data using $j = 1..m$ components.
2. For M_j, create a reconstruction r_i of every incomplete pose $p_i \epsilon P$ by finding the missing FPs, and calculate the error $e_{ij} = \|r_i - p_i\|$.
3. Calculate the total error $\xi_j = \sum_{i=1}^{n} e_{ij}$ for M_j.

The error values $\xi_j, j = 1..m$ indicate how well each model managed to generate the range of target poses from P, so picking the model with the lowest error should be straightforward. However, the models have slightly different interpretations when estimating which feature is created based on the incomplete poses. Lets consider a scenario where a user sketches a wide upper lip as shown in Figure 5.9. Arguably, the most likely intention here is to create a smile, but with this limited information it could be interpreted either as an open smile or a closed smile. The best mixture model is therefore one with a high likelihood and a low error value, while at the same time interprets individual poses based on incomplete and ambiguous input data according to human expectations.

Figure 5.9: Indicating a smile with a sketched stroke.

The most likely pose generated by a model that is chosen is not necessarily the interpretation every user is expecting. Figure 5.10 shows two possible interpretations for the same stroke. A straightforward way to remove the ambiguity is to add more strokes, but as an alternative, the system offers the possibility to choose between interpretations from two different mixture models. Here the first mixture model assumes the user is trying to sketch an open smile, while the second one finds the most likely option to be a more closed smile. The user is forced to add more strokes to clarify his intentions if neither model produces what he is after.

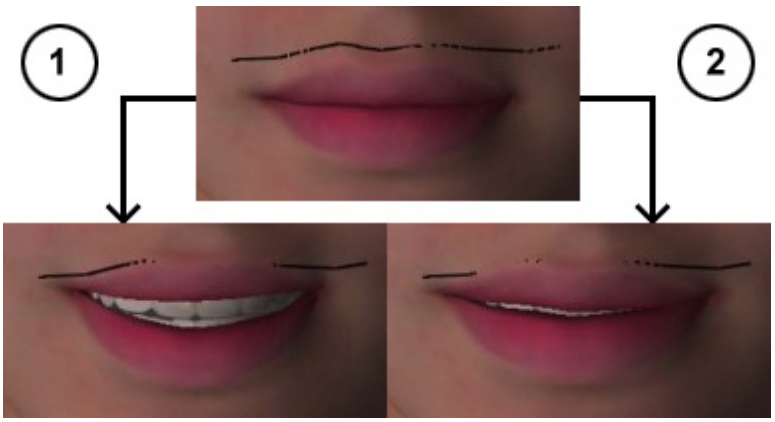


Figure 5.10: Different interpretations made by two mixture models. The interface offers the user the option to switch between them.

Another example of this is shown in Figure 5.11. There is a correlation between how far down an eyebrow is sketched, and how angry the face will get which is again reflected when generating intermediate frames. However, a face

can have slightly lowered eyebrows without necessarily being very angry. Again the interface offers the user to choose from two interpretations where the latter one decides the face should look more annoyed than angry.

Figure 5.11: Different interpretations made by two mixture models. The interface offers the user the option to switch between them.

The following examples show how well the selected generative model estimates a complete pose based on incomplete input data for a particular mixture model. Figure 5.12 shows the reconstructed pose when using three observed FPs (green) describing the left eyebrow taken from the anger pose (red x's). The blue circles represent the reconstructed pose which lies very close to the target pose despite the small amount of observed data. It is found that most mixture models interpret this the same way and with accuracy which indicates that the eyebrows are rarely shaped this way in the other poses, and therefore this shape is enough to characterise the entire pose. Figure 5.13 shows a similar scenario for an open smile, but now only one observed FP is used describing the right lip corner. The model is able to predict accurately that the intention is to create an open smile.

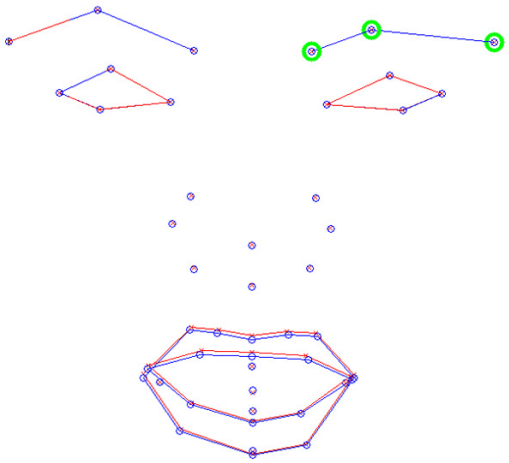


Figure 5.12: Reconstructing anger pose from incomplete data. FPs circled in green are the observed data. Red x's represent the target pose. Blue circles show the expected data.

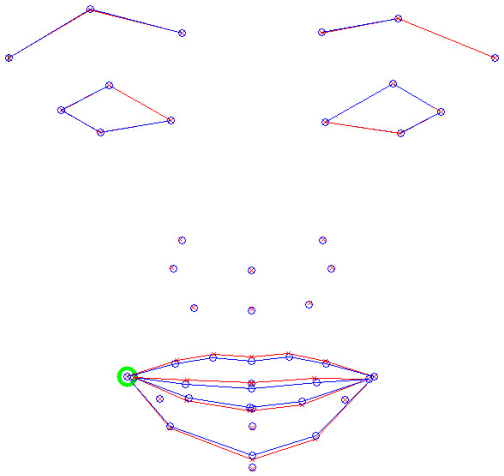


Figure 5.13: Reconstructing open smile pose from incomplete data. FPs circled in green are the observed data. Red x's represent the target pose. Blue circles show the expected data.

5.4.1 Reconstructing mesh pose from FP pose

The FPs act as control points which are used to deform the mesh vertices in order to create a range of different facial poses. The set of FP poses are referred to as P, where $p_i \, \epsilon \, P, i = 1..n$ is a single FP pose, and similarly the set of mesh poses as V, where $v_i \, \epsilon \, V$ is a single mesh pose. A statistical mapping $\Psi : P \rightarrow V$ is defined which maps a d_P-dimensional FP vector to a d_V-dimensional vertex vector.

The problem is how to map a low-dimensional set of FPs to a high-dimensional mesh. A statistical mapping method is proposed here and contrasted with a popular approach of using Radial Basis Functions. Furthermore, this statistical method outperforms different variations of the Radial Basis Function technique, artificial neural networks, point deformers, and planar bones, for the purposes of facial animation [1].

[1] Personal communication with Martinez, 2009

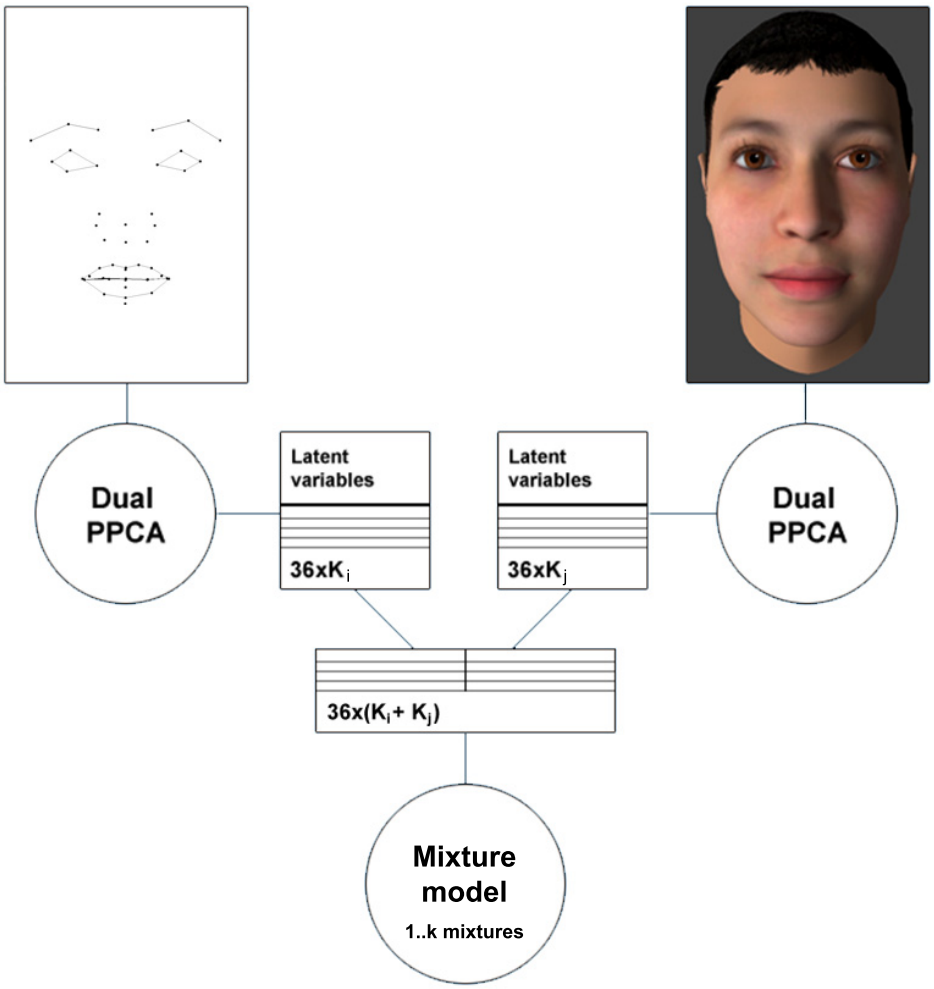


Figure 5.14: Mapping FP pose to a mesh pose.

Figure 5.14 visualises the overall mapping process where the left side describes the FPs, and the right side describes the mesh vertices. Because $d_V \gg n$, Dual PPCA is performed on both P (left) and V (right), where the latent variables for each sample are calculated to form $n \times q_P$ and $n \times q_V$ latent matrices using Equation 4.20. The two latent matrices are joined together to form a $n \times (2 \times (q_P + q_V))$ matrix which is the mapping matrix. A series of mixture models are trained on the mapping matrix k mixture components.

A mesh pose v_i is mapped from a FP pose p_i using the following procedure. Consider the reconstructed pose p_i (blue) in Figure 5.15 based on sketched points (red). The latent variables for p_i are calculated and used to fill q_P dimensions in a single mapping sample. The remaining q_V latent variables describing v_i are labelled as missing.

The expected latent variables for v_i are calculated and used with the corresponding Dual PPCA model to reconstruct a full vertex pose using Equation 4.20 (see Figure 5.16). The current mesh vertices are overwritten with the new reconstructed coordinates which updates the face model with the new pose. The user can now continue sketching to make further adjustments, or to use the current pose as a keyframe in an animation sequence (see Section 5.5).

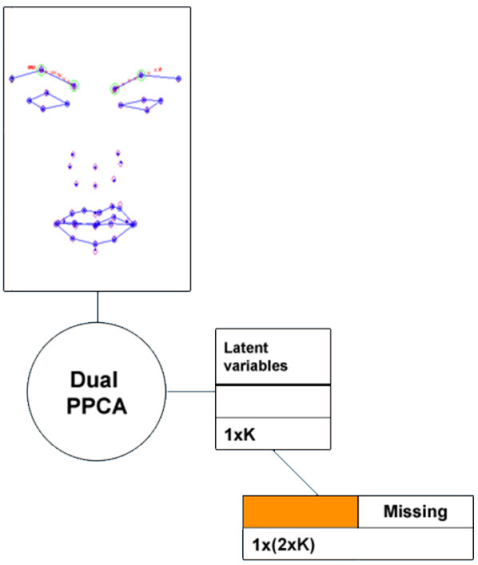


Figure 5.15: Mapping FP pose to a mesh pose.

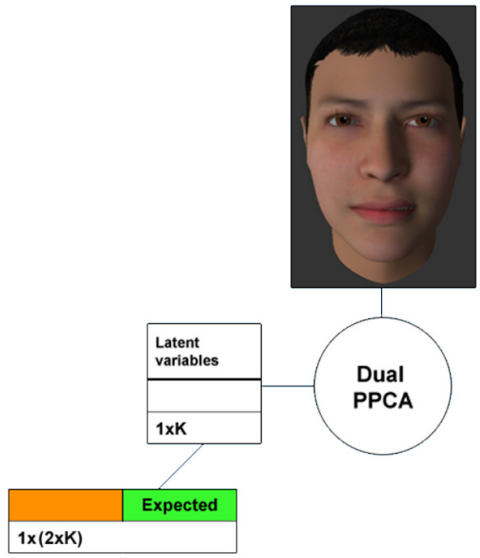


Figure 5.16: Mapping FP pose to a mesh pose.

To verify this method is capable of producing the correct mesh pose structure from only 46 feature points, the 36 target mesh poses (V) are mapped from the corresponding set of feature points (P). Figures 5.17 and 5.18 show the reconstruction of two facial poses used throughout this Chapter, the anger pose and the

open smile respectively. The upper left corner shows the FPs extracted from the target pose, and the bottom left and right show the mesh reconstruction. The red dots represent the original target vertices for the given pose, and the blue circles display the reconstructed vertices. A reconstruction for a particular vertex has zero error if the red dot lies perfectly within the centre of its corresponding circle.

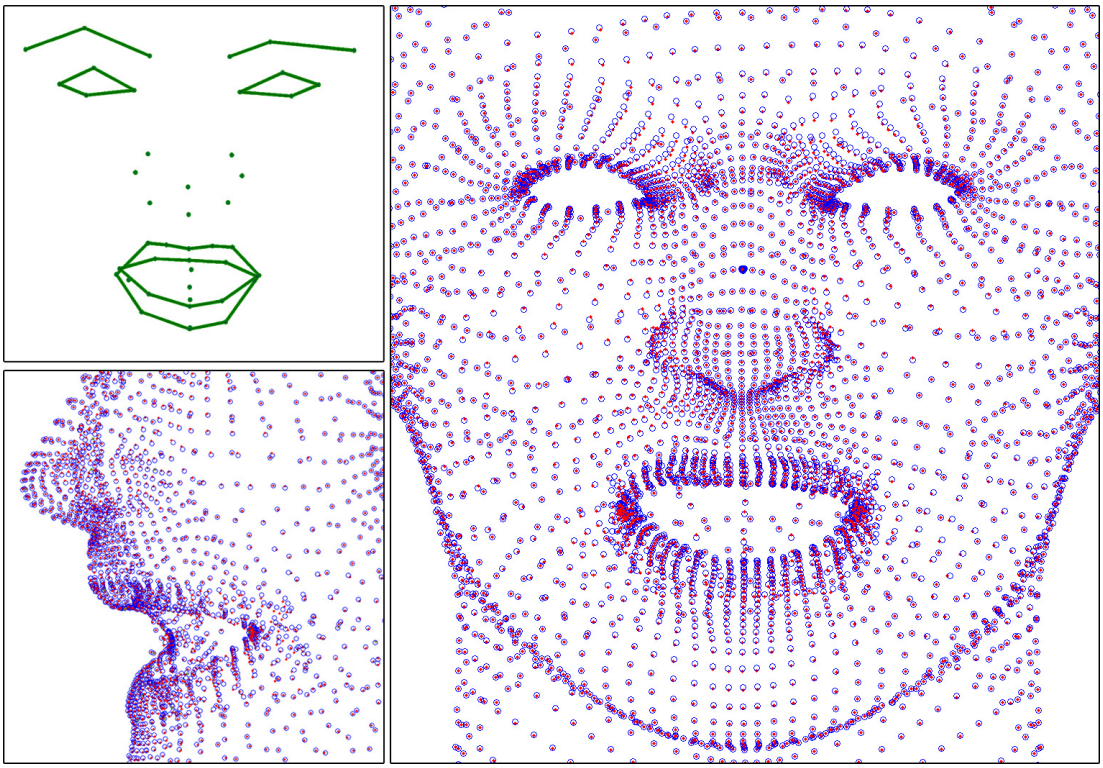


Figure 5.17: Anger pose. Reconstruction using the marked feature points for the pose. Blue circles show the reconstructed mesh, and the red dots specify the target mesh for this pose.

Radial Basis Functions is a popular mesh deformation technique and can define a mapping between P and V using mapping coefficients where each FP is defined as a single Radial Basis Function. The coefficients are used to deform a source model into a new pose (see Chapter 3).

An issue with this method is that the source model has to be chosen carefully to reconstruct the pose correctly. For instance, if the target pose has an open

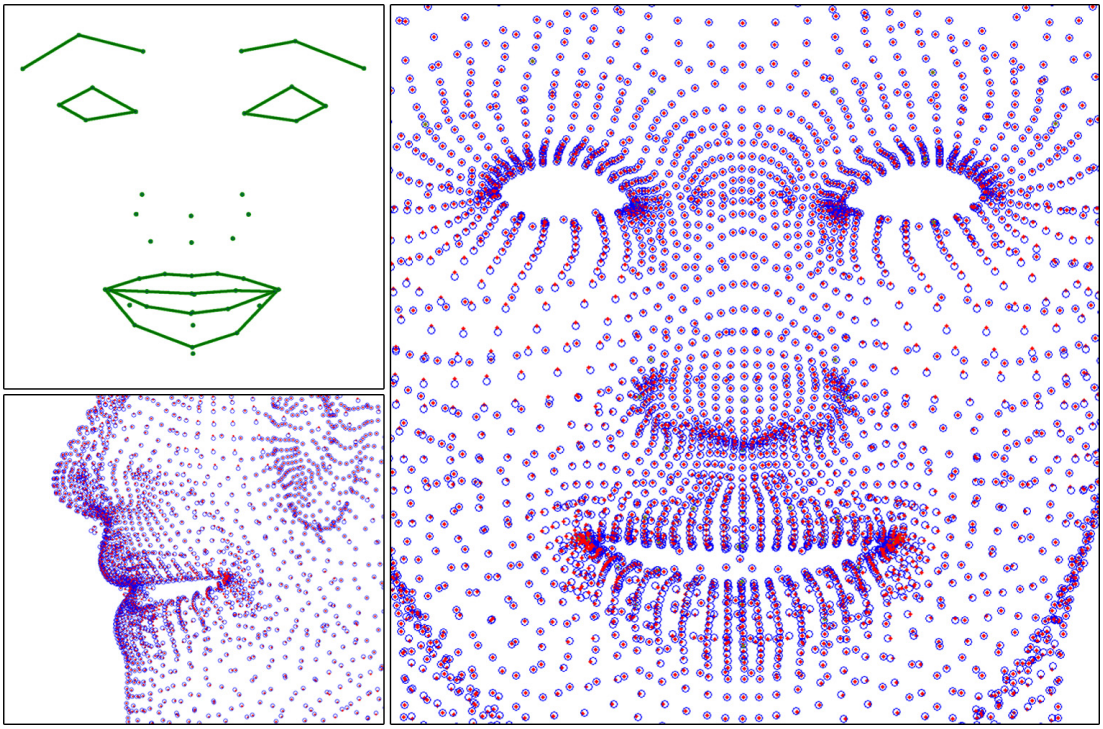


Figure 5.18: Open smile pose. Reconstruction using the marked feature points for the pose. Blue circles show the reconstructed mesh, and the red dots specify the target mesh for this pose.

mouth, a deformed source model where the mouth is closed will not produce an open mouth. Another problems with the RBF approach is that its effect is too global, lacking local details. Figure 5.19 compares the statistical mapping and the RBFs mapping where the target is the anger pose. The open smile is used as the source in the RBFs process to facilitate an open mouth.

The next section shows how using the statistical mapping, a whole range of facial expressions can be generated using only 36 target poses as a training set. The system is capable of reconstructing every target mesh pose, as well as a gradient of poses not present in the training set.

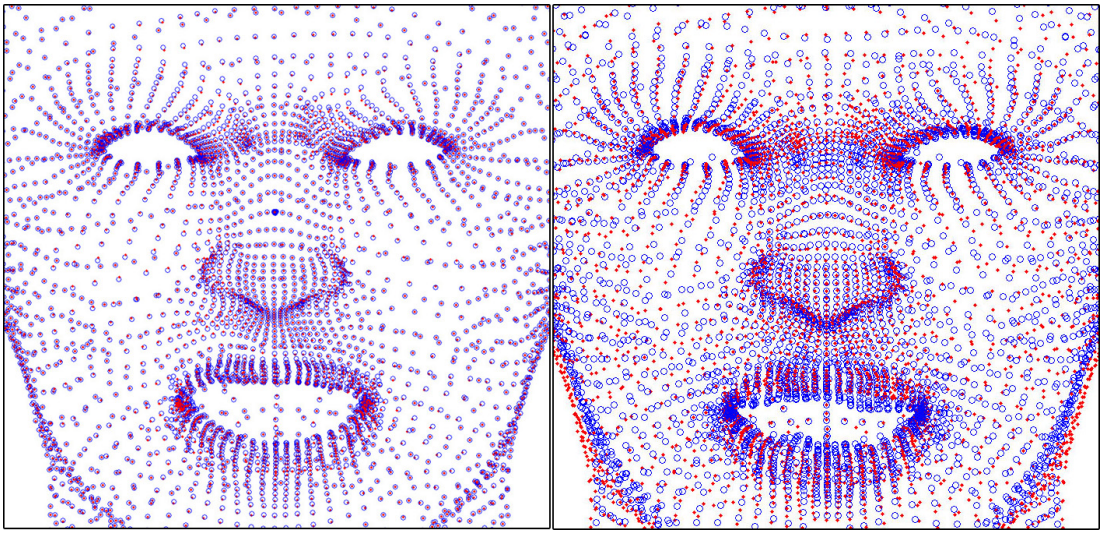


Figure 5.19: Comparison between our deformation method (left) and RBF deformation (right).

5.5 Creating an animation using keyframe poses

As a starting point, the user is presented with the neutral pose where he can sketch directly on the model from any viewpoint. When the user is satisfied the system identifies observed FPs from the sketched points using the method described in Section 5.3. The expected values for the remaining FPs are calculated (Section 5.4), and used to recover an updated vertex structure (Section 5.4.1). The user can continue sketching to make further adjustments until he is satisfied with the pose and adds it into a sequence of keyframes. Figure 5.20 shows some keyframes produced with some simple strokes starting with the neutral pose in the top left corner.

Intermediate frames are rendered to make up a complete animated sequence. This could be done by linearly interpolating between two keyframe models, but instead the keyframe FPs are interpolated using a cardinal spline and the generative model is used to reconstruct the model for each frame. This is done to prove the system can generate a gradient of facial poses which are accurate enough to generate a smooth motion for every facial feature. Figure 5.21 shows

Figure 5.20: Sketching keyframes.

two keyframes and 10 intermediate frames producing an animated sequence using the interpolated FPs as input for each frame.

5.6 Summary

This chapter presented a new approach to creating 3D facial animation through sketching. Sketching acts as a high-level control to modelling where a new pose can be created by indicating the desired outcome as opposed to applying animation targets, moving individual control points, or tweaking semantic parameters. Very few sketch strokes are needed to construct a new pose through incomplete data handling. This is accomplished using a knowledge-base in the form of a statistical model that through a maximum likelihood approach knows how poses are constructed from partial input. The input is made up of feature points describing each pose in a low-dimensional space. Modifying one aspect of the face

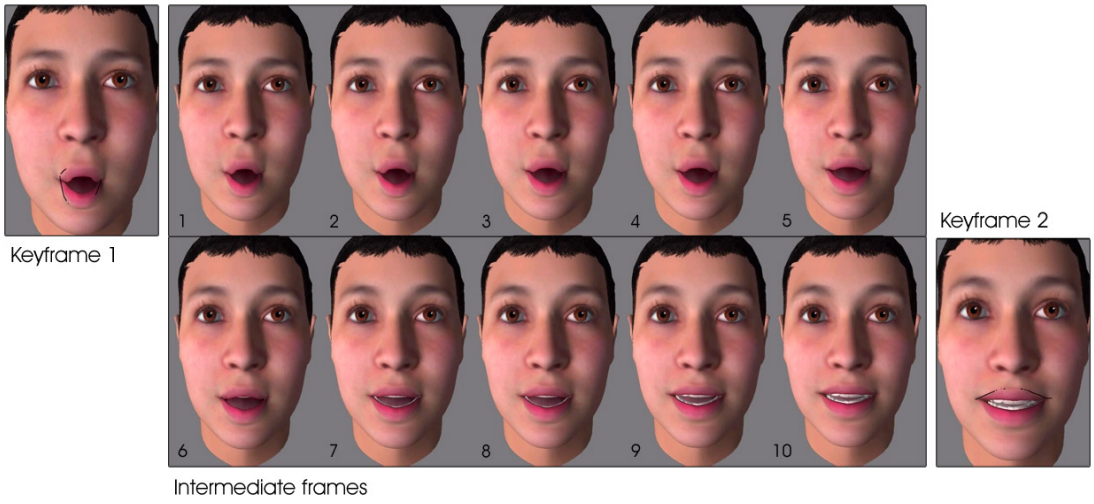


Figure 5.21: Two keyframes and 10 generated intermediate frames. The keyframes are taken from a video on the attached DVD.

automatically correlates other areas on the face to match accordingly. Facial expressions generated using this system are therefore always complete and plausible.

The system is able to create a large range of facial poses using only 36 poses as training data, and accurately calculating intermediate frames. With a more extensive data set the results could be further improved along with the addition of new poses that fall outside the likelihood range of the current system. The system is limited to making changes in areas having pre-defined FPs, but the current set of FPs classifies the main facial features where the statistical mapping makes sure the unlabelled areas are adapted to match the desired pose (e.g. cheekbones when the lips move).

Chapter 6

Conclusions

This thesis has presented a sketch-based method for modelling 3D faces. A generative model, trained on the vertex structures from a set of standardised faces, is capable of creating a range of novel faces by inferring missing vertices from observed vertices through maximum likelihood to create a complete vertex structure. This method was evaluated by asking novice users to perform two tasks which involved sketching a model with specific facial features. The users were given a verbal description of the target features in task 1, and a picture of the target from two viewpoints in task 2. The users were then asked to fill out a questionnaire where they had the chance to score the sketching interface and their experience. The results indicate that even for novice users, the sketching interface is easy to use and lets them create features they are generally happy with. However, the interface is still primarily a research tool which was reflected by some issues the users had.

Chang and Jenkins [CJ06], and Lau et al [LCXS07] sketch a reference stroke first to specify what feature is affected, and then a target stroke specifying the new location, when posing a face mesh. In contrast, the sketching interface implemented as part of this thesis follows a what-you-sketch-is-what-you-get (WYSIWYG) approach, where every stroke is treated as a new shape of a facial feature, and the system has to determine which feature is being sketched. To assign a feature specific meaning to the strokes, the stroke points are mapped to target vertices in the training set. However, it is important to limit the number of possible mappings since a vertex cloud consisting of every vertex in the training set is

very dense. A Non-Photorealistic Rendering (NPR) technique is employed which specifies which vertices are the most important from a sketching perspective. These vertices are called contours, and applying this to every mesh generates a contour cloud referred to as contour candidates. The advantage of mapping the strokes to the NPR data is that it minimises redundant and ambiguous mapping targets which improves accuracy and efficiency. An important issue here is that these contours cannot serve as a low dimensional parameter space because no two viewpoints produce the same contours for a single mesh, and no two meshes have the same contours from the same viewpoint. Therefore every vertex in the standardised mesh could potentially describe a feature for a particular mesh, from a certain viewpoint. As a result, every vertex has to be included in the training set used to fit the generative model. The vertex structure is segmented into several regions due to the large number of vertices in the face model where the vertices in each region become a separate training set. Having facial features split between different regions makes it easier to perform local changes to a feature without affecting the remaining structure.

Initially, any combination of features is equally likely since no information has been provided which leads to the assumption that the mean of the training set is the most suitable starting point in the sketching interface. Limiting the solution space of features is achieved by providing observed vertices which forms the conditional data used to determine the remaining vertices. In the sketching interface, the observed vertices are found by mapping sketched strokes to contours. Sketching can therefore be seen as applying restrictions whereby adding enough strokes can restrict the generative model to produce a particular individual.

It is justified that contours, acting as a low dimensional representation of a complete mesh, can be used to reconstruct the original mesh where the 3-dimensional shape of every facial feature is preserved. This was done both objectively and subjectively. The objective approach measured the reconstruction error, found by measuring the total vertex distance between the original model and the reconstruction. The subjective approach involved an online user experiment where 122 entries were valid. The users were asked to identify 10 faces where each face was surrounded by 8 faces to choose from. One of the choices was a reconstructed face model using the contours extracted from the original

faces (assuming an unknown vertex structure). The mean score was $\mu_X = 0.6721$ with a standard deviation $\sigma = 0.2117$. This translates to the users picking a reconstructed face over other choices 67.21% of time since there were 10 recognitions in total. In contrast, there is a 12.5% chance of picking the reconstructed face by clicking a choice randomly.

The objective and subjective measures support the idea of using contour data to reconstruct a complete face model. The key question concerns how the sketched strokes can be mapped to the view-dependent contour cloud in order to reconstruct features that accurately fit the strokes and does not compromise the vertex structure. Two methods were developed to tackle this problem, a heuristic approach, and a statistical approach which uses Hidden Markov Models (HMMs) based on the heuristic metrics.

The heuristic approach traverses the strokes and calculates a grade for each stroke point, where the grade rates the mapping potential from the point to nearby contours. The grading function uses metrics which take into account the transition from the previously assigned contour, where transitions that compromise the vertex structure or do not follow the stroke's shape are penalised. A key element in fitting accurate features to the strokes is finding the best starting point which establishes the context for the stroke. It is important here to make some assumptions to reduce ambiguity as a stroke can possess different meanings (see Section 4.6.2). Setting an incorrect context resonates through the grading process by mapping the stroke points to misleading contours. The grading scheme is simple and yet produces impressive results. The grading function however is sensitive to changes made to its blending weights and parameters. It requires a fair amount of parameter adjustment in order to find a good balance which is stable enough to generate most features accurately. Minor enhancements can be achieved by using different parameters for specific contour types but adjusting a large number of parameters is not an elegant solution and some features are made up of different types of contours.

Instead, the grading function is replaced with a joint probability distribution using Hidden Markov Models (HMMs) which offers a more principled approach to the mapping process. HMMs find the most probable sequence of hidden states for a given observation sequence which makes them suitable to classify a sequence

of sketched points [SD05; KSvdP09]. The metrics used in the heuristic grading scheme are reformulated as probability functions, as they provide the information needed to set the initial context, and measure the transition between two consecutive stroke and contour points. Additionally, it includes an improved vertex connectivity metric which uses Dijkstra's algorithm to calculate the path length between two vertices where it affects the probability of moving from one contour to another. The stroke points represent the observed data, and the contour points are the hidden latent states in a HMMs framework. The Viterbi algorithm [Vit67] is then used to find the hidden states by traversing the trellis diagram.

The HMMs approach gives far more reliable and accurate results. The heuristic algorithm only considers the last assigned contour when traversing the stroke which makes it somewhat similar to a simple first order Markov chain. In contrast, the HMMs approach however does not put a limit on the order and performs back-tracking. It provides an elegant solution to mapping sketched strokes to a vertex-trained generative model using inverse Non-Photorealistic Rendering data as a bridge. Both the heuristic approach and the HMMs approach offer the possibility of correlating unsketched features. This can be used to look faces up in a database based on limited data. One potential use for this is to find a suspect in a criminal database based on an incomplete witness description.

Although the contours can be used to create a range of novel features, they are still a low dimensional representation of the complete data. The areas between the observed contours are filled using the most plausible shapes based on the training data. However, there is no guarantee that this gives the intended surface curvature between the strokes. It was noticable when sketching the test models, that features fitting the strokes well did not necessarily relate to a good likeness if the curvature between the sketched features differed from the source model. This could be alleviated to some degree by using more detailed training samples where the contour data would be more dense. One option is to use the contour data to create the main features and some details, and then apply a geometric brush tool approach such as the one offered in Z-brush[1] and Autodesk Mudbox [2] as a post process to modify the surface and create further details.

[1] http://www.pixologic.com
[2] http://www.autodesk.com

Furthermore, the overall likeness can be affected if a single sketched feature is not reconstructed accurately, despite all the other features being fairly accurate. This is because one inaccurate feature will affect the relative shape and spatial configuration between the facial features which is believed to be the way humans encode and store faces [SBOR07; dHHR07].

Using discrete control points to create faces gives limited control over the modelling process in terms of possible shapes and characteristic details. Adding control points would allow a larger range of features with more details but it is infeasible to manipulate a large number of control points. A sketching interface based on few control points will only map a few stroke points to the control points and ignores the information provided by the remaining segments on the stroke.

Sucontphunt et al [SDN09] offers a modelling interface where the user can create different faces by pulling eight facial contour curves on a template 3D model, made up of 35 anthropometric landmarks. While easy to use, the level of control is limited to the discrete landmarks and it only operates from the front viewpoint. Individual vertices can be manipulated where the surrounding vertices are adapted by projecting the edited mesh onto PCA nodes defining the face space for different model segments. Each vertex is therefore acting as a control point which offers greater control but is not a desirable solution.

In contrast, the technique introduced in this thesis allows sketching from any angle using contour points based on the surface curvature which adapts to the viewpoint. The contours act as a non-redundant set of control points, where they provide a dense correspondence to only relevant parts of available facial features.

Creating face models using a parameter based approach such as FaceGen[1] or [ABHS06] is very easy as they offer great flexibility and control. However, transferring a specific shape of a facial feature from one's mind by adjusting parameters such as concave/convex cheekbones or round/gaunt cheeks might take several attempts of adjusting before finding the right combination of parameters needed to achieve that shape. By communicating with the modelling system using sketched strokes, this shape can be directly transferred to the system which fits the necessary features to fit the stroke. This offers a more direct approach

[1]FaceGen. Singular Inversion

where the user does not have to specify the steps taken to achieve the reconstructed shape, such as the features involved or their parameter values.

The training set used for most experiments was made up of low resolution FaceGen models. The models are noise free and already standardised which is useful when developing the core architecture for a sketch-based modelling approach, as they give reliable reconstruction results. However, the models fail to capture a number of important details such as wrinkles, dimples, and sudden changes in the curvature due to saggy skin or prominent bone structure. Using detailed high dimensional data requires a large amount of data to be stored in the memory at runtime, and a large number of segments are needed under the current method which causes blending issues.

Animation sequences are traditionally generated from keyframes where the intermediate frames are created by interpolating the key poses. A sketch-based approach for posing faces was introduced based on the modelling approach, but applied a simpler statistical mapping scheme. The training set consists of 36 high resolution poses for a single face model, where the features in each pose are labelled with MPEG-4 landmarks called Feature Points (FPs). Unsketched features are correlated which speeds up the sketching process and produces natural poses. For example, sketching a lowered eyebrow will make the eyes smaller, and produce an angry mouth shape showing some teeth.

The landmarks deform the mesh using a novel statistical mapping. It was shown how each pose can serve as a keyframe where a complete animation sequence is generated by fitting a cardinal spline through the keyframes. Intermediate frames were created by interpolating the landmarks and then reconstructing the full mesh using the statistical mapping technique, as opposed to generating the full mesh for each keyframe and then interpolating between the meshes. This was done to evaluate the mapping technique. It consistently produces a smooth and continuous sequence of intermediate frames which verifies its ability to reconstruct an accurate mesh pose from landmarks.

The animation sequence created using the sketched strokes can easily be transferred to another model given the FPs are known for the target model's poses.

The mapping technique used in Section 5.4.1 can be adapted to establish a statistical relation between the two models, and maps the key FP poses from the source model to an equivalent FP pose on the target model. Then the full model is retrieved as before using Section 5.4.1.

There are a number of possible improvements that could be investigated as future work. Several improvements were proposed in Section 4.11.2 as a result of observing the users complete the two tasks given in the evaluation process. The improvements fall into two categories, where the first one is concerned with making sure the users' intentions are interpreted correctly when the sketched strokes are inaccurate or exaggerated. Processing extreme strokes might have undesirable implications in which case the interface could allow to user to set how strict the interpretations are. This is something that could be examined further. The second category discussed how the evaluation process itself could be improved as well as exploring ways of extending the evaluation by recruiting sketch artists and designers to perform the same task, and furthermore to subjectively evaluate the accuracy of the results from this evaluation. The contour generation can be improved by including methods used in more recent NPR work [DR07; JDA07] which would allow more features to be sketched, and with more accuracy. The effect would be even more dramatic if detailed high resolution meshes were used as the primary data, replacing the smooth low resolution faces. A combination of FaceGen meshes, and standardised laser scans from the USF HumanID 3-D Database [BV99] would provide a good variance of clean and customisable facial features (using FaceGen conformation parameters), and detailed features and characteristics that cannot be produced with FaceGen. The surface curvature and view direction is used to generate a sketchable subset of the mesh vertices referred to as suggestive contours. A way of exploring how more details can be sketched is to use the curvature data to produce a cross-hatch representation of the mesh instead of the contour representation. This simulates shading based sketching, which combined with the contour based sketching could be used to quickly sketch the main features using simple strokes, and then add details by sketching cross-hatches. Instead of defining and adjusting the parameters for the HMMs probability functions manually, they could be learned automatically by

observing a number of users sketch strokes to fit predefined features. This could be taken even further by facilitating a user specific sketch detection behaviour where the system automatically adapts to each user or alternatively invites him to run a calibration process. In addition, other statistical models could be considered such as Generative Topographic Mapping (GTM) [Bis07] and Gaussian Process Latent Variable Models (GP-LVM) which non-linearises the linear mappings from the embedded space of a Dual PPCA model [Law04].

The animation work presented here was based on an earlier mapping approach which has been improved considerably in the modelling approach by employing view-dependent contours and a HMMs. A natural progression is to incorporate this new approach into the animation which would offer better accuracy and enhanced sketchability from every viewpoint. The cross-hatching method could also be incorporated in the animation approach to adjust minor areas around the main features to create character specific details.

A long term goal is to generalise the sketch-based approach in order to make it possible to sketch different classes of 3D object such as caricatured faces, cartoon faces, and the human body. Using a training set of cartoon characters could be used to quickly generate key poses for an animated cartoon [SBr05]. It is also worth exploring whether similar classes of objects can be standardised using the same generic mesh, and combined in one training set with the goal of creating hybrid objects. An example of this could be to combine human faces with caricature faces. Another possibility is to offer a mode-free sketching interface which supports a large number of different object classes but each containing a stand-alone training set and corresponding generative model, where an object recognition process would classify the first strokes in a new sketch and initialise the appropriate generative model. Implementing a sketch-based texture enhancements would provide an intuitive way of adding a number of facial details such as beard and moles [LCODL08].

Appendices

Appendix A

Polygon File Formats

The laser scanned face models in the USF HumanID 3-D Database [BV99] are stored in a binary Inventor file format and are processed with a program called *ivfix*, written by James Ward [1] which converts them into an ASCII Inventor format.

The models are exported to the Polygon File Format (ply) which is a simpler, and more efficient format.

A.1 Inventor (iv) V2.1 ASCII file format

The Open Inventor file format [2] is an object based format which allows extensive description of a whole 3D scene with grouping, cameras, lights, materials etc. It requires the header

```
#Inventor V2.1 ascii
```

The main elements are then as follows (see code below). The objects are specified with a Separator. The object in our case is constructed using a triangle strip set. It specifies the following properties for each vertex, XYZ vertex coordinates (*vertex*), XYZ normal coordinates (*normal*) and UV texture coordinates (*texCoord*). Then it lists the triangles (faces) making up the triangle strip (*coordIndex*),

[1] http://www2.dcs.hull.ac.uk/simvis/personnel/jww.htm
[2] http://oss.sgi.com/projects/inventor/

referencing the corresponding vertex numbers in the vertex list where the end of each triangle is indicated with -1. The same pattern is used for the triangle (face) normals (*normalIndex*), only now referencing the numbers in the normal list.

```
Separator {
   IndexedTriangleStripSet {
      VertexProperty {
         vertex [ 533.22 -4323.24 6486.28,
         ...
         ]
         normal [ -0.00739541 -0.999645 0.025593,
         ...
         ]
         texCoord [ 0.77 0.326,
         ...
         ]
      }
      coordIndex [ 84989, 84991, 84862, -1,
      ...
      ]
      normalIndex [ 263286, 263290, 262947, -1,
      ...
      ]
   }
}
```

A.2 Polygon (ply) file format

The Polygon file format, also known as Stanford Triangle format, was developed at Stanford University to store 3-dimensional data from 3D scanners [1], and can

[1] http://www.graphics.stanford.edu/data/3Dscanrep/

only store a single object. A typical structure of a ply file is:

Header
Vertex List
Face List

It supports few other elements but custom made elements are required to specify more complex properties such as texture coordinates. The header specifies which elements are expected and what properties they possess. A typical polygon object with 5590 vertices and 10999 faces has the following header:

```
ply
format ascii 1.0
element vertex 5590
property float x
property float y
property float z
element face 10999
property list uchar int vertex_index
end_header
```

It states that each vertex has XYZ coordinates and the faces are represented as a list of integers of an arbitrary length that index the vertices.
The following elements are added to handle the UV coordinates and texture name:

```
element texture_coordinate 5590
property float u
property float v
element texture_name 1
```

An example of a ply file with the complete header is then as follows:

```
ply
```

```
format ascii 1.0
element vertex 5590
property float x
property float y
property float z
element face 10999
property list uchar int vertex_index
element texture_coordinate 5590
property float u
property float v
element texture_name 1
end_header
-6000.533984 1285.712925 6294.708409
...
3 593 594 591
...
0.520369 0.545208
...
models/images/Face1.bmp
```

Note that the second sequence containing the face list states with the first number how many vertices make up the face and then lists the vertex indices.

Appendix B

Calling Matlab from C++

Matlab offers powerful matrix calculations and methods which can only be accessed in C++ by utilising a number of external modules. In C++ however, it is possible to use a Matlab engine to perform complex calculations by using the following two steps:

1. The first thing that needs to be done is to include the header and library files to instantiate and communicate with the Matlab engine.

`#include <engine.h>` (located in $Matlab$\extern\include)

These library files should be included: `libeng.lib, libmat.lib` and `libmx.lib` (located in $Matlab$\extern\lib\win32\microsoft\msvc60\).

2. To open and close the Matlab engine, use the following code:

```
Engine *matlabEngine;
matlabEngine = engOpen(NULL);
if(matlabEngine == NULL)
{
    // Error!  Failed to connect to MATLAB engine
}
```

```
// Some code..

// Close Matlab engine
engClose(matlabEngine);
```

Executing a Matlab command

For example, to transpose a matrix, use the following command:

```
engEvalString(matlabEngine, "A = A'");
```

Retrieving feedback from Matlab

It can be useful when running a Matlab command to see the result of running the command. This is done by instantiating a buffer which reads the result from Matlab every time any command is executed:

```
char matlabBuffer[1024];

...

engOutputBuffer(matlabEngine, matlabBuffer, 1024);
```

Sending data to Matlab

It can prove quite troublesome to send dynamic multidimensional arrays to Matlab. However, it can be achieved using the following function:

```
void addMultiDimensionalArray(int w, int h, string array, double
**values)
{
    mxArray *mxValues;
    mxValues = mxCreateDoubleMatrix(h, w, mxREAL);
```

```
    double *mP;
    mP = mxGetPr(mxValues);

    for( int i = 0; i < w; i++ )
    {
        for( int j = 0; j < h; j++ )
        {
            mP[i * h + j] = values[j][i];
        }
    }

    engPutVariable(matlabEngine, array.c_str(), mxValues);
}
```

Retrieving data from Matlab

The following method is used to get an array from Matlab. The array is always retrieved and stored as a one-dimensional array in C++ (`cData`).

```
double *cData;
mxArray *mData;
mData = engGetVariable(matlabEngine, "name-of-matrix");
cData = mxGetPr(mData);
```

The array can be restructured into a multidimensional array by employing the following principle. A $n \times m$ array containing elements v_{ij}, $i = 1..n$, $j = 1..m$, is stored of the form $[v_{11},..,v_{1m},v_{n1},..,v_{nm}]$. For example, a 2×2 matrix is stored as $[v_{11}, v_{12}, v_{21}, v_{22}]$.

Appendix C

User Evaluation - Models

The following figures are the sketched models from tasks 1 and 2 in the user evaluation discussed in Section 4.11. These tasks involved volunteers using the sketching interface, developed as a part of this thesis, to create 3D faces which would resemble a provided target face. Both tasks featured the same target face. However, task 1 only provided a verbal description of the target, while task 2 provided a picture of it from the front and profile view.

The figures are ordered based on the user number. Every figure is laid out in the same fashion where the target face is at the top, the sketched model from task 1 is on the left, and the sketched model from task 2 is on the right. Accompanying each sketched model is a colour coded model which visualises the vertex distance to the target model (model error). The colour scale is fixed to an absolute value for all users to allow a direct comparison of the users' models.

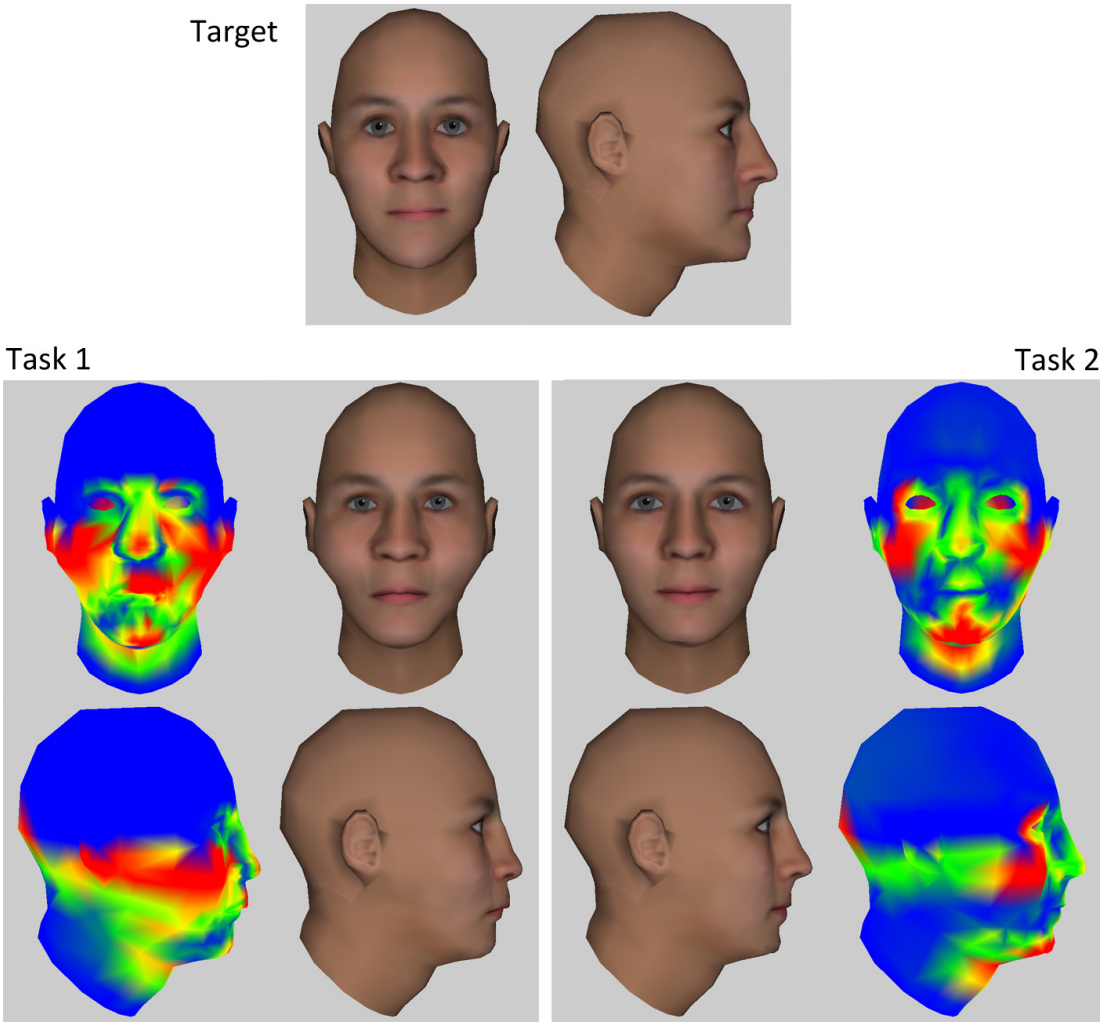


Figure C.1: User 1

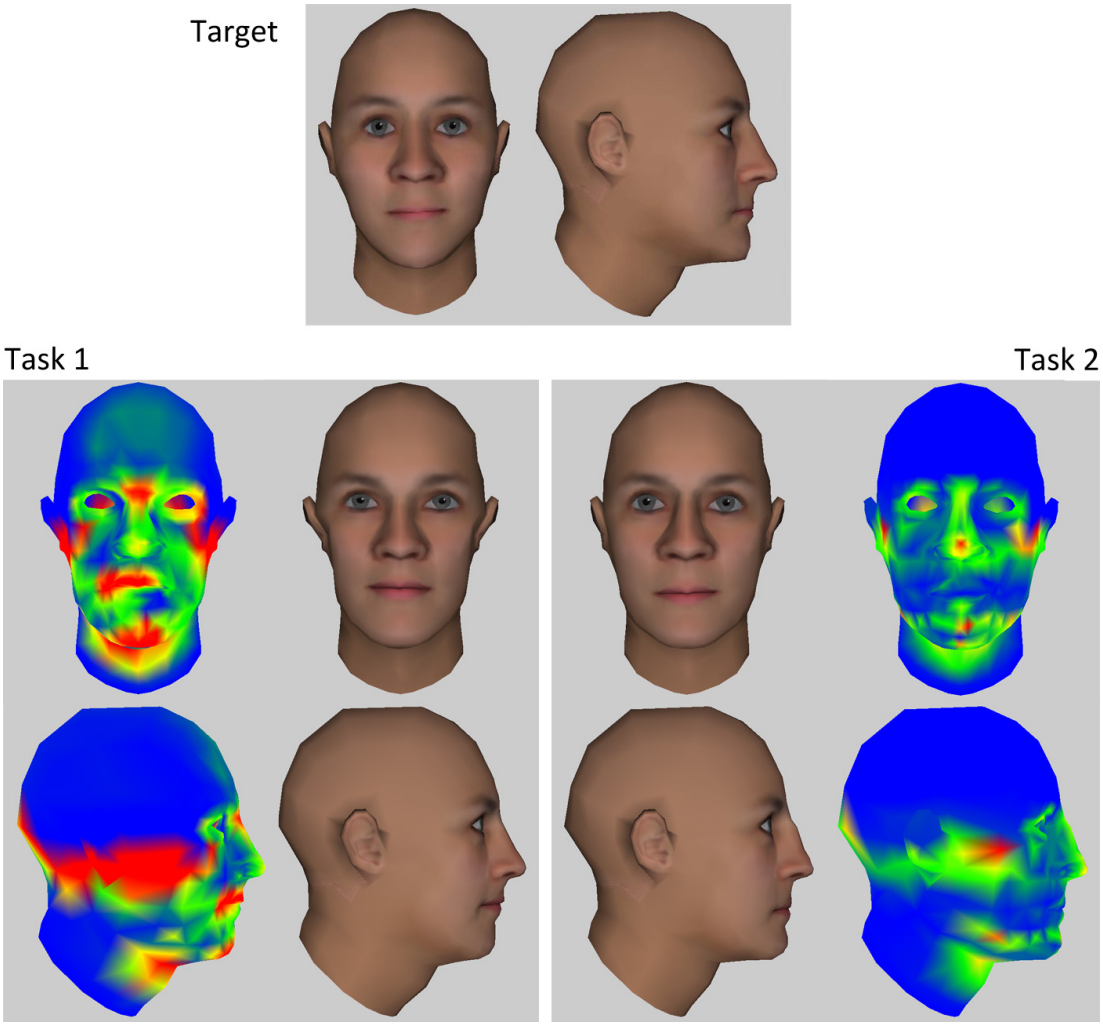


Figure C.2: User 2

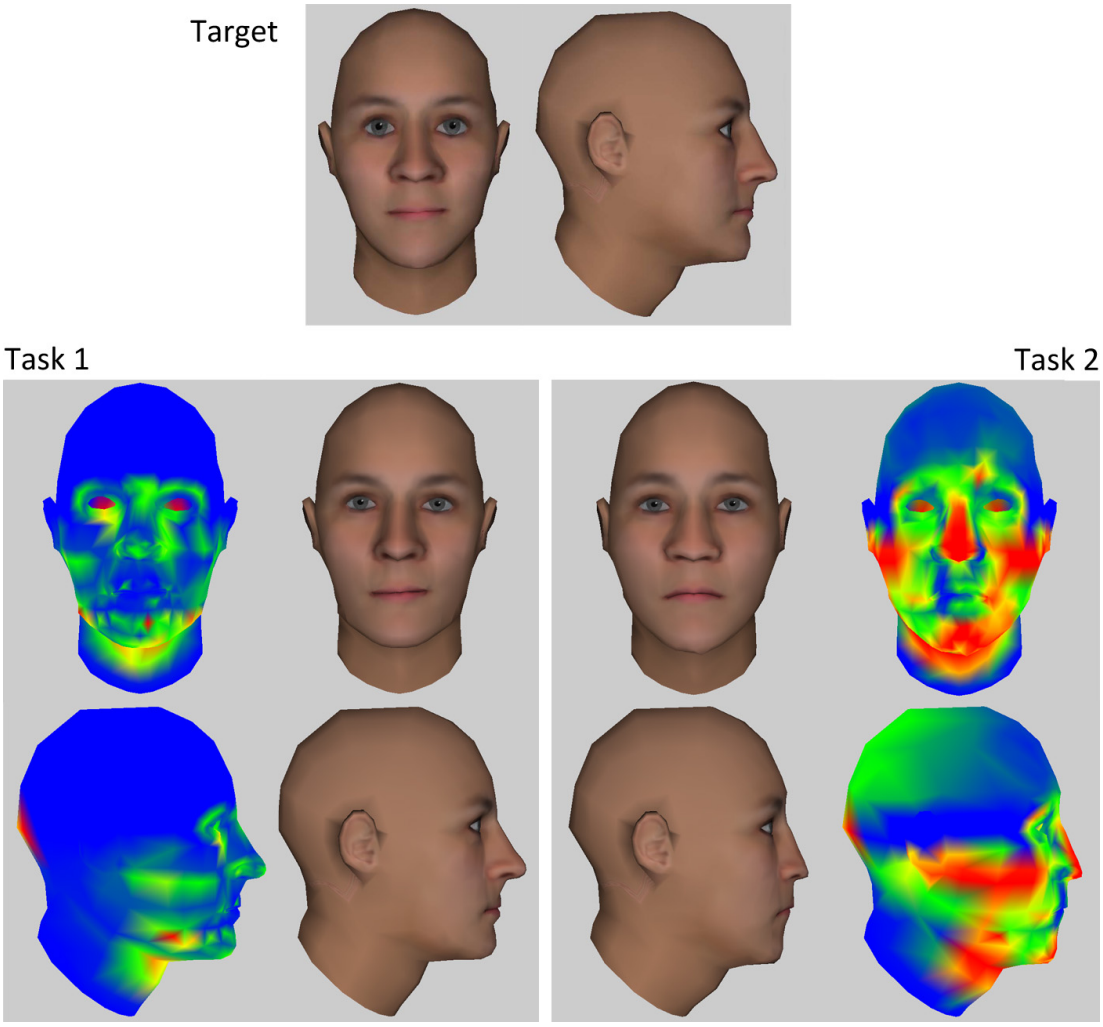


Figure C.3: User 3

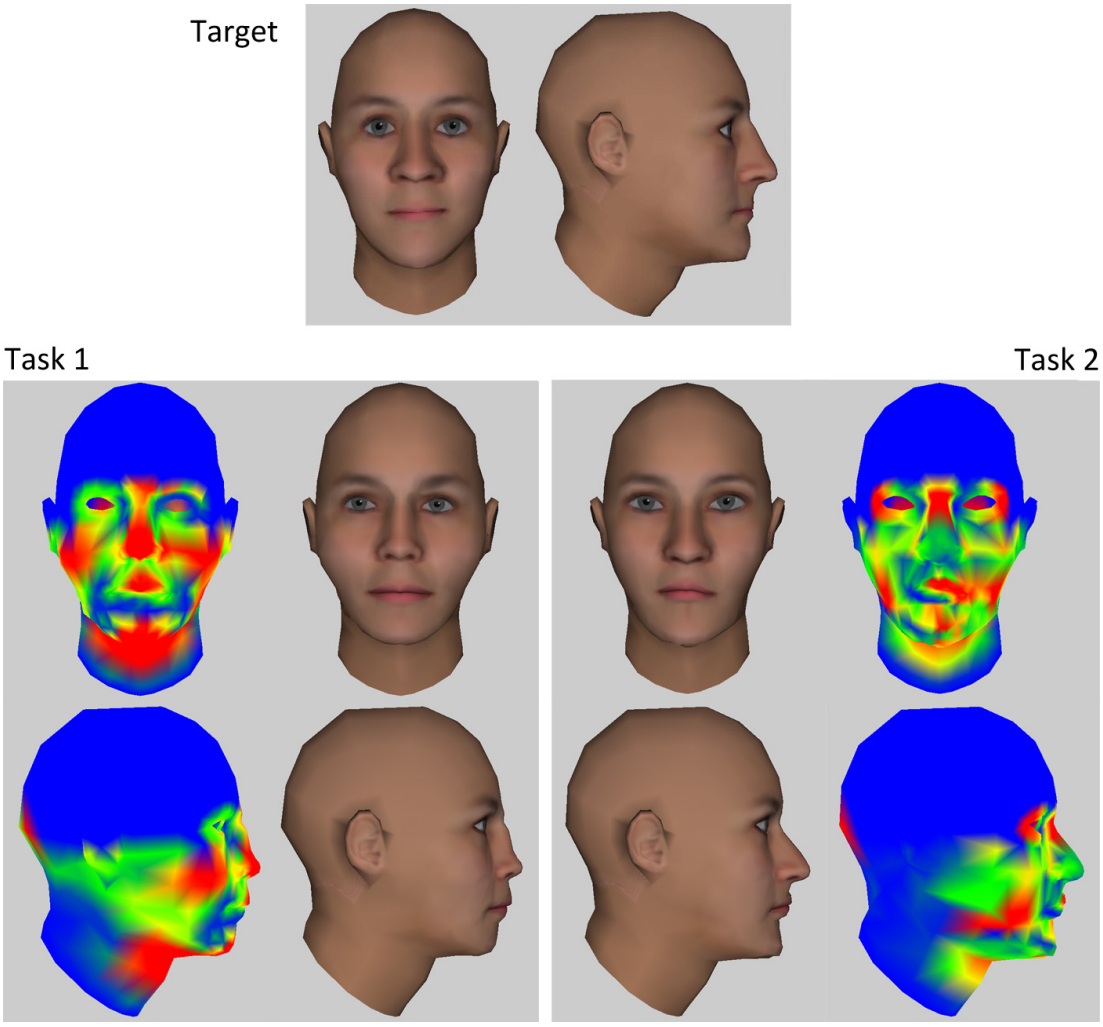


Figure C.4: User 4

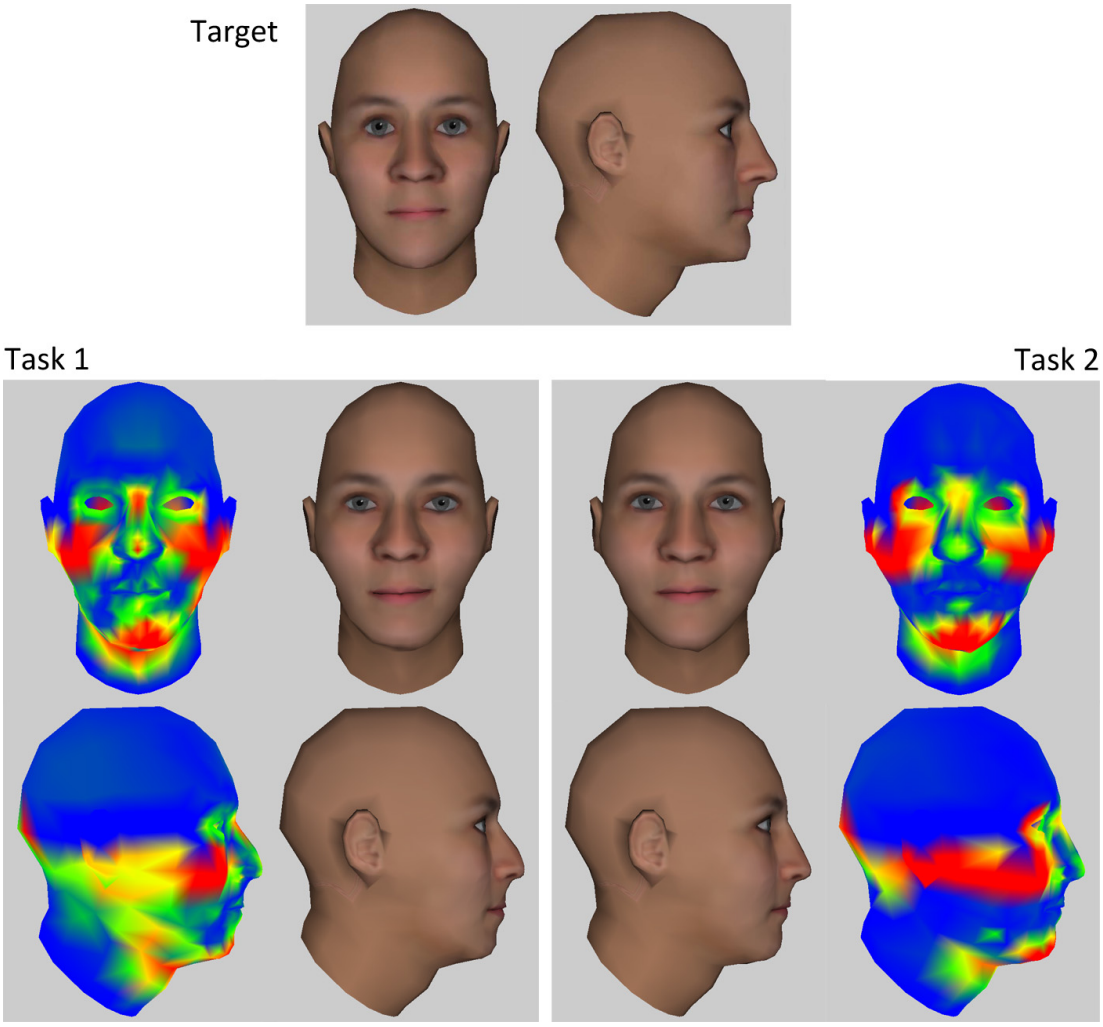


Figure C.5: User 5

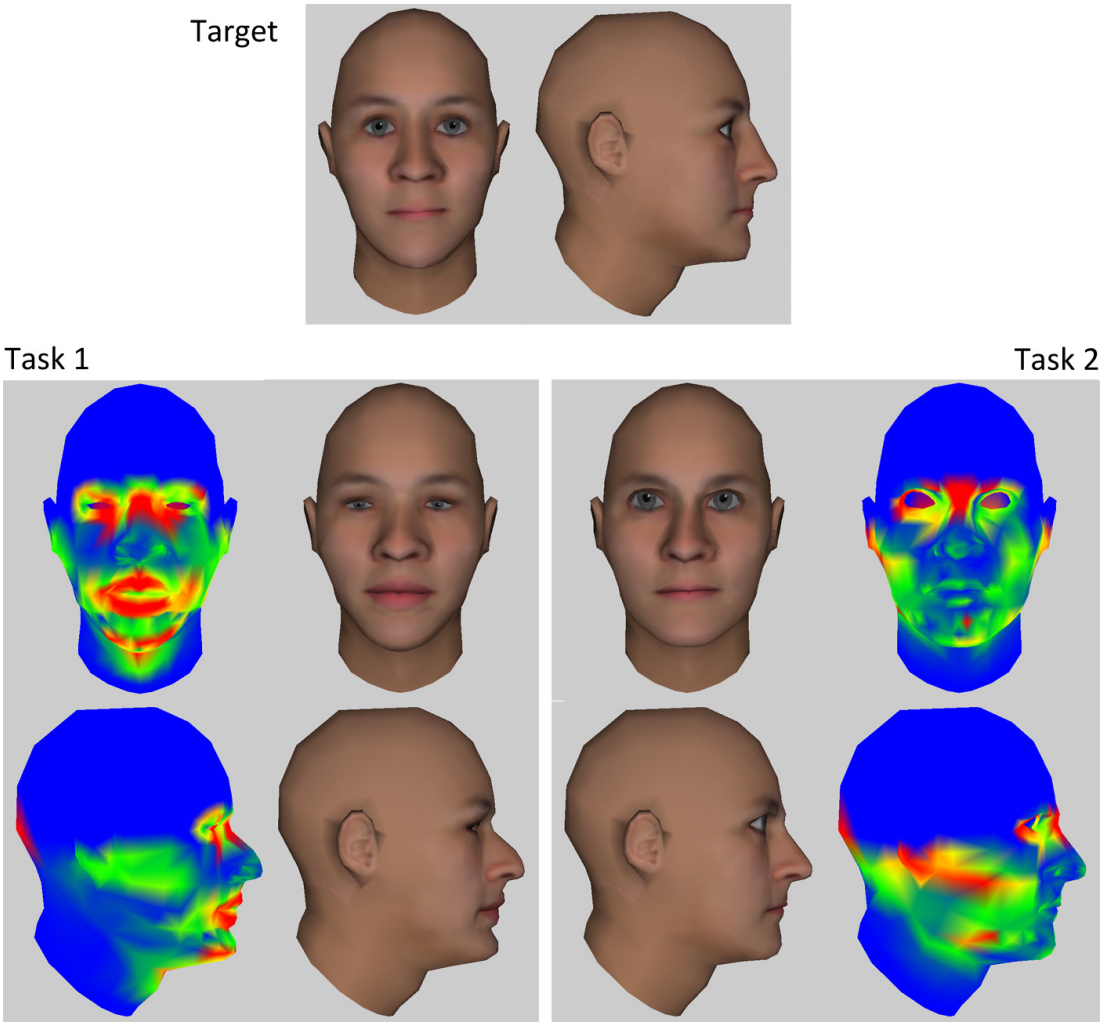


Figure C.6: User 6

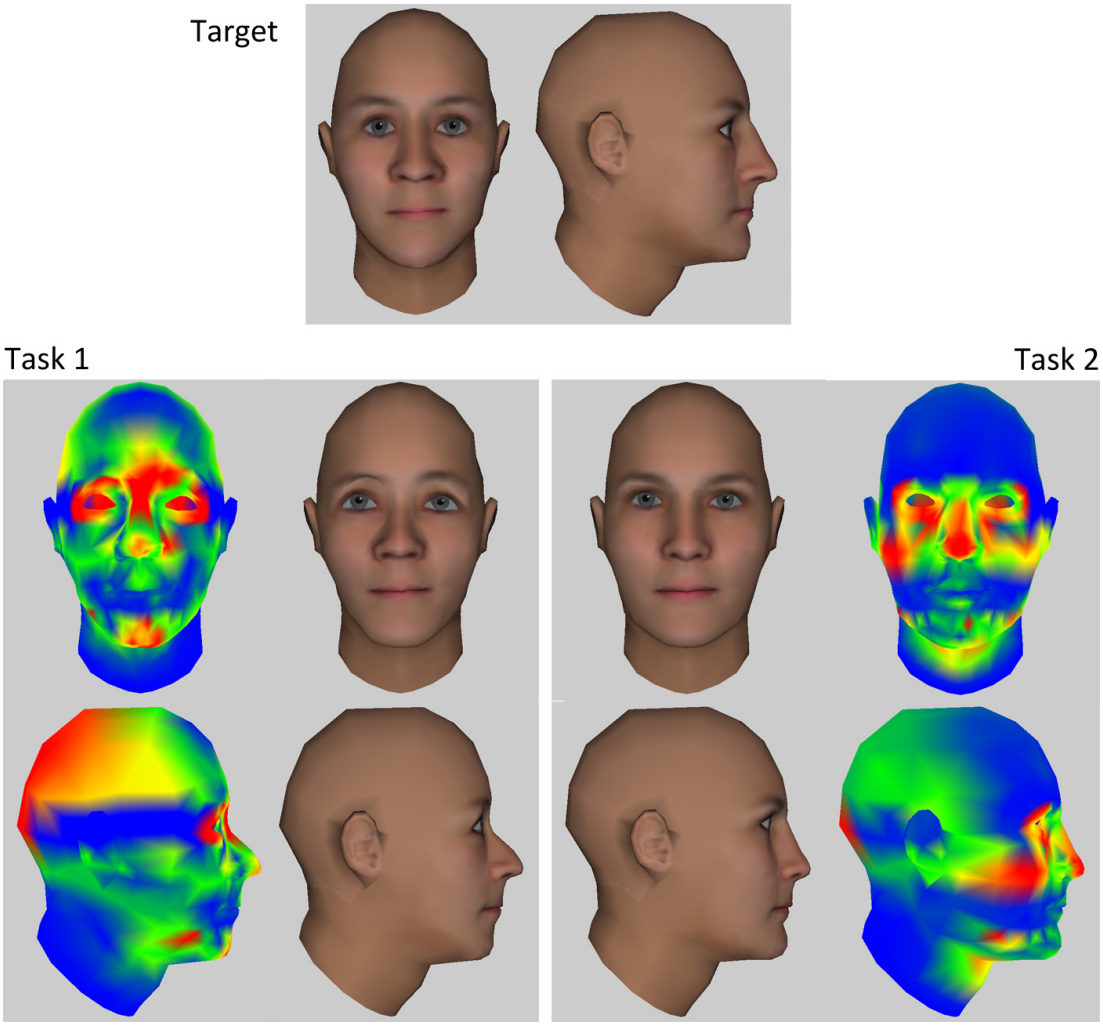


Figure C.7: User 7

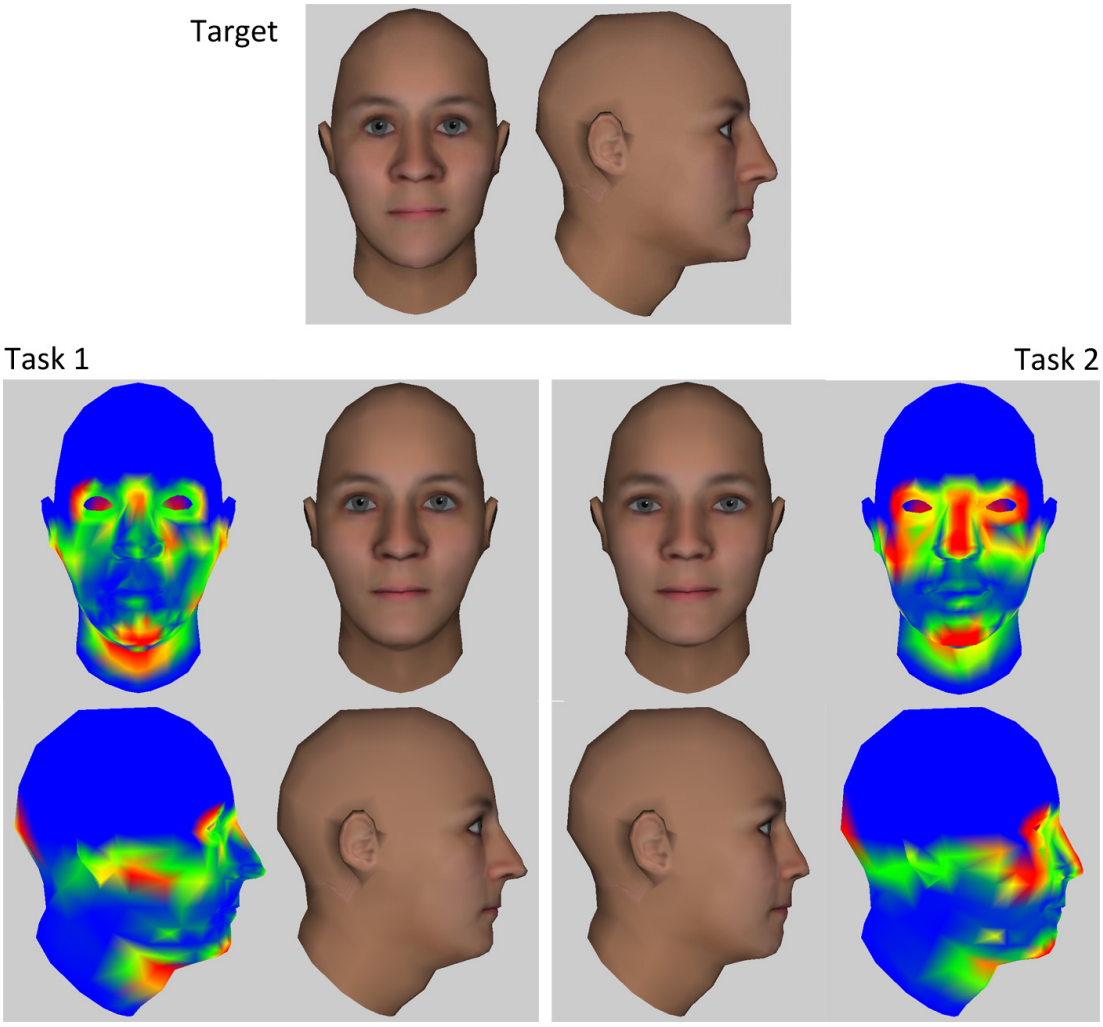


Figure C.8: User 8

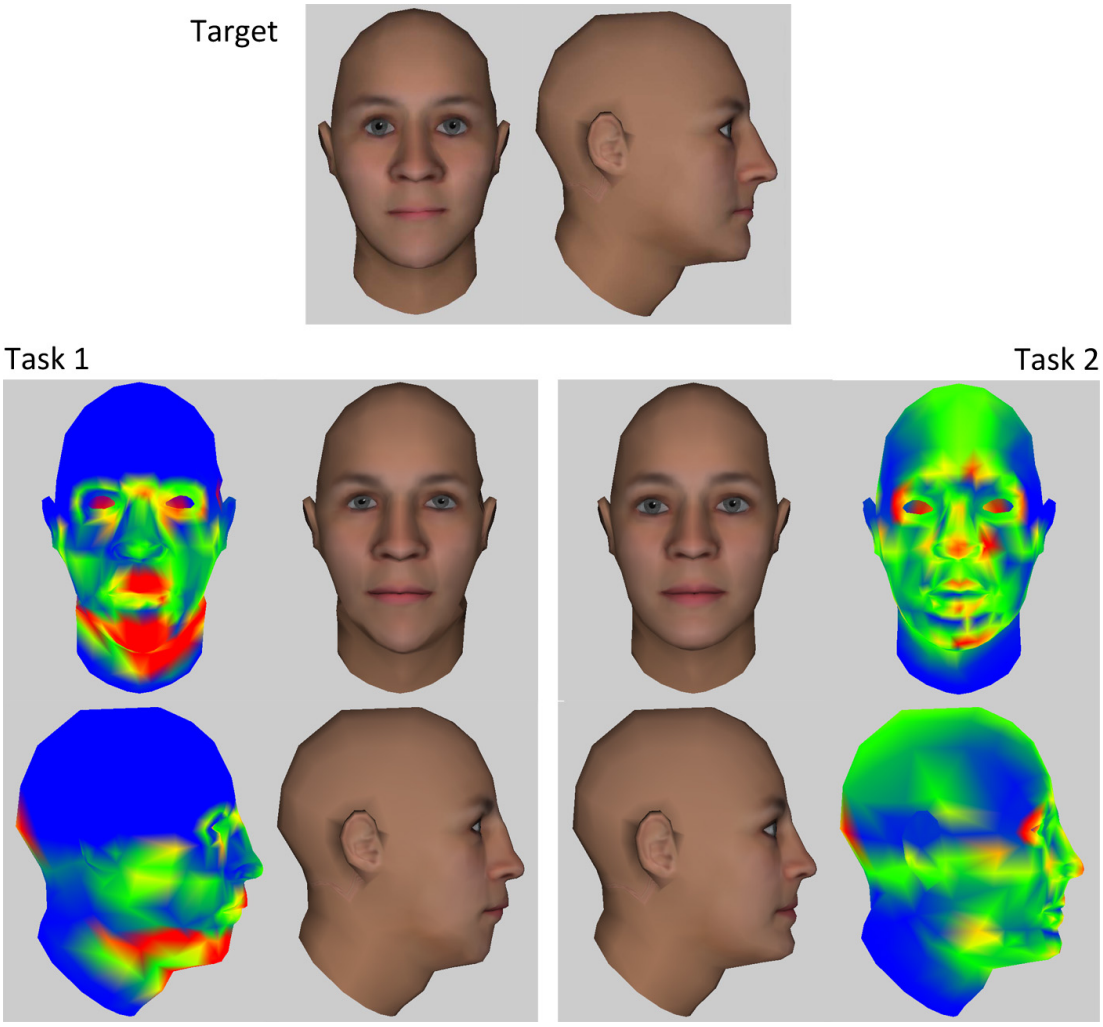


Figure C.9: User 9

References

[ABHS06] I. Albrecht, V. Blanz, J. Haber, and Seidel. Creating face models from vague mental images. *Eurographics 2006*, 25(3), 2006.

[ACH09] A. Albin-Clark and T.L.J. Howard. Automatically generating virtual humans using evolutionary algorithms. In *In Proceedings of Theory and Practice of Computer Graphics*, Cardiff, 2009. Eurographics.

[and96] Horace Ho-Shing Ip and. Constructing a 3d individualized head model from two orthogonal views. *The Visual Computer*, 12(5):254–266, 1996.

[ASW93] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace. Automatic creation of 3d facial models. *IEEE Comput. Graph. Appl.*, 13(5):16–22, 1993.

[BA83] P.J. Burt and E.H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[Bad79] A. Baddeley. The limitations of human memory: implications for the design of retrospective surveys. *The Recall Method in Social Surveys. London: University of London Institute of Education*, pages 13–27, 1979.

[BAQ+05] Florence Bertails, Basile Audoly, Bernard Querleux, Frédéric Leroy, Jean-Luc Lévêque, and Marie-Paule Cani. Predicting natural hair shapes by solving the statics of flexible rods. In J. Dingliana and

F. Ganovelli, editors, *Eurographics (short papers)*. Eurographics, August 2005.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[BM96] R. Brunelli and O. Mich. Spotit! an interactive identikit system. *Graphical Models and Image Processing*, 58(5):399–404, 1996.

[BPK00] N. Brace, G. Pike, and R. Kemp. Investigating e-fit using famous faces. In In A. Czerederecka, T. Jaskiewicz-Obydzinska, and J. Wojcikiewicz (Eds.), editors, *Forensic psychology and law*, pages 272–276, Krakow, 2000. Institute of Forensic Research Publishers.

[BPV06] Curzio Basso, Pascal Paysan, and Thomas Vetter. Registration of expressions data using a 3d morphable model. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 205–210, Washington, DC, USA, 2006. IEEE Computer Society.

[BSVS04] V. Blanz, K. Scherbaum, T. Vetter, and H.P. Seidel. Exchanging faces in images. *Computer Graphics Forum*, 23(3):669–676, 2004.

[BV99] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[CB08] Erika Chuang and Chris Bregler. Performance driven facial animation using blendshape interpolation. 2008.

[CBK+06] Cristóbal Curio, Martin Breidt, Mario Kleiner, Quoc C. Vuong, Martin A. Giese, and Heinrich H. B. Semantic 3d motion retargeting for facial animation. In *APGV '06: Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, pages 77–84, New York, NY, USA, 2006. ACM.

[CF04] Tzu-Pei Grace Chen and Sidney Fels. Exploring gradient-based face navigation interfaces. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 65–72, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[CGL+08] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM.

[CH08] S. Cheon and S. Han. A template-based reconstruction of plane-symmetric 3d models from freehand sketches. *Computer-Aided Design*, 40(9):975–986, September 2008.

[Chi05] Jane Chiu. Sketchex: Sketch-based interface for 3d face modeling. Master's thesis, University of California at Los Angeles, 2005.

[CJ06] Edwin Chang and Odest Chadwicke Jenkins. Sketching articulation and pose for facial animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 271–280, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[CLR+04] Hong Chen, Ziqiang Liu, Chuck Rose, Yingqing Xu, Heung-Yeung Shum, and David Salesin. Example-based composite sketching of human portraits. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 95–153, New York, NY, USA, 2004. ACM Press.

[CSD+09] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? *ACM Trans. Graph.*, 28(3):1–9, 2009.

[CSJ05] Dirk Colbry, George Stockman, and Anil Jain. Detection of anchor points for 3d face verification. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 118, Washington, DC, USA, 2005. IEEE Computer Society.

[CSSJ05] Joseph Jacob Cherlin, Faramarz Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 137–145, New York, NY, USA, 2005. ACM Press.

[CYvdP05] Dana Sharon Chen Yang and Michiel van de Panne. Sketch-based modeling of parameterized objects. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 63–72, 2005.

[DAC+03] James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popovic;, and David Salesin. A sketching interface for articulated figure animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 320–328, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[DC76] M. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[DFR04] D. DeCarlo, A. Finkelstein, and S. Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 15–145, 2004.

[DFRS03] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[dHHR07] Adelaide de Heering, Sarah Houthuys, and Bruno Rossion. Holistic face processing is mature at 4 years of age: Evidence from the composite face effect. *Journal of Experimental Child Psychology*, 96(1):57 – 70, 2007.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[DMS98] Douglas DeCarlo, Dimitris Metaxas, and Matthew Stone. An anthropometric face model using variational techniques. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 67–74, New York, NY, USA, 1998. ACM Press.

[DR07] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 63–70, New York, NY, USA, 2007. ACM.

[EBE95] Lynn Eggli, Beat D. Bruderlin, and Gershon Elber. Sketching as a solid modeling tool. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 313–322, New York, NY, USA, 1995. ACM Press.

[Eck91] M. Eck. Interpolation methods for reconstruction of 3d surfaces from sequences of planar slices. *CAD und Computergraphik*, 13(5):109–120, 1991.

[EF78] P. Ekman and W. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto, 1978.

[EH03] Alaa El-Hussuna. Statistical variation of three dimensional face models. Master's thesis, IT-University of Copenhagen, 2003.

[FBR+07] C.D. Frowd, V. Bruce, D. Ross, A. McIntyre, and P.J.B. Hancock. An application of caricature: how to improve the recognition of facial composites. *Visual Cognition*, (15):1–31, 2007.

[FKS+04] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004.

[FLS+07] Paul Fedor, Peter Levius, Hong Suck Suh, Matt Hartle, Mark Snoswell, and Steven Stahlberg. *Creative ESSENCE: The Face*. Ballistic Publishing, second edition edition, 2007.

[FN] FG-NET. Fg-net aging database. `http://www.fgnet.rsunit.com`.

[Fro02] Charlie Frowd. *EvoFIT: A Holistic, Evolutionary Facial Imaging System*. PhD thesis, University of Stirling, 2002.

[GM08] Orn Gunnarsson and Steve Maddock. Sketching faces. In *Proc. Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Annecy, France, June 11-13 2008.

[GS09] O. Gunnarsson and S.Maddock. Sketch-based facial animation. In *Facial Analysis and Animation, One-Day BMVA Symposium*, Informatics Forum, 10 Crichton St, Edinburgh, UK, June 10th 2009. The School of Informatics, University of Edinburgh. (poster, refereed).

[GVL96] G. H. Golub and C. F. Van Loan. *Matrix Computations, 3rd edition*. The Johns Hopkins University Press, 1996.

[HADF+05] A. Henzen, N. Ailenei, F. Di Fiore, F. Van Reeth, and J. Patterson. Sketching with a low-latency electronic ink drawing tablet. In *Proceedings GRAPHITE 2005*, 2005.

[HMR+09] YueZhu Huang, Ralph R. Martin, Paul L. Rosin, XiangXu Meng, and ChengLei Yang. Expressive line drawings of human faces from range images. *Science in China Series F: Information Sciences*, 52(2):295–307, 2009.

[HZ00] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, 2000.

[Iga07] Takeo Igarashi. Sketch-based interfaces for interactive computer graphics. In *SIGGRAPH '07: Course 3*, 2007.

[IH03] Takeo Igarashi and John F. Hughes. Smooth meshes for sketch-based freeform modeling. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 139–142, New York, NY, USA, 2003. ACM Press.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH 99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[JDA07] Tilke Judd, Frédo Durand, and Edward Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[KFS+07] István Kokai, JFinger, Randall C. Smith, Richard Pawlicki, and Thomas Vetter. Example-based conceptual styling framework for automotive shapes. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 37–44, New York, NY, USA, 2007. ACM.

[KH06] O.A. Karpenko and J.F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (TOG)*, 25(3):589–598, 2006.

[KHR02] O. Karpenko, J.F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3):585, 2002.

[KHR04] O. Karpenko, J.F. Hughes, and R. Raskar. Epipolar methods for multi-view sketching. *Proceedings of the 2004 Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM-04)*, pages 167–174, 2004.

[KHYS02] Kolja Kahler, Jorg Haber, Hitoshi Yamauchi, and Hans-Peter Seidel. Head shop: Generating animated head models with anatomical structure. *SIGGRAPH*, pages 55–64, 2002.

[KK06] Dae Hyun Kim and Myoung-Jun Kim. A new modeling interface for the pen-input displays. *Computer-Aided Design*, 38(3):210–223, 2006.

[KMM+02] R.D. Kalnins, L. Markosian, B.J. Meier, M.A. Kowalski, J.C. Lee, P.L. Davidson, M. Webb, J.F. Hughes, and A. Finkelstein. Wysiwyg npr: drawing strokes directly on 3d models. *ACM Transactions on Graphics*, 21(3):755–762, 2002.

[KMMtT91] Prem Kalra, Angelo Mangili, Nadia Magnenat-thalmann, and Daniel Thalmann. Smile: A multilayered facial animation system. In *In T.L. Kunii, editor, Modeling in Computer Graphics*, pages 189–198. Springer-Verlag, 1991.

[KS96] J. Kolar and E. Salter. Craniofacial anthropometry: Practical measurements of the head and face for clinical, surgical and research use. charles c, 1996.

[KSvdP09] Vladislav Kraevoy, Alla Sheffer, and Michiel van de Panne. Modeling from contour drawings. In *SBIM '09: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 37–44, New York, NY, USA, 2009. ACM.

[Law04] Neil Lawrence. Gaussian process latent variable model. In *Neural Information Processing Systems (NIPS) 16*, 2004.

[Law05] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LCODL08] Tommer Leyvand, Daniel Cohen-Or, Gideon Dror, and Dani Lischinski. Data-driven enhancement of facial attractiveness. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3), aug 2008.

[LCXS07] Manfred Lau, Jinxiang Chai, Ying-Qing Xu, and Heung-Yeung Shum. Face poser: interactive modeling of 3d facial expressions using model priors. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 161–170, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[LDI06] Joseph LaViola, Randall Davis, and Takeo Igarashi. An introduction to sketch-based interfaces. In *SIGGRAPH '06: Material presented at the ACM SIGGRAPH 2006 conference*, page 18, New York, NY, USA, 2006. ACM Press.

[LEKM03] Manuel S. Lorenzo, James D. Edge, Scott A. King, and Steve Maddock. Use and re-use of facial motion capture data. *Vision, Video and Graphics*, pages 1–8, 2003.

[LF08] Jeehyung Lee and Thomas Funkhouser. Sketch-based search and composition of 3d models. In *Proc. Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 97–104, Annecy, France, June 11-13 2008.

[Llo82] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[LMLH07] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F. Hughes. Line drawings via abstracted shading. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 18, New York, NY, USA, 2007. ACM.

[LTW95] Yuencheng Lee, Demetri Terzopoulos, and Keith Walters. Realistic modeling for facial animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62, New York, NY, USA, 1995. ACM Press.

[Mac67] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[Mal05] Shahzad Malik. A sketching interface for modeling and editing hairstyles. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 185–194, 2005.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[MK92] Robert Mauro and Michael Kubovy. Caricature and face recognition. *Memory and Cognition*, 20:433–440, 1992.

[ML05] M. Masry and H. Lipson. A sketch-based interface for iterative design and analysis of 3d objects. *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 109–118, 2005.

[MPK09] Umar Mohammed, Simon J. D. Prince, and Jan Kautz. Visiolization: generating novel facial images. *ACM Trans. Graph.*, 28(3):1–8, 2009.

[MQW09] Chen Mao, Sheng Feng Qin, and David Wright. A sketch-based approach to human body modelling. *Computers Graphics*, In Press, Corrected Proof, 2009.

[MS04] AB Moreno and A. Sánchez. Gavabdb: a 3d face database. *In Proc. 2nd COST275 Workshop on Biometrics on the Internet*, pages 75–80, 2004.

[MSK00] Jun Mitani, Hiromasa Suzuki, and Fumihiko Kimura. *3D Sketch: Sketch-based Model Reconstruction and Rendering*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[MT83] Leonard S. Mark and James T. Todd. The perception of growth in three dimensions. *Perception Psychophysics*, 33(2):193–196, 1983.

[MT91] A. Pentland M. Turk. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, 1991.

[MYH+06] Jun Murakawa, Ilmi Yoon, Tracie Hong, , and Edward Lank. Parts, image, and sketch based 3d modeling method. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 67–74, 2006.

[Nab01] Ian Nabney. *NETLAB: Algorithms for Pattern Recognition*. Springer-Verlag London Ltd, November 2001.

[NBHN02] Hayley Ness, V Bruce, P J B Hancock, and C Newman. Four heads are better than one: combining face composites yields improvements in face likeness. *Journal of Applied Psychology*, 87(5):894–902, 2002.

[NF07] Gabriele Nataneli and Petros Faloutsos. Sketching facial expressions. In *ACM SIGGRAPH 2007 Technical Sketches*, page 60, 2007.

[NLE+99] U. Neumann, J. Li, J. Enciso, D. Noh, and T. Kim. Constructing a realistic head animation mesh for a specific person, 1999.

[NSACO05] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, 2005.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, 2004.

[oND] University of Notre Dame. Biometrics database. `http://www.nd.edu/~cvrl/UNDBiometricsDatabase.html`.

[OSSJ09] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Technical section: Sketch-based modeling: A survey. *Comput. Graph.*, 33(1):85–103, 2009.

[oY03] The University of York. The 3d face database. 2003.

[Pak02] Algirdas Pakstas. *MPEG-4 Facial Animation: The Standard, Implementation and Applications.* John Wiley & Sons, Inc., New York, NY, USA, 2002.

[Par72] Frederick I. Parke. Computer generated animation of faces. In *ACM'72: Proceedings of the ACM annual conference*, pages 451–457, New York, NY, USA, 1972. ACM Press.

[Par74] Frederic Ira Parke. *A parametric model for human faces.* PhD thesis, 1974.

[PCN05] Manuel Contero Pedro Company, Ana Piquer and Ferran Naya. A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computer and Graphics*, 29:892–904, 2005.

[PFS+05] P. Jonathon Phillips, Patrick J. Flynn, Todd Scruggs, Kevin W. Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and

William Worek. Overview of the face recognition grand challenge. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 947–954, Washington, DC, USA, 2005. IEEE Computer Society.

[PHL+98] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. *Computer Graphics*, 32(Annual Conference Series):75–84, 1998.

[PHWF01] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[PSAR07] E. Patterson, A. Sethuram, M. Albert, and K. Ricanek. Comparison of synthetic face aging to age progression by forensic sketch artist. In *Proceedings of the Seventh IASTED International Conference on Visualization, Imaging, and Image Processing*, pages 247–252, Palma de Mallorca, Spain, August 2007. ACTA Press.

[PTJ08] Unsang Park, Yiying Tong, and Anil K. Jain. Face recognition with temporal invariance: A 3d aging model. In *FG*, pages 1–7. IEEE, 2008.

[PW96] Frederic I. Parke and Keith Waters. *Computer facial animation*. A. K. Peters, Ltd., Natick, MA, USA, 1996.

[RBC02] I. Rudomin, A. Bojorquez, and H. Cuevas. Statistical generation of 3d facial animable models. *Proceedings of the Shape Modeling International 2002 (SMI'02)*, 2002.

[RCB09] Narayanan Ramanathan, Rama Chellappa, and Soma Biswas. Computational methods for modeling facial aging: A survey. *J. Vis. Lang. Comput.*, 20(3):131–144, 2009.

[RMW74] R.E.Shaw, M.McIntyre, and W.Mace. The role of symmetry in event perception. *Perception: Essays in honor of James J. Gibson*, pages 276–310, 1974.

[SBOR07] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell. Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94(11):1948–1962, January 2007.

[SBr05] Daniel Skora, Jan Burinek, and Ji˚ra. Sketching cartoons by example. In *2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBM'05)*, pages 27–34, Dublin, Ireland, August 2005.

[SCOL+04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. R, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[SD05] Tevfik Metin Sezgin and Randall Davis. Hmm-based efficient sketch recognition. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 281–283, New York, NY, USA, 2005. ACM Press.

[SDN09] Tanasai Sucontphunt, Zhigang Deng, and Ulrich Neumann. Crafting personalized facial avatars using editable portrait and photograph example. In *VR '09: Proceedings of the 2009 IEEE Virtual Reality Conference*, pages 259–260, Washington, DC, USA, 2009. IEEE Computer Society.

[SFNTH05] Ari Shapiro, Petros Faloutsos, and Victor Ng-Thow-Hing. Dynamic animation and control environment. In *GI '05: Proceedings of Graphics Interface 2005*, pages 61–70, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[SJ99] Simone Santini and Ramesh Jain. Similarity measures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):871–883, 1999.

[SL03] E. Saund and E. Lank. Stylus input and editing without prior selection of mode. *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 213–216, 2003.

[SMND08] T. Sucontphunt, Z. Mo, U. Neumann, and Z. Deng. Interactive 3d facial expression posing through 2d portrait manipulation. In *GI'08: Proc. of Graphics Interface*, Windsor, Ontario, Canada, 2008.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

[SS] Signal and Image Centre (SIC). 3d rma: 3d database. `http://www.sic.rma.ac.be/~beumier/DB/3d_rma.html`.

[SSSB07] Kristina Scherbaum, Martin Sunkel, Hans-Peter Seidel, and Volker Blanz. Prediction of individual non-linear aging trajectories of faces. In *The European Association for Computer Graphics, 28th Annual Conference, EUROGRAPHICS 2007*, volume 26 of *Computer Graphics Forum*, pages 285–294, Prague, Czech Republic, 2007. The European Association for Computer Graphics, Blackwell.

[Sut80] Ivan Sutherland. *Sketchpad: A Man-Machine Graphical Communication System.* Garland Publishers, 1980.

[SvdP06] D. Sharon and M. van de Panne. Constellation models for sketch recognition. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 19–26, 2006.

[Tau95] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

[TB97] Michael Tipping and Christopher Bishop. Probabilistic principal component analysis. Technical report, NCRG/97/010, Neural Computing Research Group, Aston University, September 1997., 1997.

[TB99] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[TN96] Bulthoff H.H. Troje N.F. Face recognition under varying poses: The role of texture and shape. *Vision Research*, 36:1761–1771(11), 1996.

[TO99] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[TW04] X. Tang and X. Wang. Face sketch recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):50–57, 2004.

[TZF04] C.L. Tai, H. Zhang, and J.C.K. Fong. Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum*, 23(1):71–83, 2004.

[UGB06] C.W. Urquhart, D.S. Green, and E.D. Borland. 4d capture using passive stereo photogrammetry. *IET Conference Publications*, 2006(CP516):196–196, 2006.

[Vit67] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[Wat87] Keith Waters. A muscle model for animation three-dimensional facial expression. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 17–24, New York, NY, USA, 1987. ACM.

[Wat99] Alan Watt. *3D Computer Graphics, Third Edition*. Addison Wesley, 1999.

[WBC07] Jamie Wither, Florence Bertails, and Marie-Paule Cani. Realistic hair from a sketch. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 33–42, Washington, DC, USA, 2007. IEEE Computer Society.

[WCO05] G.L. Wells, S.D. Charman, and E.A. Olson. Building face composites can harm lineup identification performance. *Journal of Experimental Psychology: Applied*, 11:147–156, 2005.

[WD09] Di Wu and Qionghai Dai. Sketch realizing: Lifelike portrait synthesis from sketch. In *CGI09: Proceedings of Computer Graphics International 2009*, Victoria, British Columbia, Canada, 2009.

[WFJ+05] B. Wyvill, K. Foster, P. Jepp, R. Schmidt, MC Sousa, and JA Jorge. Sketch based construction and rendering of implicit models. *Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 67–74, 2005.

[Wil90] Lance Williams. Performance-driven facial animation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 235–242, New York, NY, USA, 1990. ACM Press.

[WTBS07] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Shapepalettes: interactive normal transfer via sketching. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 44, New York, NY, USA, 2007. ACM.

[XCZL08] Zijian Xu, Hong Chen, Song-Chun Zhu, and Jiebo Luo. A hierarchical compositional model for face representation and sketching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:955–969, 2008.

[Xia04] D. Xiao. *Sketch-based Instancing of Parameterized 3D Models*. PhD thesis, The University of British Columbia, 2004.

[YHR04] Ioannis A. Ypsilos, Adrian Hilton, and Simon Rowe. Video-rate capture of dynamic face shape and appearance. In *FGR*, pages 117–122, 2004.

[yNN01] Jun yong Noh and Ulrich Neumann. Expression cloning. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 277–288. ACM Press / ACM SIGGRAPH, 2001.

[ZD07] Ulrich Neumann Zhigang Deng. *Data-Driven 3D Facial Animation*. Springer, 1st edition, 2007.

[Zha06] Yu Zhang. An efficient texture generation technique for human head cloning and morphing. In *GRAPP 2006, First International Conference on Computer Graphics Theory and Applications*, pages 267–275. Insticc Press, 2006.

[ZHH96] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, pages 163–170. Addison Wesley, 1996.

[ZNA08] Johannes Zimmermann, Andrew Nealen, and Marc Alexa. Sketching contours. *Computers Graphics*, 32(5):486 – 499, 2008.

[ZS03] Roman Zenka and Pavel Slavik. New dimension for sketches. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, pages 157–163, New York, NY, USA, 2003. ACM Press.

[ZSCS04] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, August 2004.