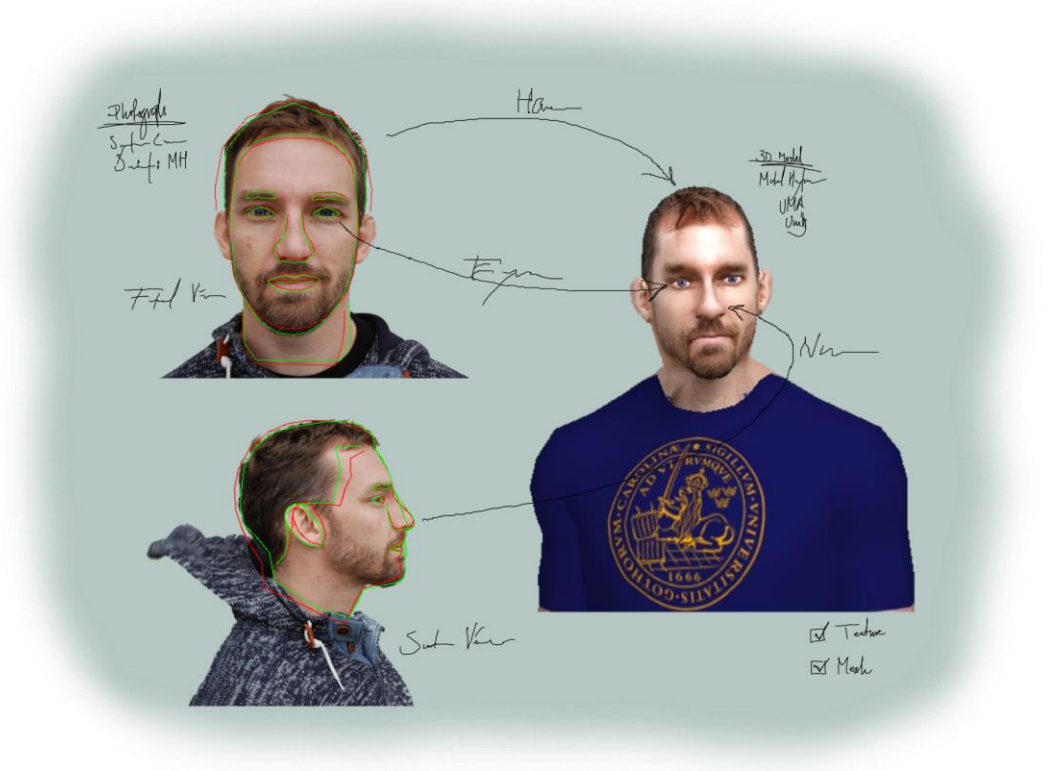
Selfie to Avatar

Creating a 3D Model of a Face from Front and Side View Photos



Author: Mikael Högström, mikael_hogstrom@hotmail.com

Supervisor: Prof. Kalle Åström, kalle@maths.lth.se

Asst. Supervisor: Erik Bylow, bylowerik@gmail.com

Centre for Mathematical Sciences

Lund University

2016-01-25

Abstract

In this thesis we present a system for creating a textured, animated 3D model of a user's face in real time. The model, which we call an avatar, is reconstructed from two photos of a subject, one from a front view and one from a side view. Each photo is segmented using Active Shape Models. The shapes determined by the segmentation are used to reconstruct a 3D model and the pixel colors of the images are used to texture it. In order to evaluate the system, we use a test where test respondents try to match real world photos of people to their corresponding avatars. Although we do not take any special care of modelling hair, which is a difficult problem, we achieve a recognition rate of 82.7 % which demonstrates that our method already can provide realistic avatars of a person with little input from the user.

Acknowledgements

I would like to thank my supervisors Prof. Kalle Åström and Erik Bylow for their support in technical aspects as well as for their moral support.

Their technical support has always been sound and their moral support has been absolutely invaluable. Many of our meetings were held when there was a tough technical hurdle to overcome and even if the solutions to the problems themselves were not thus instantly found the result was, every time, a strengthened morale on my part.

Furthermore, I would also like to thank my family and friends for not only their moral support but also for volunteering as test subjects for the developed system. Without all the data from more than 60 volunteers who took the time to take their selfies and sending them to me this project would not have been possible.

Content

1 Introduction 7
1.1 Background 7
1.2 Related work 9
1.2.1 3D Avatar store 9
1.2.2 EA Sports NBA 2015 10
1.2.3 EA Sports Game Face 10
1.2.4 Automated Avatar Creation for 3D Games 11
1.3 Capabilities and restrictions of existing systems 12
1.3.1 3D Avatar Store 12
1.3.2 NBA 2015 12
1.3.3 EA Games Game Face 12
1.3.4 Automated Avatar Creation for 3D Games 12
1.3.5 Comparison table 13
1.4 Problem description 13
1.5 Purpose 13
1.6 Research questions 14
1.7 Outline 14
2 Theory 16
2.1 Active Shape Models 16
2.1.1 Shape mean and variance 16
2.1.2 Edge search 16
2.1.3 Image scale space pyramid 18
2.1.4 Principal Component Analysis 19
2.2 Graphics pipeline 20
2.2.1 Shader program 20
2.2.2 Vertex 20
2.2.3 Triangle 21
2.2.4 Mesh 21
2.2.5 UV coordinates 21

2.2.6 Render target 22
2.3 Skeletal Animation 22
3 Methodology 24
3.1 Overview 24
3.2 Building and using Active Shape Model 25
3.2.1 Train two shapes, front view and side view of a face 26
3.2.2 Run ASM on both images 31
3.2.3 Setup target shapes in texture 33
3.2.4 Project pixel values from segmented source shape in image to target shape in texture 33
3.2.5 Setup eigenspace coordinate connections to bone hierarchy 36
3.2.6 Use eigenspace coordinates to set proper bone transforms in skinned model 39
3.2.7 Allow user to do final adjustments of bone transforms 39
3.2.8 Save data in a format reusable by games or other applications 40
4 Results and Discussion 42
4.1 Final result 42
4.1.1 Comparison of avatars from different cameras 44
4.1.2 Comparison to existing systems 46
4.2 Individual algorithm steps' performances 46
4.2.1 Image segmenting performance 46
4.2.2 Pixel projection performance 47
4.2.3 3D model adjustment performance 47
5 Conclusion 49
5.1 Final result 49
5.1.1 Quality differences between avatars created from different cameras 50
5.1.2 User interaction with the algorithm 50
5.2 Future work 51
5.2.1 Add hair with volume 51
5.2.2 3D reconstruction through a statistical method 52
5.2.3 3D reconstruction through structure from motion 52
5.2.4 Image processing 52
5.2.5 Eigenfaces to describe face texture 52

6 Critique....54
7 Bibliography....55
8 Appendix A- Face match test....57

1 Introduction

The word avatar originally stems from Indian mythology where an avatar was the body used to physically represent a god on earth. In modern language the meaning of the word avatar is mostly used to describe a representation of a person on the internet. This representation could be as simple as a chosen name on a forum or as advanced as an animated 3D model. In this work we will concern ourselves with the latter form. When using the word *avatar,* we will be focusing on a 3D representation of the user unless otherwise stated.

In early arcade games of the 1980's it was common to include a high score list where the user could write three letters. This made several youngsters spend too much of their time and money trying to get their name on that list. This could be seen as an early display of the power of avatars, as the players wanted to get the representation of themselves in a high position. Moving on from there the avatars got more advanced. Sometimes a player could choose a color on the controlled entity of the game, sometimes she could write a name at the beginning of the game which could be written on the screen during gameplay and then written out on the high score list after the game ended. Avatars moved on to include 2D images representing the users. This is still widely used today on internet forums. With the advent of 3D computer games sometimes a user could construct an avatar of their own by setting values for how wide the nose should be, what color the hair should have etc. As the massively multiplayer online games like Everquest and World of Warcraft entered the scene such representations became ever more important as now other people could meet this avatar online and learn to recognize her. Today some video games have been experimenting with letting the user make a copy of herself in the game to use as an avatar. This is a technology that is still in its youth and that we believe could be improved. We also believe that it will spread outside the realm of computer games and find other uses like representing the user in social networks, online shopping dressing rooms and probably other uses that we have not thought of yet.

In this document we will propose a method for constructing a 3D avatar of a user that looks as much as her in real life as possible, that is animated and ready for use.

1.1 Background

Virtual personalized avatars are used in a number of different contexts today. The most prominent use may be to represent a player in a videogame where the user can create an avatar to represent oneself in the game world. This is done setting the shape of different facial features such as nose, chin, ears and so on. The user can also often set color of skin, eyes and hair. These systems can be very detailed and given enough time the user may be able to recreate something very similar to their own face.

Other applications than games are also gaining ground such as representing a user in an online shopping dressing room, online meetings or other virtual reality (VR) contexts such as cognitive

behavioral therapy or training simulations. It could be argued that in VR applications the view is usually from a first person perspective and the user does not see her own face. There could however be other people in the same scene who could also see the avatar and there could be reflections of one's own face in mirrors or metallic surfaces so a lifelike representation of a user can still have a significant effect on immersion.

Most avatar creation systems are built from predefined 3D models where shape and texture is somehow modified in different ways to create avatars with different appearances. Some systems actually allow the users to create themselves in an automatic process to facilitate them in representing themselves in order to create a higher level of immersion. These are not yet very common but it can be assumed that as users get used to being able to quickly create such an avatar they will expect that other applications give the same possibility. In the future this might not be considered a fancy gadget in one app but rather an expected feature for all of them.

Today many people buy their clothes online. There are some companies that offer online dressing rooms where you can try on the clothes on an avatar of your own size and measurements. This too may be a necessity for a company to offer in the future.

Figure 1 Left: Example of an avatar in an online dressing room for a web shop. Image courtesy of Fitle company. Right: Example of a computer game (Star Wars - The Old Republic) manual avatar creation process. Note the sliders to the right used to edit the avatar.

1.2 Related work

Some automated avatar creation systems do exist already. Their approaches differ, some use specialized hardware, other use very demanding calculations that require server farms to run in reasonable time. Note that the systems mentioned here all work in real time. There are also offline systems that are part of the production pipeline that create very realistic representations of existing people. They often take very long time to use and utilize very specialized tools like laser scans and they can reproduce a person perfectly but as they cannot be used to recreate an arbitrary user they are not studied further in this work.

1.2.1 3D Avatar store

3D avatar store is an online system where users can upload images and reconstruct an avatar of themselves [1]. Developers can also implement an API for the system to embed this creation system in their own applications. It uses a server farm to deliver neural net produced characters that are textured, rigged and animated and ready to use in any game or application. It is a proprietary system and uses a set of algorithms that originally stems from the field of face identification, many which are unpublished, so its exact implementation is hard to compare to the others.

The algorithm automatically segments an image then lets the user manually fine tune that segmentation if needed. Then it uses the segmented image as input to a set of neural nets that perform the modeling and texturing.

Blake Senfter CEO of 3D Avatar Store explains:
"What they/we do is a series of small to medium sized trained algorithms (the neural nets) that each perform some computation that would normally be too time consuming for real time use. These are things like finding faces in images, fitting facial features in images, identifying individualizing characteristics within faces, identifying ethnic facial structure in images, identifying shading and highlights versus surface color variation, recovering skin and hair tones, identifying the presence of eyewear and jewelry, identifying eyewear reflection versus eye surface reflection, constructing an estimate of surface structure, refinements to various pieces and parts of the face & head with special purpose correction 'nets, and so on.

Each of these small trained algorithms are in a compiled data pipeline, with a memory footprint of about 1.8 gigs. When presented one or more images or video of an individual, the system generates a best guess of that person's true 3D head & face shape. This estimate exists in a statistical structure, which yet another trained algorithm pipeline collapses into a specific representation - as the system is able to place reconstruction results into different geometric formats (different topologies) before final output as some specific geometry file format (such as OBJ, or Collada, or STL.) (personal communication, 26 November, 2015)

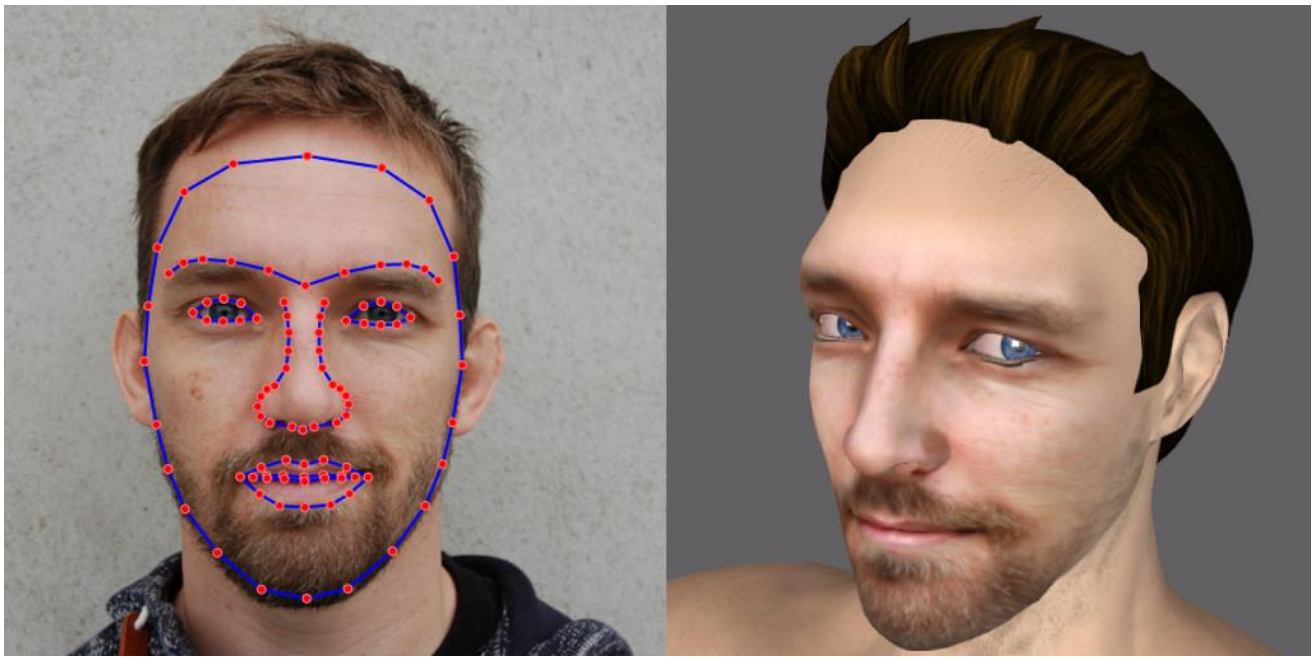


Figure 2 Avatar store example. On the left is the segmented image. Note that the eyes, hair style and color has been selected manually in the application.

1.2.2 EA Sports NBA 2015

EA Sports NBA 2015 is a video game that allows the user to automatically create a 3D avatar in the game. It uses a depth sensor to create a rigged, textured and animated avatar, but only usable in that specific game. While it was praised by gaming press for generating very lifelike avatars when it succeeded [2], wishing that this type of technology was more widespread, after it had been tested more thoroughly it also received a lot of negative press [3], claiming that it "produced monsters". These produced monsters were the results of when the algorithm failed to properly segment faces. Regardless of whether it failed due to user error or a poor segmenting algorithm there was no way for the user to fix the error other than starting the process all over from the beginning.

1.2.3 EA Sports Game Face

EA Sports also have an online system where a user can upload images to their website and generate an avatar to be used in their games that support the feature, such as FIFA Soccer 13 and Madden NFL 13.

A user can upload one photo from the front and optionally one from the side. There is no automatic segmentation, the user positions a few landmarks in the face manually.

Figure 3 Examle input for Game Face. The green crosses are the manually positioned landmarks.

Figure 4 EA Sports game face example. Note that the eyes, hair style and color has been selected manually in the application. Also note the lack of texturing on the sides of the face model.

1.2.4 Automated Avatar Creation for 3D Games

Automated Avatar Creation for 3D Games is a system constructed by Hogue, Gill and Jenkin described in [4]. It does not produce a usable avatar though as it is not skinned and thus cannot be animated. Using two cameras in a stereo configuration it generates a 3D point cloud and generates a mesh and a texture from it. The article describes a proof of concept rather than a complete system but its novel approach requiring no special hardware makes it worth mentioning.

Figure 5 Automated avatar creation for games example. Image was originally published in [4].

1.3 Capabilities and restrictions of existing systems

The systems mentioned previously all suffer from some technical limitation that prevents them from being widely used.

1.3.1 3D Avatar Store

3D Avatar Store, while producing very good results, needs a server farm to run. Even if you can afford this, an internet connection is needed to access it. We also cannot reproduce their algorithms as they are confidential at the moment of this writing.

1.3.2 NBA 2015

NBA 2015 requires a depth sensor to work and also the process is fully automatic and cannot be guided or aborted by the user. When it works the results can be very lifelike though.

1.3.3 EA Games Game Face

EA Games Game Face requires no special hardware but does not produce a texture for the whole face, even when supplied with images of both front and side views of the face. The 3D mesh also lacks in detail which is a result of the very few landmarks used for segmenting.

1.3.4 Automated Avatar Creation for 3D Games

The method described in Automated Avatar Creation for 3D Games would require many work hours by an artist to generate a finished usable avatar as it is not rigged or animated or actually even a finished model as the backside of the person is unfinished in their example. If the method was extended though, it would probably be well on par with other methods.

1.3.5 Comparison table

System name	Requires special hardware	User guided	Textures the whole head	Animated avatar
3D avatar store	Server farm – Internet connection	Yes	Yes	Yes
NBA 2015	Depth Sensor	No	Yes	Yes
EA Games Game Face	No	Yes	No	Yes
Automated Avatar Creation for Games	No	No	Yes	No

1.4 Problem description

In the preliminary research for this work no avatar creation system has been found that satisfies the following requirements:

- Requires no special hardware, preferably can be used with a low quality camera
- Is automated but can be user guided to avoid "monstrous faces"
- Textures the whole head
- Results in an animated avatar

The different methods of creating an avatar described above have different hardware requirements. EA Sports NBA 2015 is using a depth sensor like Xbox Kinect to assist in the creation of a textured 3D model of a face. Depth sensors are still not available to many users and are not commonplace technology in mobile devices yet. The most common depth sensors in home electronics today is likely with video games consoles like Xbox or PlayStation. If you need to use a virtual avatar for other purposes than games then there might be a need for a method to create said avatar without the need of specialized sensors, preferably needing only to use the built in camera in a mobile device or a web camera.

1.5 Purpose

We will examine the feasibility of creating a facial 3D model using the information that can be gathered from one photograph of the front view and one photograph of the side view of a person's face. The 3D model should represent the face in both color and shape as closely as possible. The 3D model should be rigged and animated, thus being directly usable in an application requiring a user avatar.

The system should be able to take images directly using built in camera or load images from file.

The system should be transparent enough that even during automated parts the user should be able to see what is going on and be able to abort or correct the process in order to avoid problems such as e.g. getting stuck on the wrong structures of a face or edges in the background.

The system should run locally on the device without the need to contact a server.

This work aims to establish a method to automatically create such an avatar meeting these requirements. We shall call this method *Selfie to Avatar* henceforth.

1.6 Research questions

The core of the problem is generating a model we can use for games or visualization from as little data as possible. This leads us to our first research question:

Can we create an animated and fully textured 3D face for a virtual avatar from one front view and one side view photo to such a quality that the result can be matched to that person?

If we are to use a low quality camera like the one built into a phone or a web camera, then we need to make sure that the quality of the finished result is good enough to be of value to the user. This leads us to the second research question:

How much does the quality of the results differ when using a low quality camera like a standard mobile device camera compared to a high quality one like a system camera?

Some problems that have been observed and pointed out by journalists with the advent of the EA Sports NBA 2015 avatar creation process was that it sometimes produces "monstrous results" and the users losing patience because the process takes too long or is simply too tedious. This leads us to our last research question:

Is it feasible to let the user guide the processes that are automated in order to avoid problems of the algorithm reaching convergence in bad local minima, thereby avoiding problems with "monstrous faces"?

1.7 Outline

Chapter 2: Theory - In this chapter we present the theory needed to understand the processes used. Active shape models, the graphics pipeline and skeletal animation are all explained briefly.

Chapter 3 - Methodology: In this chapter the individual parts of the algorithm are described. We explain how Active Shape Models are used to segment faces in images, how pixels are projected from those images to the 3D model and how the 3D model is adjusted to fit the shape of the subjects' faces.

Chapter 4 - Results and Discussion: In this chapter we present the results of both generated 3D models of subjects and the results of a survey that was used to measure the performance of the system. The results of individual parts of the algorithm steps are also presented and some weaknesses in these are pointed out.

Chapter 5 - Conclusion: In this chapter we reach some conclusions as to why the algorithm has different performance depending on subject gender as well as suggest the most important areas of improvement on the system.

Chapter 6 - Critique: In this chapter we point out some weaknesses of the way performance was tested and propose a possibly improved way to measure avatar likeness to subject.

2 Theory

The Selfie to Avatar method will build on a set of other methods that are well established in image analysis and computer graphics. In this section each of them are described briefly.

2.1 Active Shape Models

Active shape models (ASM) performs a local search in an image for a complex object that has some known statistical characteristics of shape and image structure. This search determines the shape of an object in an image, thus providing a segmentation. It was first described by Cootes, Taylor, Cooper and Graham in [5] with the primary purpose of segmenting biomedical images automatically. It was further described by Cootes in [6] and it is this article that has been the basis for the face segmentation part of *Selfie to Avatar*.

2.1.1 Shape mean and variance

The model is constructed by first manually segmenting a set of training images by placing landmarks in corresponding positions in each image. Each image's landmarks are then aligned to a common coordinate frame by finding the translation, rotation and scale that minimizes the pointwise error to the mean of all shapes. This is performed in an iterative process until a convergence of error is reached. This process is called Procrustes Analysis and is discussed briefly in [6] and in detail in [7].

Once every shape in every image is in a common coordinate frame a final mean shape and a covariance of point positions are calculated. By performing principal component analysis on the covariance matrix the most important modes of variance are determined. The most important modes are kept and the least important ones, containing mostly noise, are discarded. When a shape is then described as closely as possible using the modes that are kept, noise in that shape will be removed.

2.1.2 Edge search

When improving the fit of a shape in a given image better locations for landmarks are searched for along the shape's normal at each landmark. If a better fit for the land mark is found along the search normal the landmark is moved. What describes a good fit for a landmark may be as simple as the most prominent edge but for many cases such a naïve approach is not viable. Instead the image structure along a landmark need to be considered and such a structure may consist of several edges of different strength or of edge gradients.

To calculate a landmark profile, the mean edge along the normal of each landmark is calculated. Once the mean edge of each landmark has been calculated a covariance matrix is constructed. Searching for an optimal fit for a profile of a landmark then becomes a question of minimizing a cost function based on the statistical edge data for that particular landmark. This is best visualized with an example.

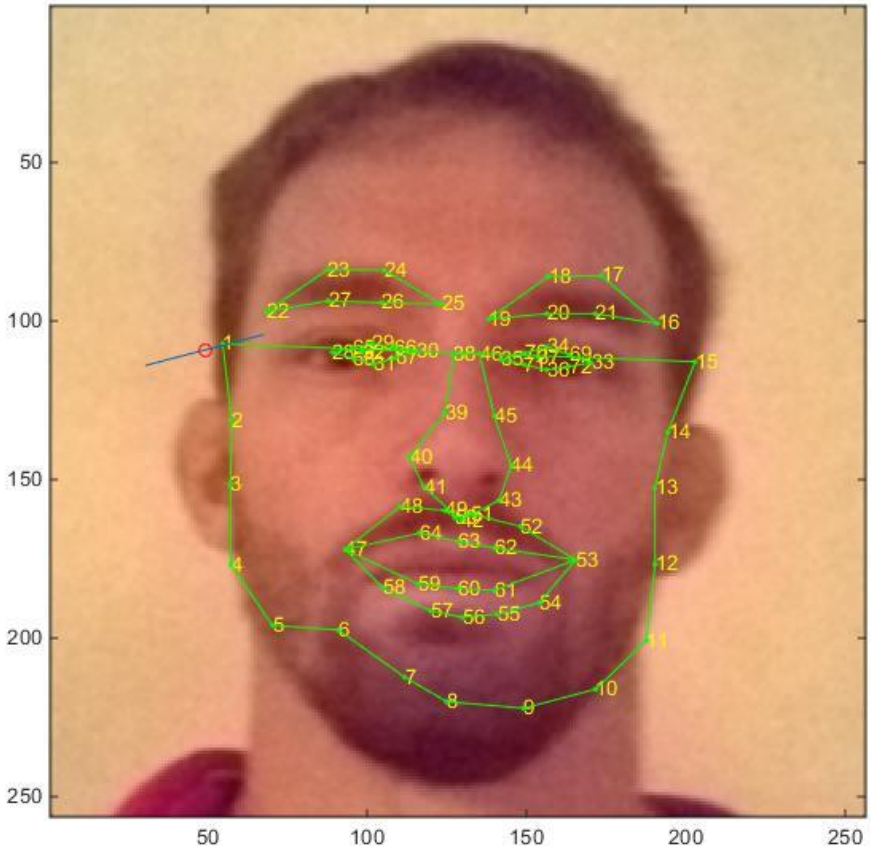


Figure 6 An example of an ongoing search along a normal in point 1.

Figure 7 Pixel value samples along normal.

Figure 8 Absolute derivatives of pixel samples along normal.

Figure 9 The edge value profile to look for along sampled normal.

Figure 10 Edge search result. The minimum cost position of the mean edge profile found.

For this example, we are considering only the search along the normal for point 1 in Figure 6. The pixels are sampled along the normal, resulting in an array of pixel values shown in Figure 7. The absolute derivatives of these pixels are calculated and compared to the mean derivatives of that particular landmark in all the shapes we have used in training. When doing this comparison, not only the mean is considered but also the variance of each pixel derivative to calculate a cost using the Mahalanobis distance function, further described in [6]. In Figure 10, the search has found a local point that generates a minimum error as its edge profile is the most similar to the mean. In Figure 6, the old point, displayed as a red circle, has been moved to a new point corresponding to that local minimum as can be seen in the green shape.

This type of search is a bit more complex than e.g. simply selecting the strongest edge value along a normal but allows us to take more of the image structure into account in the search. This is important when not all landmarks will be positioned at the locally strongest edge. This is the exact case of a face where the eyebrows may have weaker edges than the eyes or there might be some noise like beards that produce "false positive edges".

In one iteration of searching for a better fit all landmarks normal are sampled, looking for points with a lower cost (i.e. a shorter Mahalanobis distance). Once all points have been set, this new shape is projected onto the eigenspace determined by the available modes of the shape. This projection will get rid of outliers that are inconsistent with the shape that has been trained.

This search continues until some stop criteria has been reached. These stop criteria could be that the shape has to improve by some set amount or, as Cootes describes it in [6], when a certain percentage of the landmarks remain within 50% or the normal length.

2.1.3 Image scale space pyramid

An image scale space pyramid is a data structure where we create copies of an image with less and less detail in them. We use these copies with less detail in them when we look for coarser structure in the image and do not want to be disturbed by noise.

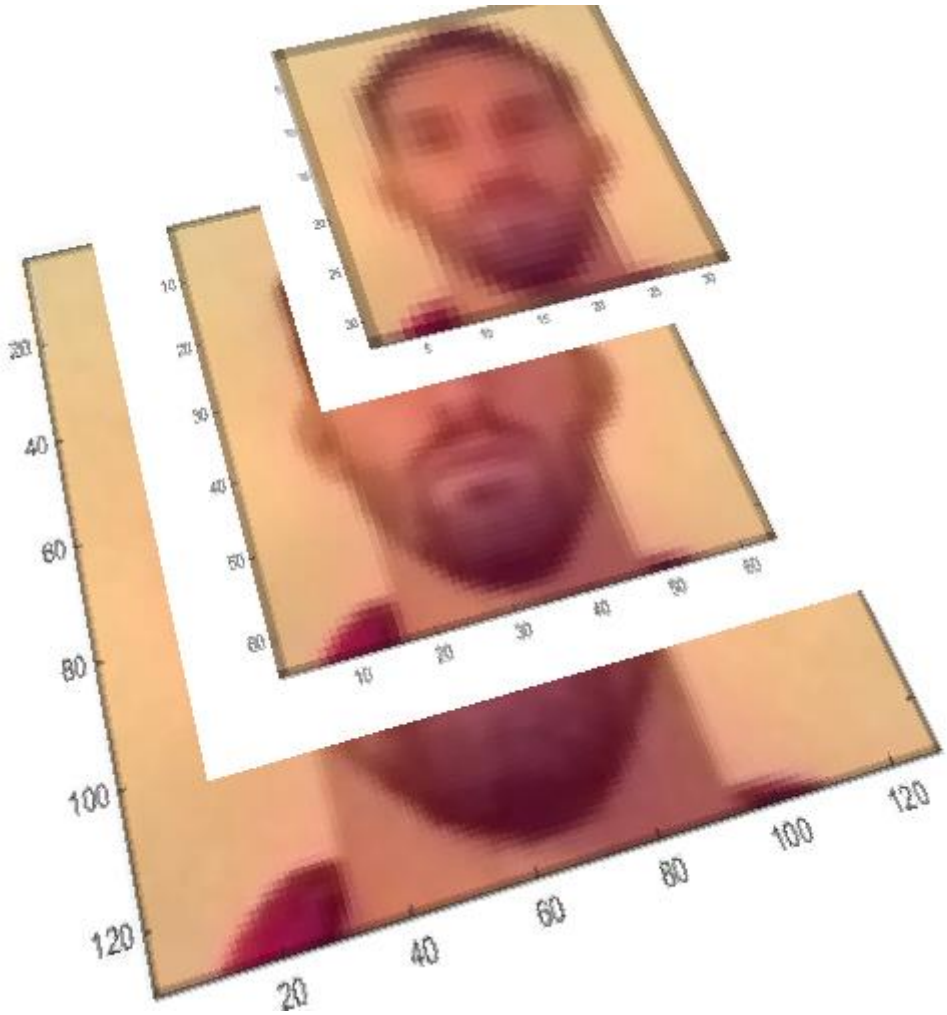


Figure 11 An image scale space pyramid with three levels: 128 by 128, 64 by 64, and 32 by 32 pixels. At the top layer we can only discern the most prominent structures of the image.

In order to reduce the risk of running into local maxima in an image and to speed up the execution time we use an image scale space pyramid. The algorithm can then be used with a few iterations in each scale space, starting at the lowest resolution, working from course to fine detail. In the lower resolution images, the shape is likely to move large distances in each iteration and then move shorter and shorter distances as resolution gets finer.

2.1.4 Principal Component Analysis

Principal Component Analysis (PCA) can be used to find the most important dimensions of change of a set of points in an arbitrary space. If we consider a simple example think of a set of points in 3D space positioned according to some distribution which mechanics may not be fully known by us. As we can determine visually by inspecting the image the main variance of the point's positions is in one axis, marked by a line. If we were to describe the exact position of a point we would need 3 values of data, for example the coordinates along the x, y and z axes. We could however, describe each point's position as a coordinate along the line t in the image. This would not give us the exact position but it

would be an approximation that might be good enough for our purposes. If we estimate that the measurement of the point positions also contains some error then an estimation like this might actually have a purpose by eliminating some of that noise, especially if we have a model understanding that tells us that the points actually should be positioned along a line.

Given a set of points we can calculate their mean and covariance matrix. If we calculate the eigenvalues and eigenvectors of this covariance matrix then the eigenvector belonging to the largest eigenvalue will be describing the most important dimensions of variance, the second largest eigenvalue's eigenvector the second most important and so on. The direction of each of these axes in the data will be given as an eigenvector. Each eigenvector's corresponding eigenvalue will also tell us how much of the total information (if compared to the total of all eigenvalues) in the training dataset is described using that particular eigenvector. This makes it possible for us to select such a number of eigenvectors that we describe a percentage of the information, e.g. for a 100 dimensional problem we may need 17 eigenvectors (17 dimensions) to describe 95% of the information. In such a case the last 83 dimensions will contain mostly noise.

2.2 Graphics pipeline

The graphical pipeline of a rendering engine is an advanced concept and will not be explained in full here. Some simplifications will be made and only those parts of the pipeline that are absolutely critical to the method will be explained, and then only briefly. For a more complete description of the graphics pipeline see [8].

2.2.1 Shader program

A shader program is a program that runs on the Graphical Processing Unit (GPU). It is generally highly parallelizable and can run, at least in theory, simultaneously for many vertices or many pixel fragments. Since a GPU may contain orders of magnitudes more cores for floating point precision operations than a CPU it is often better for performance if calculations are moved from the CPU to the GPU if these are used for rendering only and not the rest of the logic of a program.

Shader programs can be run in different parts of the graphics pipeline so one shader program could process vertices, a vertex shader, and another one could process pixels, a pixel shader (also called fragments in Open GL).

2.2.2 Vertex

A vertex is used to describe a point in space. A vertex can also contain data such as texture coordinates, normal, vertex colors and any other data that is needed for shader programming. A vertex will not be rendered on its own, it needs to be part of a triangle for the GPU to process it in any meaningful way. A vertex can share a location with other vertices.

2.2.3 Triangle

A triangle is a set of three vertex indices of vertices that describe a surface that should be rendered. Triangles can share vertices. All vertex data is interpolated over the triangle, be it normals, vertex colors or any other arbitrary data.

When a pixel is rendered, its barycentric coordinates of the triangle will be used to interpolate what the normal, vertex color etc. is in that particular point. Every barycentric coordinate will determine the weight of its corresponding corner vertex when calculating the data value for that particular pixel. Thus there is a smooth transition over the triangle for all data. This will be used in order to smoothly blend two textures in the target shapes in this work.

Note that the triangle itself contains no data, it will only interpolate the data of its vertices. Thus if a triangle should have a flat surface i.e. a consistent normal then all its three vertices need to have the same normal.

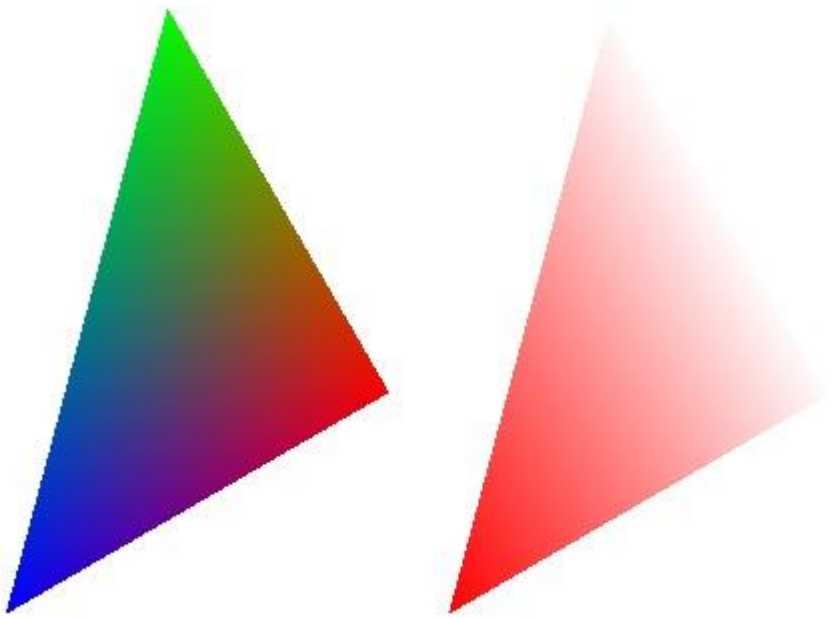


Figure 12 Two examples of triangles where the pixel colors are interpolated over the triangle based on barycentric coordinates. The vertices of the left triangle have their vertex colors set as opaque red, green and blue colors. The triangle to the right has all vertex colors set to red, but two of the vertices have their alpha value set to 0, i.e. fully transparent.

2.2.4 Mesh

A mesh is a set of vertices and triangles that represent a coherent entity. We often refer to it as a 3D model.

2.2.5 UV coordinates

In order to render a texture on a triangle one needs to know how the triangle should be positioned in the texture. This is done by setting the coordinate of each vertex in the texture space. These

coordinates are called UV-coordinates. To determine the color of an individual pixel these UV-coordinates are determined by barycentric interpolation and the result is looked up in the texture. Note that this is a somewhat simplified explanation which is explained in detail in [8].

2.2.6 Render target

A GPU may instead of rendering directly to a screen, render pixels to a texture instead. This texture can then be used for further processing, even be used as a texture in a scene and will then work like a screen in the scene showing what that camera is currently viewing.

2.3 Skeletal Animation

In computer graphics a 4x4 matrix is often used to describe the rotation scale and position of an object. These matrices have the advantage that they can be used in chains to describe an object hierarchy. Such a hierarchy might be used to describe our solar system with the sun as the parent, the Earth a child of the Sun and the Moon a child of the Earth. The transform matrix of the Moon in this case will describe the Moon's rotation, scale and position in relation to the Earth. If the Earth moves, then the Moon will move with it retaining its position in relation to its parent. Meanwhile the Moon will continue its rotation about the Earth. This type of hierarchy is described in detail in [8].

A similar hierarchy can be used to describe the human body. We can consider the hip bone to be the parent of the spine, the spine the parent of the head, the head the parent of the nose etc. This approach is commonly used when animating a 3D model. Every entity in the hierarchy is then called a bone, but that does not mean that each of them represent an actual bone in the body nor that every skeletal bone has a unique hierarchical bone representing it. In that case when the model is moved that is applied to the parent bone (often the hip bone). In an animation when a bone is rotated or moved each of the children will follow suit. In one 3D model a vertex can be assigned to one or several bones. In the case of several bones each bone is assigned a weight relative to the others. The position of the vertex will be interpolated based on the transforms of its assigned bones and their corresponding weights. This is part of the vertex data that will be considered in the shader programs used to display the animated model, and those shader programs will calculate the final position of the vertex based on the animation that is being played.

Such bones can have other uses than animations though, and it is these alternative uses that we will use in this work. Consider the vertices belonging to the nose of a 3D model of a face. Assume that the nose is represented by a bone of its own. If we want a nose that is wider then we can apply a scale to the bone against the axis that corresponds to the nose width, increasing the nose size along that axis. If we want the nose to point out more then we can rotate the bone a little bit to achieve the transformation that we want.

Note that in some literature and in many rendering engines a mesh with bone structure for skeletal animation is called a *skinned mesh* or a *rigged mesh.* While the subject is touched upon in [8], for more detail see Eberly's explanation [9], especially the chapter *Skin and Bones.*

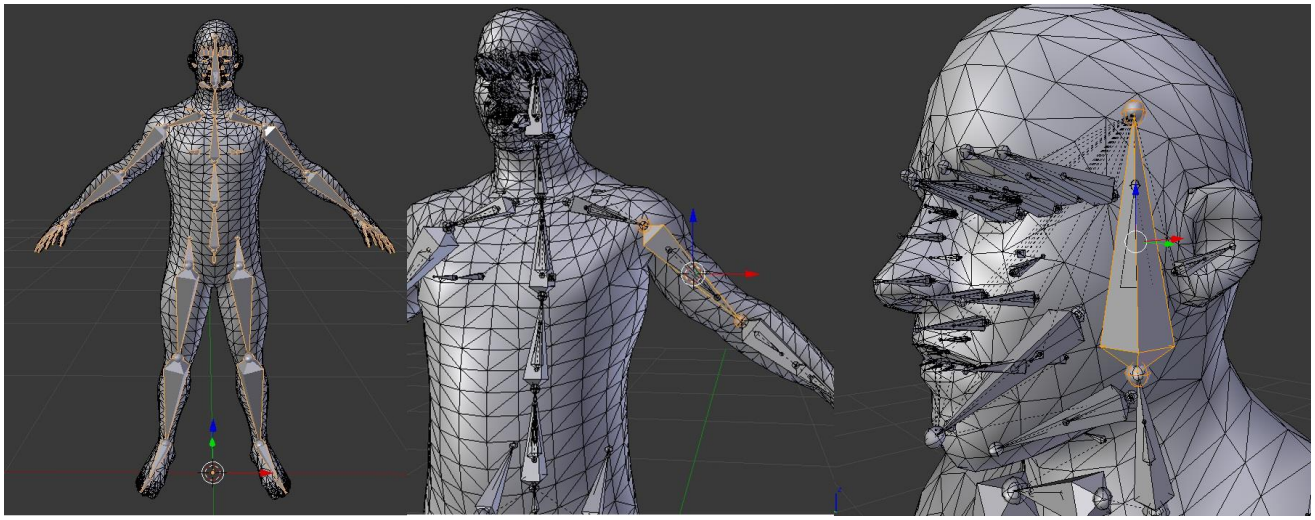


Figure 13 Left: The full bone structure of a human male body. Middle: The bone for the upper arm selected. Right: The bone structure of the face.

3 Methodology

In this chapter we will describe the algorithm we use in *Selfie to Avatar*. We first explain how it works in general by presenting an overview of the steps of the algorithm. We then explain each step in detail. We start by describing how ASM are constructed and used. We then proceed to describe how the segmented images are projected onto a texture. Finally, we show how the 3D face model is reconstructed.

3.1 Overview

Figure 14 describes in a simplified way the process that we propose for reconstructing a 3D avatar of a face. First two source images are used, taken from perpendicular angles. In each of these photos ASM are used to segment the face, as can be seen in the green outline. The red outline is the mean shape. These segmentations are then projected and merged onto a render texture. This render texture is then used to texture a 3D model of a face. The shape of the face is adjusted based on the eigenspace coordinates of the two segmentations that were determined in the original photos. The user is then prompted to make final adjustments to the 3D shape by moving sliders each representing some variation of shape in the face, like the nose width or chin size. Finally, the user can set eye color and clothing of the avatar.

Figure 14 Visualization of the process start to finish. Top left: Segmented front view image. Top right: Segmented side view image. Middle left: The original face texture. Middle right: Face texture after projecting segmented photos. Bottom: Two views of the finished avatar.

3.2 Building and using Active Shape Model

Finding a comprehensive database for training both front and side view ASM proved a difficult task. MUCT Face Database [10] was used for front view images, the Siblings Database [11] was used for font and side view images. Photos collected from volunteers also proved to be very important as they could be directed to have e.g. pose, lighting conditions and varying image qualities suited for training the models to handle such occurrences.

3.2.1 Train two shapes, front view and side view of a face

There are two shape models used, one for photos of the front of a face and one for photos of the side of a face. Each of these models need to be trained by manually segmenting a large set of images. In this project we used an approach where the model could increase its performance as more photos were added. This meant that initially the photos had to be segmented entirely manually but with every photo that was segmented and added to the database the model would increase its performance a little bit. After approximately 30 photos the algorithm could be used to assist the manual labor by first running an automatic segmentation and then improving on it manually.

Figure 15 Images of manually segmented front views. Green is the actual segmentation and red is the Procrustes transformed mean shape.

Figure 16 Images of manually segmented side views.

Manually segmenting images is extremely time consuming. The face frontal view shape in this work used 139 landmarks and the side view shape used 119 so a manual segmentation means at least that many manual operations. To facilitate manual segmentation some tools were created so that the whole shape could be translated, rotated and scaled by dragging landmarks. The same was done for groups of landmarks like eyes, nose, mouth and so on which reduced the manual labor extensively.

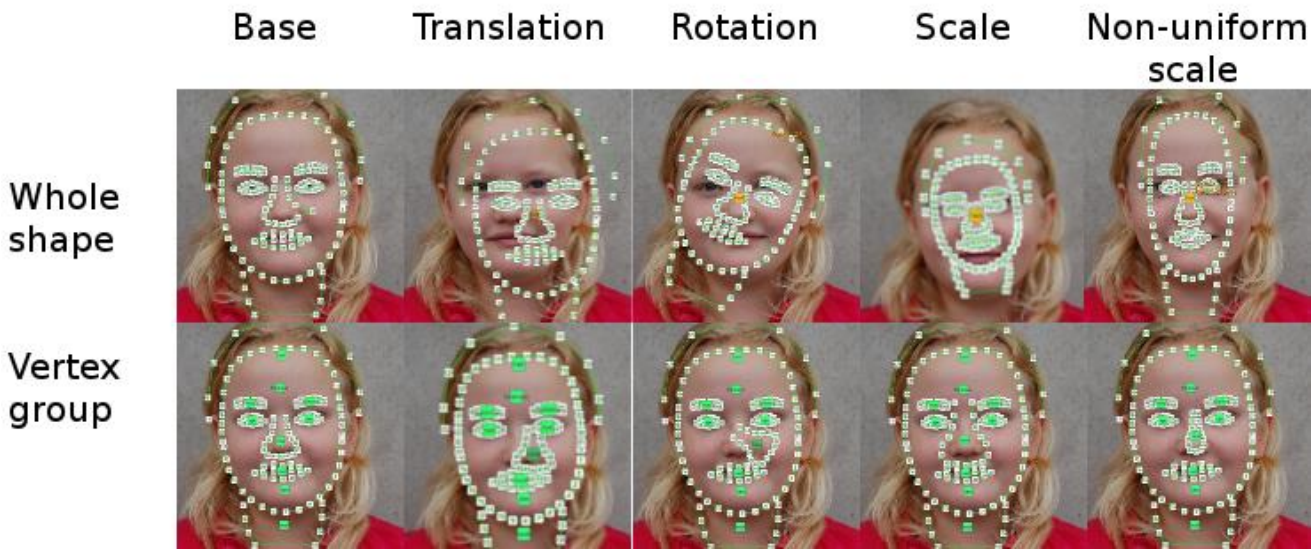


Figure 17 Visualization of tools used for manual segmentation. Note that the same tools can be utilized by the user in the end product to perform manual fine tuning of segmentation results, or to guide the process during runtime search.

As incremental learning was used the principal modes changed slightly as the work progressed but changes generally get smaller and smaller as the model gets better and better. Once the changes become minute it is possible to determine what effect an eigenvector will have in describing the shape. Once reaching this stage it is possible to proceed to link the eigenspace coordinates to bone transforms in the 3D model as described in section 3.2.5.

Even though the eigenvectors may change very little with further training, it is important to note that the direction of an eigenvector can switch to its diametrical opposite whenever a new entry is added to the training database. In case eigenvectors are being used for other applications (like in this work, section 3.2.5, where it is used to describe also adjustments to the 3D model) then it is important to keep track of the old and switch signs of entries if it switched directions. If this issue is handled, then we can let the application use every segmented image from every user in the future as a training sample which allows for continual improvement over time as the application is used as long as we can make minor manual adjustments in case the eigenvectors change slightly over time. These changes should however converge to some "global mean" though and then the need for such fine tuning should be reduced.

The final mode variations after PCA of the training data are as displayed in Figure 18 left side and Figure 19 left side.

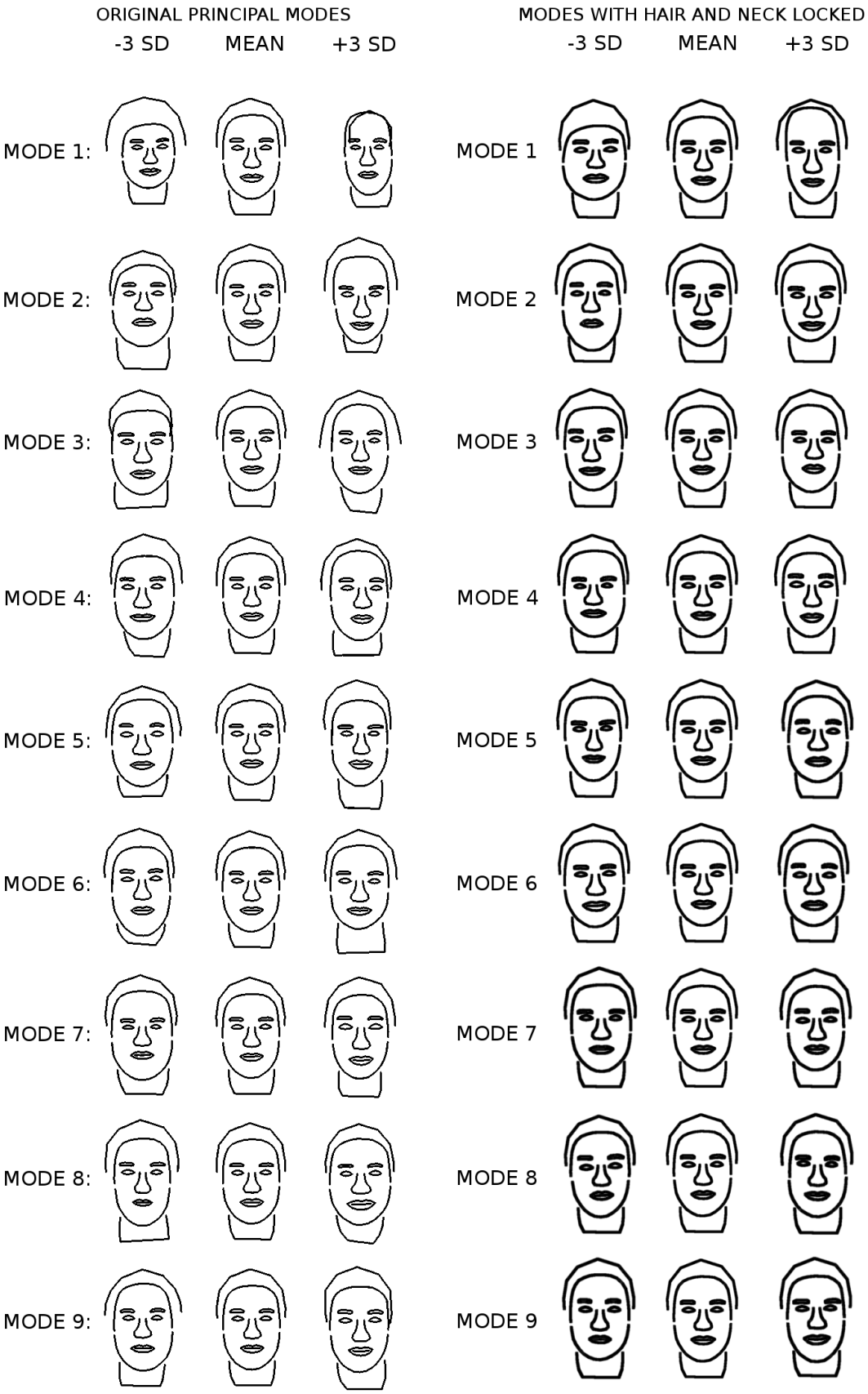


Figure 18 Left: The first 9 principal modes of shape variations. Right: The first 9 principal modes of shape variations with hair and neck locked

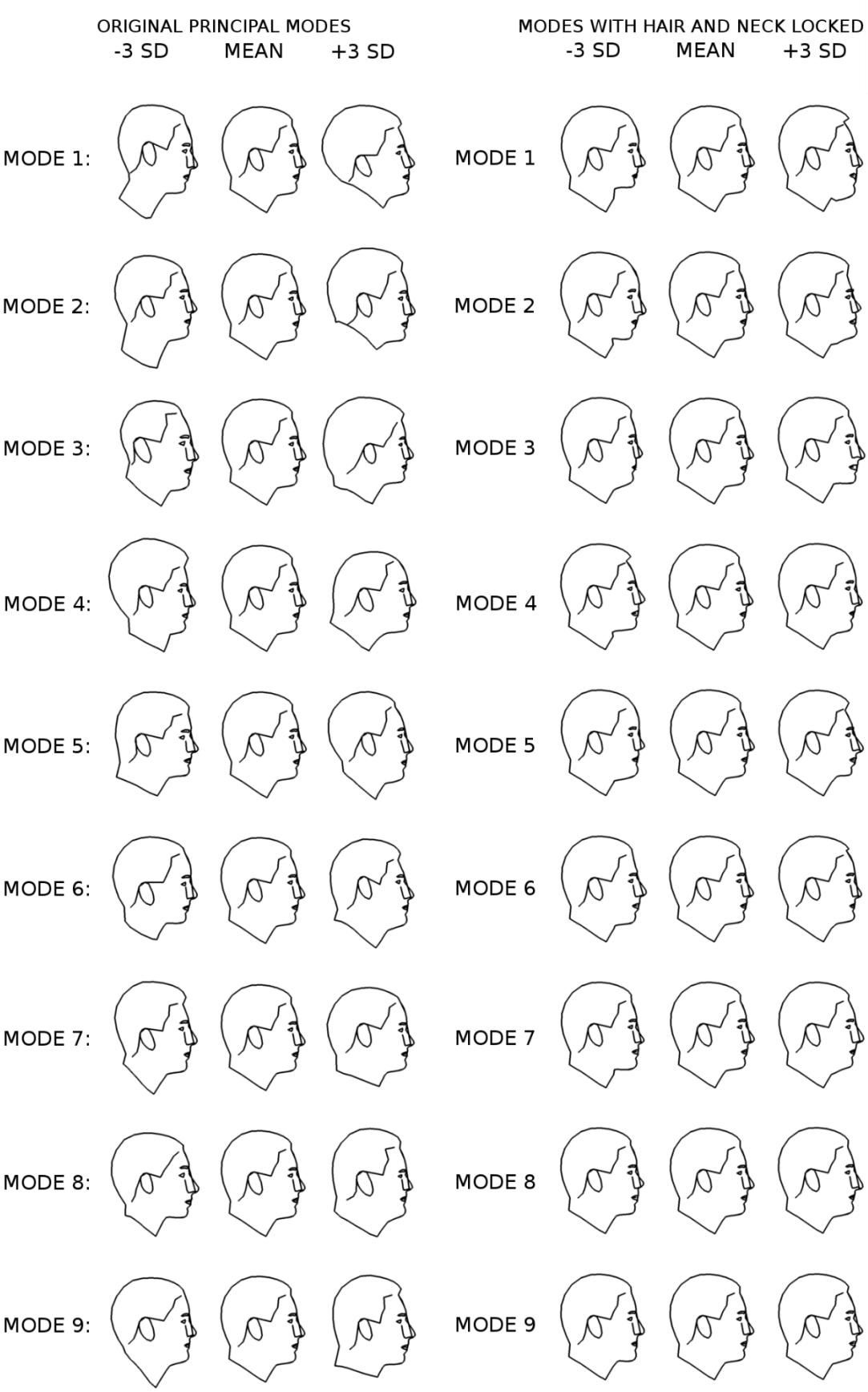


Figure 19 Left: The first 9 principal modes of shape variations. Right: The first 9 principal modes of shape variations with hair and neck locked

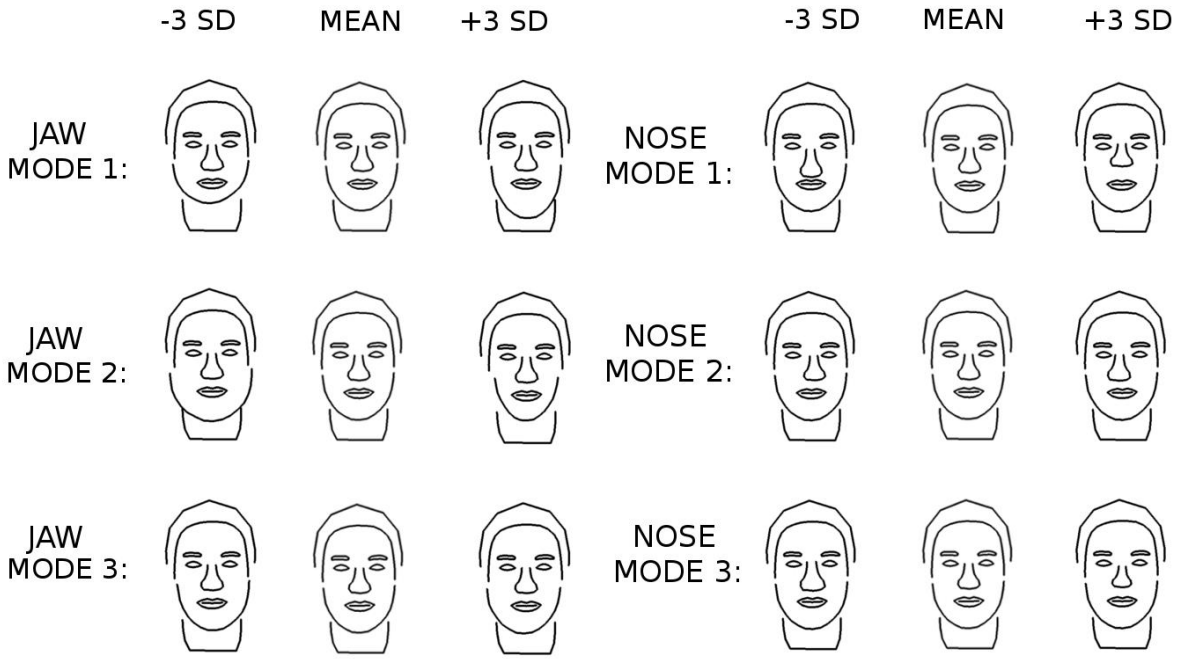


Figure 20 Left: The first three principal modes of jaw variations when all other vertices have been locked to the mean value. Right: The first three principal modes of nose variations .

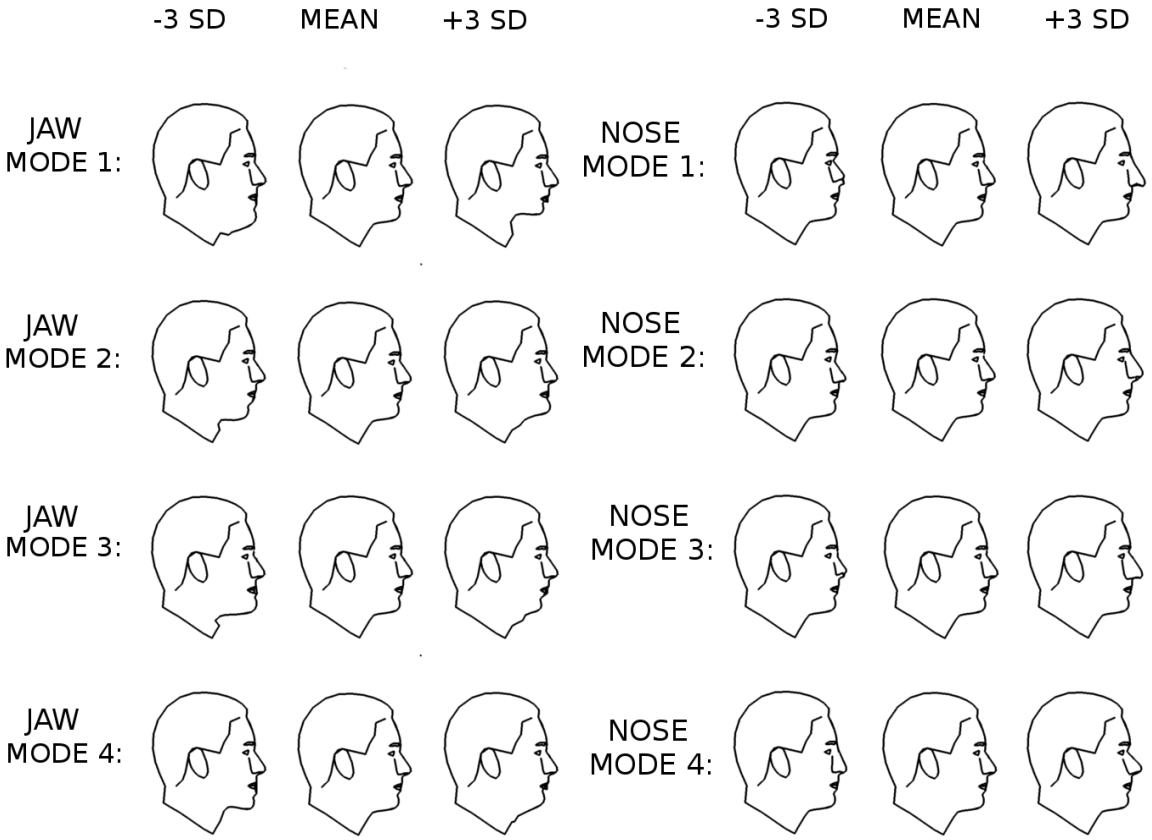


Figure 21 Left: The first four principal modes of jaw variations when all other vertices have been locked to the mean value. Right: The first four principal modes of nose variations.

In order to segment out the hair as well as face the shape had to accommodate for different hairstyles. This is not a good use of the ASM. The ASM, as Cootes describes it, are *not necessarily appropriate for object with widely varying shapes (e.g. amorphous things, trees, long wiggly worms etc.)* [6] which unfortunately applies for hairstyles. This results in the principal modes of the face model to represent mainly the variance of hairstyles which is not something we have any use for when trying to describe the 3D characteristics of a face. This can be seen in the first three modes in the left column of Figure 18. The modes describing important shape variations of the actual face do not really start to appear until mode four through nine in both front and side view shapes. As each mode will contain less and less information it can be very hard to identify which part of the face to adjust based on the modes in the left columns of Figure 18 and Figure 19. In order to make it a bit easier we lock all vertices that we do not find critical to shape, such as hair and neck vertices, to the mean and study the shape variations of those in the right columns. This makes it easier to determine shape variations. As an example one can study mode 7 of the right column of the side view shape. In that mode it is easy to determine through inspection that the coordinate of that mode can be used to set the size of the nose.

Taking this approach even further we locked all vertices except of those relating to one face part, e.g. the jaw or the nose. As can be seen in Figure 20 and Figure 21, only three to four modes are needed in that case and the variations give much clearer indications of shape variance. The use of the coordinate for each mode is further described in section 3.2.5.

Note that we still need the hair to be segmented as it is an important feature of the texture of a head and for this segmentation we use the modes with all vertices "unlocked". The modes with "locked" vertices are only used in the 3D face reconstruction.

3.2.2 Run ASM on both images

When the ASM are trained they can be used to automatically segment faces in images. The ASM are slightly prone to reaching false local minima, especially if the background contains too many edges, so an approximation is best set manually by dragging, scaling and rotating the shape in place. An alternative to this is to automatically find a good start criteria using some other method. One such possible method would be the Viola-Jones algorithm which is very good at finding the general location of faces in images [12].

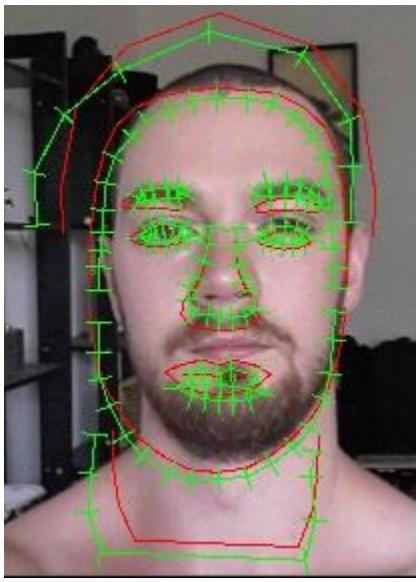


Figure 22 Example of the ASM "getting stuck" on edge structures in the background of an image. Note that this also distorts the rest of the shape as the mean shape (displayed in red) is not correctly positioned.

Even when properly trained the ASM may still fail spectacularly and find structures in the image that do not belong to the object that the user wants to segment and instead "get stuck" on objects in the background that have strong edges, thereby generating local error minima. In order to avoid such incidents, the implementation in this work was made to be entirely transparent to the user and will show exactly where it is searching in real time. If the user sees the algorithm diverging in its search she can easily adjust it by dragging one or several vertices to the correct position using the shape control points as described in Figure 17 even while the search is still running.

Figure 23 Images of automatically segmented front views.

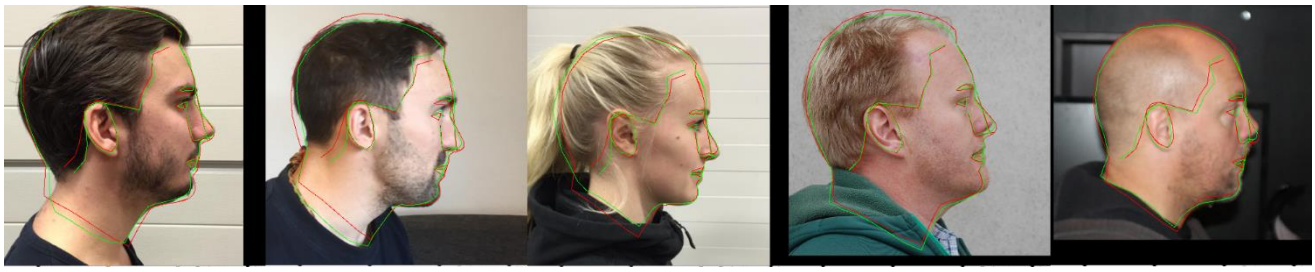


Figure 24 Images of automatically segmented side views.

Once the automatic segmenting is done the user has the option of adjusting the results manually if needed. Most of the time this is not needed but in Figure 23 the leftmost face has converged

incorrectly so that the positions of the eyes, nose and mouth need to be adjusted. Using the tools described in section Figure 17 this normally takes no more than a few seconds in the case where it is needed. During these adjustments there is also a view of a 3D-model of the face that will update changes in real time giving instant feedback of changes made.

3.2.3 Setup target shapes in texture

After segmenting the two images we know what image data we need to use in the final texture. In order to project this data to the correct texture location we create two target shapes in the texture. We position each landmark at the corresponding position in the target texture. Next, Delaunay triangulation [13] is used to calculate a mesh for each of the target shapes.

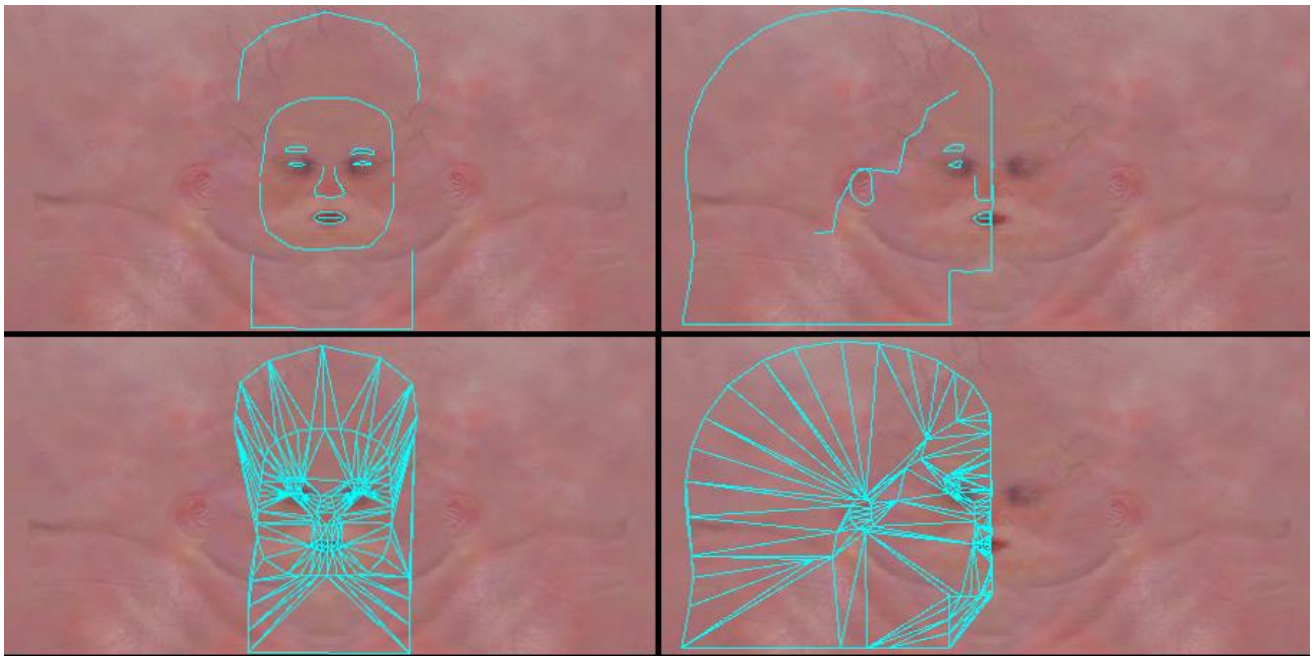


Figure 25 Top row: Target shapes positioned in the texture. Bottom row: Target shapes as triangulated meshes

3.2.4 Project pixel values from segmented source shape in image to target shape in texture

What needs to be done next is to project every triangle from the source image to the target texture. A naïve approach would be to, for every pixel in the target texture, look up the corresponding location in the source image and sample it. This approach was tested in a prototype version and worked but had a few drawbacks. It took a lot of time, ranging from 2-10 seconds to complete a 2k x 1k texture. It also introduced artifacts due to floating point precision errors that would require special solutions which would require even longer execution times.

A more suitable, albeit a bit more complex, solution is then to set the two images as textures for the two meshes we constructed previously and setting the UV-coordinates of each vertex to the position of the corresponding landmark in the source image. We then have two models in the scene, one for each target shape, that is textured using the original photos. If we position these target shapes in a scene and view them using a camera with orthographic projection and render the camera result to a target

texture, then we have a texture we can use on the face 3D model. Not only will this resulting texture be rid of the visual artifacts that the simpler solution suffered from, it also renders at 60 frames per second, letting us give the user real time feedback of the final results from image segmentation directly on the avatar.

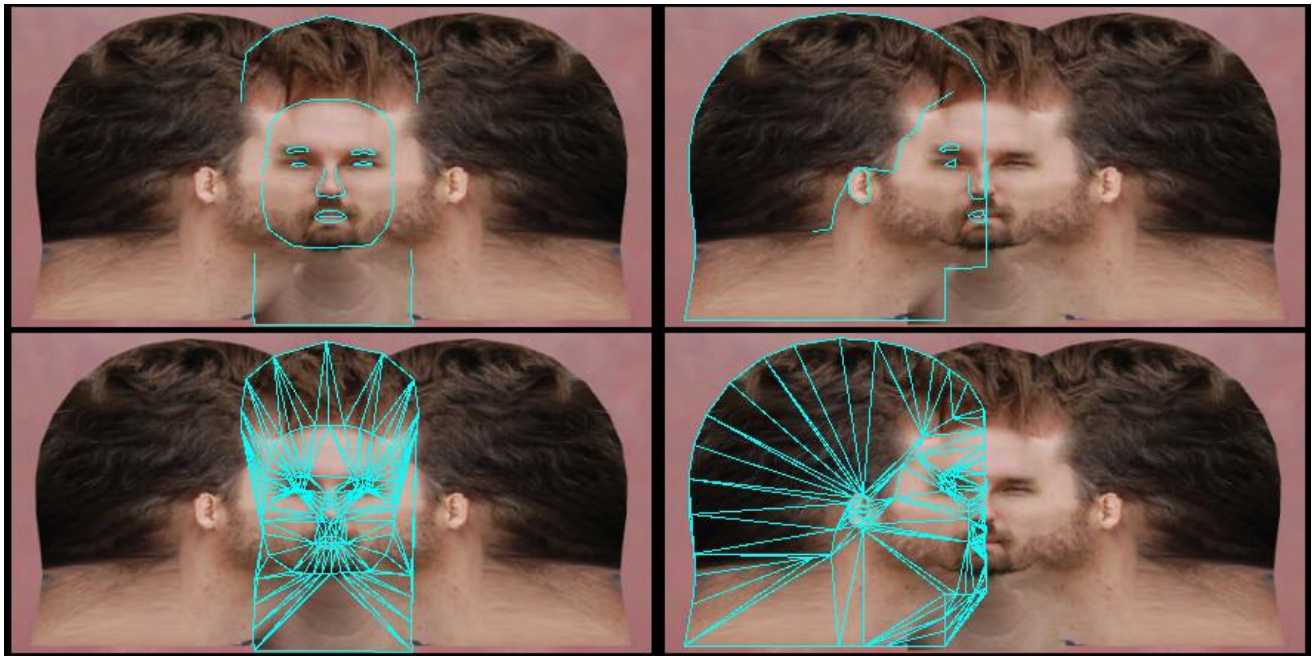


Figure 26 Top row: Target shapes positioned in the texture. Bottom row: Target shapes as triangulated meshes

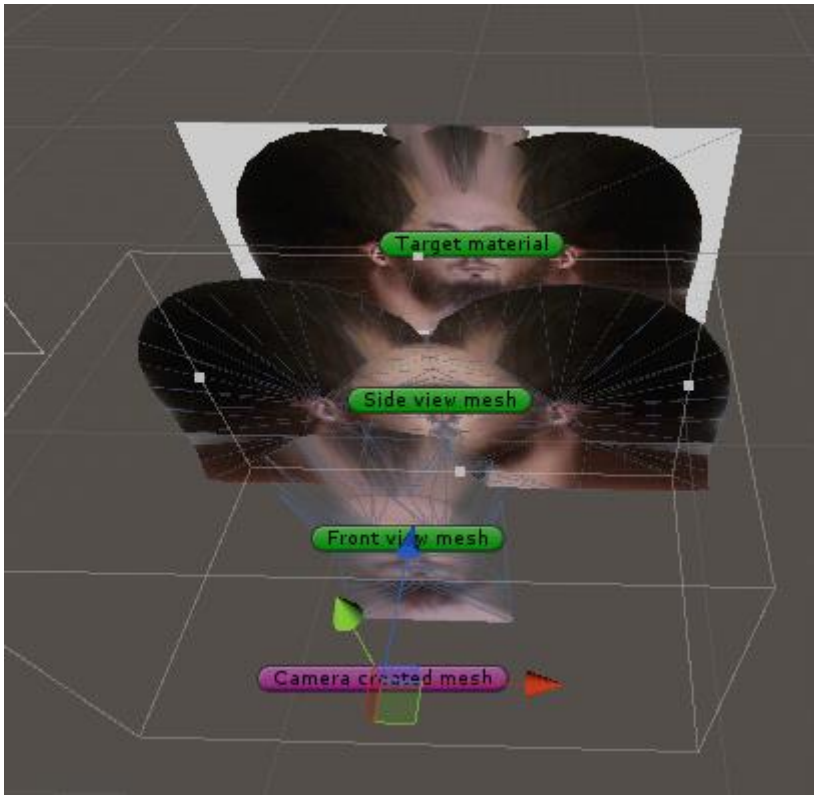


Figure 27 Visualization of the render-to-texture setup. The grey box shows the view frustum of the orthographic camera which renders the Front view mesh *in front of the* Side view mesh. *The camera renders this to a texture which in turn is*

displayed on Target material. *Note that the* Front view mesh *is fully transparent at its edges due to the edge vertices being set to transparent vertex color.*

Two minor details need to be addressed. First, we only use one photo from the side, although the system could easily be extended to use another one, so we need to duplicate the mesh that corresponds to the side view segmentation and mirror it along the center of the face. This is trivial in a rendering engine though. Second, as can be seen in the images below, the two shapes intersect. We solve this by setting the rim vertices of the front view mesh to have a vertex color that is entirely transparent. As has been explained in section 2.2.3, and displayed in Figure 12, this will lead to an interpolated transparency of each of the rim triangles that will make the two source images blend nicely in the final result. If we do not handle this issue, there will be a very noticeable edge in the transition between the two target shapes.

Figure 28 A resulting texture with input images that had different lighting conditions. Note the poor result when transparency is not interpolated along edge triangles of the front view image.

Figure 29 Image with alpha values of edge triangles interpolated. Note that even though the source images had different lighting conditions the result produced is acceptable.

Figure 30 The final resulting texture. Note the blending of the two source images. Note also the difference in quality compared to Figure 29 due to input images with uniform lighting conditions in this case.

3.2.5 Setup eigenspace coordinate connections to bone hierarchy

After projecting the correct texture to a 3D model, the coloring is done. This gives an illusion of shape as can be seen in Figure 33, where an observer will often falsely think she notices a change of 3D structure. In fact, all of the male subjects in Figure 33 have exactly the same 3D structure which may not be apparent at first visual inspection.

As the texture comes from a real world photo the texture actually contains some information of minute changes in 3D structure in the real world resulting in small shadows in wrinkles and dimples. We still need to adjust larger structures such as the shape of the jaw, nose, forehead etc. though.

In the segmentation step the coordinates of the shape eigenspace were calculated and the coordinate of each eigenvector in this space can be used to discern some of the structural information of the face.

It is important to note that as we have only two 2D projections of the real 3D data to work with (the photos) there might be some errors in our data. E.g. if a subject leans her head back slightly the head shape will be considered more round and the jaw more distinct. If we had a 3D point cloud or if we did pose reconstruction this problem could be rid of entirely but for now we will have to leave the corrections of such errors to the user in the last step.

In section 3.2.2 we presented the problems of discerning distinct face part changes in the modes of the entire face. Thus we construct the modes that describe changes of one face part at a time. We set up each mode's variation to affect different parts of the bone hierarchy of the face 3D model. We do this by visually inspecting each mode and how each mode variation should affect each bone transform manually. See Figure 31 and Figure 32 for examples of how the jaw and the nose vary in shape in the 3D model based on the variations of each mode. We also do the same for the mouth, eyebrows, eyes and forehead. In the Figure 32 we can see that mode 1, 2 and 3 have a profound effect on the nose scale, mode 1 and 4 have an effect on the rotation and 1 and 2 also affect the position. These need to be manually set up for the 3D face models being used and may differ based on how the bones are

defined in each 3D model. In our example we have one model for male subjects and one for female and each of these have slightly different bone setups, even if the bone hierarchy is the same.

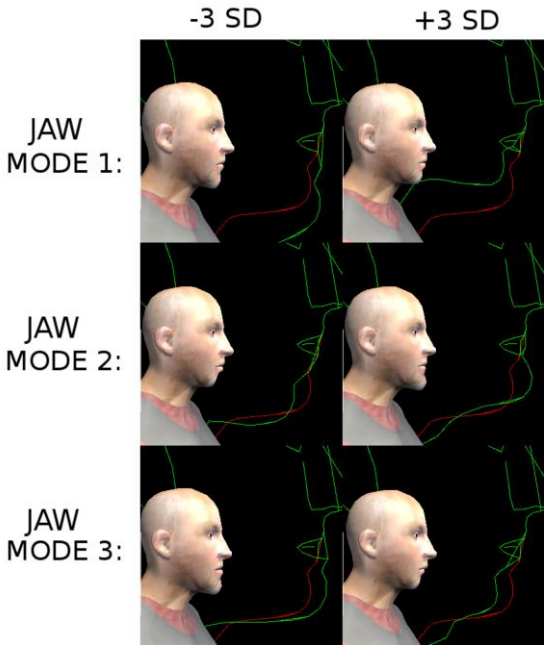


Figure 31 The first three modes of variance of the jaw. The mean shape is displayed in red color and the mode variations are displayed in green.

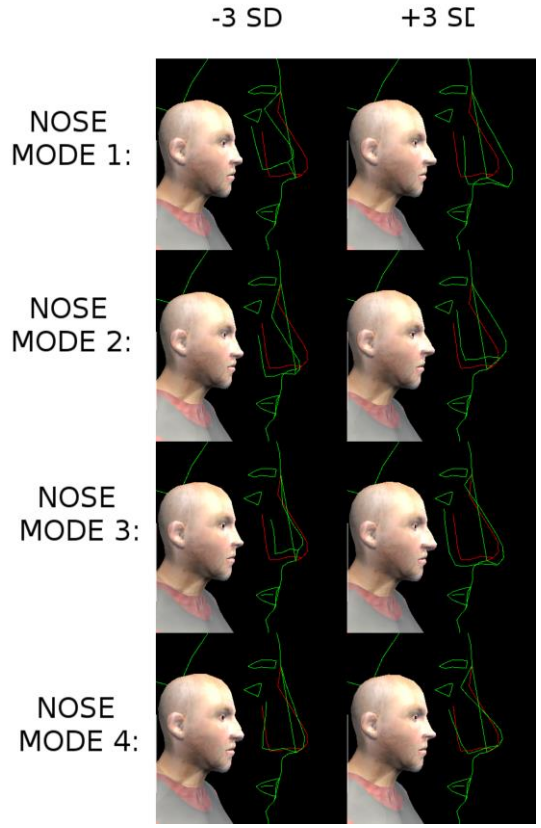


Figure 32 The first four modes of variance of the nose. For most face parts we use no more than three modes but the nose 4th mode contained, as can be seen, important information about how the nose "leans".

3.2.6 Use eigenspace coordinates to set proper bone transforms in skinned model

As the eigenspace coordinates are set up to adjust bone transforms we can then apply these to the 3D model. Compare Figure 33 where the model is only textured and Figure 34 where the model has also had the bone hierarchy adjusted based on eigenspace coordinates. In most cases the applied texture is enough to recognize the subject but some people have more distinct facial features that need to be handled by actually adjusting the 3D structure. The leftmost subject in Figure 33 and Figure 34 is a good example of this as he needs to have his jaw made a bit squarer.

Figure 33 Texture applied to the base model. Note the illusion of shape change between the male models, even though they are exactly the same. This gives a clue as to how much shape information is actually relayed through texture. The pose varies slightly in the different images.

Figure 34 eigenspace coordinates applied to adjust the base model. Note in the fourth subject how the nose has been made too thin. This is an example of errors that can occur that the user needs to adjust manually.

3.2.7 Allow user to do final adjustments of bone transforms

As some errors are likely to occur some manual adjustments needs to be done in some cases. In the final adjustment the color of the eyes and clothing can also be set. The user interface for the manual adjustment consists of sliders representing change of bone transforms of one dimension each, translated into a readable state like “nose height” or “lip size”.

Figure 35 Final manual adjustments made. The nose in the fourth subject has been fixed.

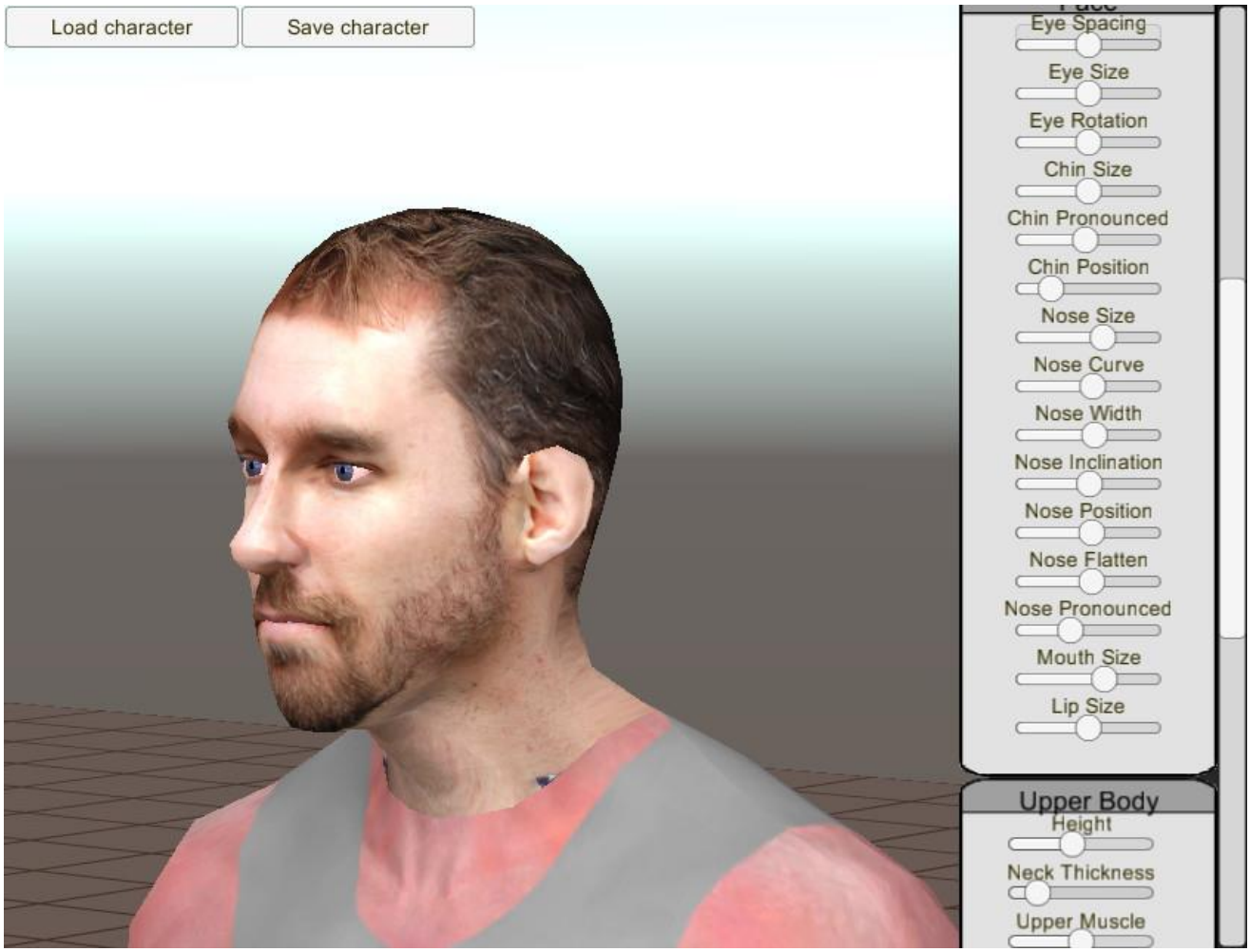


Figure 36 On the right are the sliders available to the user. Note that bone transforms have been translated to more easily understandable names.

3.2.8 Save data in a format reusable by games or other applications

As the user finishes the construction of the face it should be saved in a format reusable in different games and applications. Optimally this would be in a database that each application can connect to so that the user only needs to perform the segmenting and modeling once.

Figure 37 An example of an avatar used in a game.

4 Results and Discussion

This chapter is divided into two sections. First there is an evaluation of the performance of the method as a whole where the system is tested using an online survey. Following that there is an evaluation of each of the individual parts of the system, from the segmenting to the model reconstruction.

4.1 Final result

The final result is a 3D model of a face that can be recognized as the same person in 81.7% of the cases.

20 examples of finished avatars and corresponding real life photos are displayed below:

Figure 38 Comparison of a photo of each person and their corresponding avatar below them for 10 females of varying ages.

Figure 39 Comparison of a photo of each person and their corresponding avatar below them for 10 males of varying ages.

In order to verify our results an online form was constructed where respondents were asked to try to match the above photos of the 10 male and 10 female test subjects above to their corresponding avatar. A total of 352 people finished the test, each answering 20 questions. It was possible to leave an answer blank in which case it counted towards the "Do not know"-category. In cases where respondents left half or more of the form empty it was assumed that they had aborted the test and in those cases all of their responses were deleted from the results. The images that were used for testing can be seen in Appendix A.

The most important results are 81.7% total success rate, 84.9% male success rate, and 78.3% female success rate. It is worth noting that male avatars have a significantly higher success rate and many respondents have also pointed out that they thought they were much more confident of their choices for the male avatars.

Table of results:

All	Total answers:	Correct:	Errors:	Do not know:	Success rate: (%)	Failure rate: (%)	Do not know rate: (%)
	7040	5750	770	520	81.7	10.9	7.4
Male avatars	Total answers:	Correct:	Errors:	Do not know:	Success rate: (%)	Failure rate: (%)	Do not know rate: (%)
	3520	2987	387	146	84.9	11.0	4.1
Female avatars	Total answers:	Correct:	Errors:	Do not know:	Success rate: (%)	Failure rate: (%)	Do not know rate: (%)
	3872	3033	454	385	78.3	11.7	9.9

4.1.1 Comparison of avatars from different cameras

No resulting avatar can be better than the source images used to create it. If the source image is taken with a low quality camera or under sub optimal light conditions, then the final result will suffer. Consider the example source photos below, all taken of the same subject:

Figure 40 Photo taken with, Left : Webcamera indoors light from lamps only, Middle: Mobile camera indoors light from lamps and camera flash, Right: System camera outdoors with only natural light.

Visual inspection easily determines that image quality can differ significantly depending on camera hardware as well as light conditions. If the quality is considered to be good enough greatly depends on the usage scenario however. Consider the resulting avatars below that were created from the source images above:

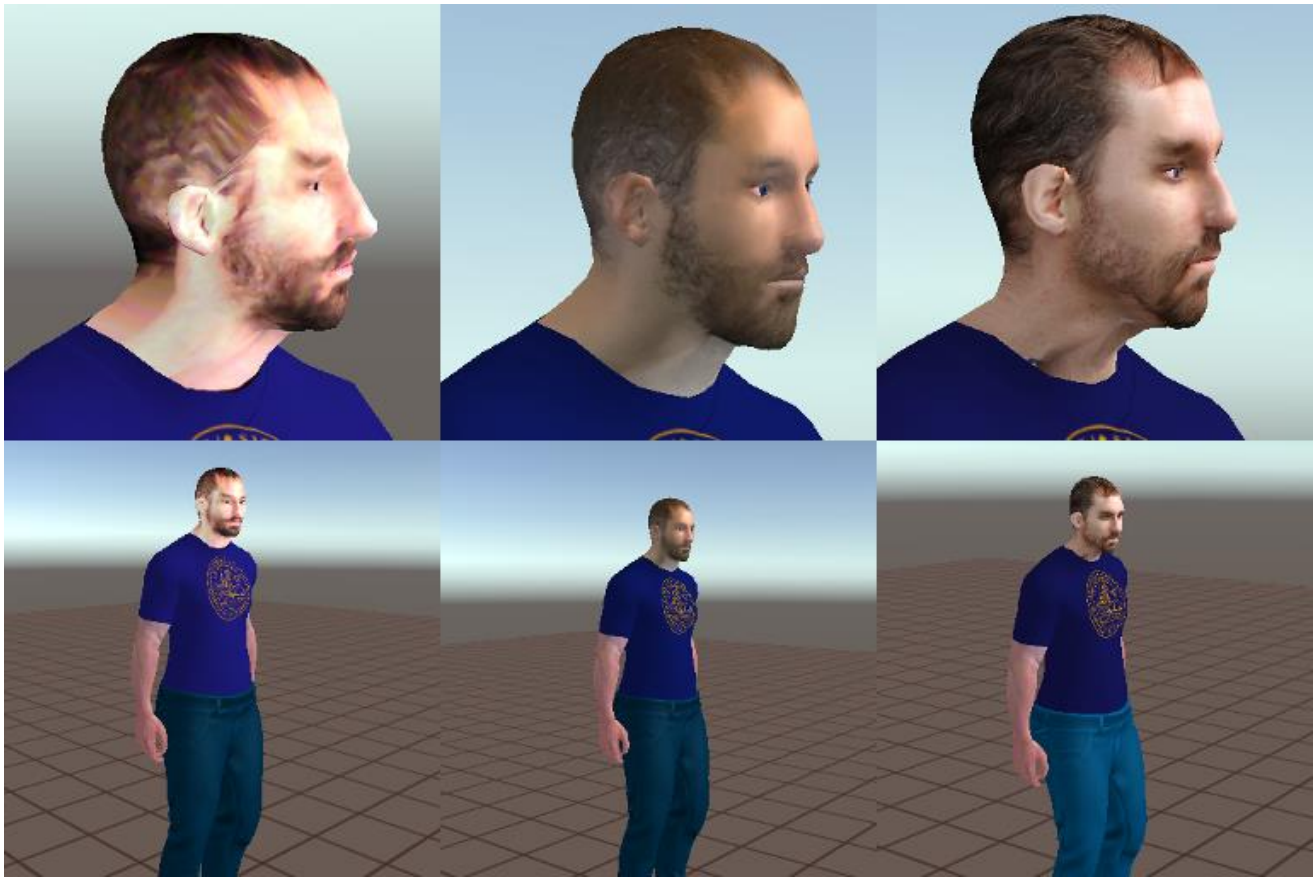


Figure 41 Resulting avatar from photos taken with, Left: Web camera, Middle: Mobile camera, Right: System camera. Note also how the different light conditions in the images result in different skin colors hues in the avatars where the web camera light is "too cold" due to overexposure, the mobile camera light is too warm due to too little light while the system camera light is closest to true colors.

The final quality is, once again, easily determined to be superior in the avatar to the right, at least when viewed in a camera close up. Consider however what happens if the camera moves back enough that the entire avatar is visible in the camera view. At this scale the quality difference is not as easily determined. This may be the view that will be more prominent for many use cases such as games etc. These use cases may not require the highest quality avatar but could actually suffice with one generated from only a web camera. On the other hand, a VR-meeting application, where user interact up close, might suffer with anything but the highest quality avatars.

4.1.2 Comparison to existing systems

Not all of the existing systems are available for testing without acquiring expensive consumer products. Some of them are either free or very cheap to use though, and these have been tested using the same input as the input used in Selfie to Avatar.

Figure 42 Avatars created using different systems. Left to right: EA Games Gameface, 3D Avatar Store and Selfie to Avatar. Note that in all of these some postprocessing has been performed.

Excluded here are EA Games NBA 2015 as it is not free and Automated Avatar Creation for 3D Games as it is not available. In all of the systems displayed above the eye color has been manually selected. All of them allow for some manual shape adjustment after avatar generation. Only Selfie to avatar has automatically generated hair, the other two systems require a hair 3D model to be selected but none of them actually texture the hair with source image data. Selfie to Avatar textures hair but does not adjust vertex positions for the hair which means it will not display any volume for it.

4.2 Individual algorithm steps' performances

In this section we evaluate the individual steps of the Selfie to Avatar method. This gives us a baseline for later discussion on how the method could be improved incrementally by improving each of these steps.

4.2.1 Image segmenting performance

As has been mentioned before the ASM does have problems with hairstyles but generally succeeds in finding the overall shape of a face. When the algorithm succeeds there is still often an error of one or a few pixels. As the border pixels do not have a great effect on the resulting texture and a few pixels do not affect the general shape much this small error can usually be ignored. In cases where it does have a negative effect, this can be seen in real time and the user can manually correct it. It often fails to position the eyes with pixel perfect precision. This might be due to a complex image structure around the eyes with many edges, not only for the eyes, but also from wrinkles

4.2.2 Pixel projection performance

The pixel projection is most accurate in those parts of the image that are perpendicular to the camera. Areas that are not perpendicular do get some texture stretching due to each pixel covering a larger area. This is mostly visible on the top of the head, the back of the head and under the chin. This could be improved by taking images from more directions or by performing procedural texturing in such areas. In some cases, it might be hidden by use of props like hats or scarves depending on the use case scenario.

One weakness of the current state of the method is that it is susceptible to effects of different light conditions. As was shown in the comparisons between cameras the light and color composition of an image has a profound effect on the final result. Currently the only way to get a high quality result is to take photos outdoors in the shade or in cloudy weather. Overexposure of an image due to too strong direct light can currently not be handled.

4.2.3 3D model adjustment performance

The manual step sometimes makes faulty adjustments on certain facial parts. For example, a nose may be made too thin like in figure 14 fourth image from the left. This is an artifact that stems from the method used to "lock" vertices (See section 3.2.1) to the mean except for those belonging to one facial part. If the face then has a pose that is not accounted for, like head tilted to one side, then the projection to shape eigenspace will find the projection with the least error which may include e.g. a thinner nose than what the person has in real life. If we could instead rely entirely on the whole shape without locking any vertices to the mean this problem would be reduced due to the modes that describe head pose. This might be possible if we had many more samples to train the model than we currently have, hopefully giving us eigenvectors representing shape variations that are more easily distinguished when setting connections between eigenspace coordinates and bone transforms.

Another facial shape that is prone to exaggeration due to variations of the head pose is the jawline. Consider a head that is tilted back. The jawline now becomes more square and the head more round. If the head is tilted slightly forward the effect is the opposite. Again, if we could use the full face modes instead of locking vertices, this pose too would be properly described by some of the modes and then we could simply ignore those modes and use the remaining ones to describe shape.

If we consider the second mode of variation in figure 5 we can see that this mode describes such a pose variation. If we could use the eigenspace coordinates of the full shape of the face, then the pose variations of the head would likely be taken into account by the second mode and we could then better "trust" the modes following it. Currently, as has been previously mentioned in the methodology chapter, this is not possible as it is too hard to discern individual mode variations' connections to bone transforms but if we had a significantly larger training database then those modes may become more clear to us as users and such a connection could be made.

Another problem with the 3D model adjustment is that the current method, using two perpendicular images, will have some trouble adjusting shapes to parts of the face that do not produce a clear edge from any of those two views. Setting the height for cheekbone is one such facial feature that is important for the final result but that produces little discernable edges in the image structure from the two directions we use.

5 Conclusion

In this chapter we first discuss the general results, focusing on the difference in recognition rates of male and female avatars and answer the initial research questions. We proceed to discuss results of the individual parts of the algorithm where we initially focus on new features that could be added to the system and then go into detail of how the quality of the existing features could be improved.

The research questions were:

Can we create a 3D face for a virtual avatar from one front view and one side view photo to such a quality that the result can be matched to that person?

How much does the quality of the results differ when using a low quality camera like a standard mobile device camera and a high quality one like a system camera?

Is it feasible to let the user guide the processes that are automated in order to avoid problems of the algorithm reaching convergence in bad local minima, thereby avoiding problems with "monstrous faces"?

5.1 Final result

We have shown that we *can* indeed produce avatars from only two images and the results can be matched to that person in 81.7% of the cases. Whether a recognition rate of 81.7% is good enough to warrant widespread practical use remains to be shown. There are many areas that allow for improvement of the method. This indicates that the recognition rate is likely to be improved in the future.

Male avatars are significantly easier to recognize than female ones when created with this particular system. There are several interacting causes. First and probably most important, the system does not handle volume in hair. Hairstyles that incorporate some volume will not be correctly modeled. As women generally have longer and more voluminous hair than men it is natural that their avatar quality will suffer. Short hair on the other hand works well to portray using only texture data.

The second reason that male avatars are more easily recognized is related to 3D data versus texture data as well. As the system does not use an exact 3D reconstruction it becomes more dependent on texture data than vertex data. Many males have short beards or stubble, so called "five-o-clock-shadows", which will help distinguish them, and these features are well portrayed using texture data. Female face textures, containing no beards, do not vary as much as those of males and are thus harder to distinguish.

A third reason depends on the base 3D models used to display the avatars. These have been created by a 3D artist who has not been aware of the exact mean shape of males and females. Of these two

models the male one seems to more closely resemble the mean male shape than the female model resembles the mean female shape. As these 3D models are then adjusted to "fit" a certain face, the female shape will generally be a bit harder to work with as there are more adjustments that need to be made. It would be interesting to see if the results could be improved by implementing models created from true means in measured 3D data. This is a very interesting conclusion as it could be generalized outside of *automatic* avatar creation and carries weight in all types of avatars. If human faces should be modeled in some way, then the 3D artist building the base models need to use a true mean. Such means can be calculated from existing face databases e.g. the 3D_RMA database [14].

The main conclusion from this is that the main weakness of the *Selfie to Avatar* system is its inability to properly model hair. See section 5.2.1 for a suggestion how this could be improved.

Another very interesting detail to note is that a very significant amount of data is contained in only the texture as compared to the 3D shape of a face. When comparing Figure 33, where the models have only had their texture applied, to Figure 34, where they also have their shape adjustments applied it is apparent that the most important step is applying the texture. This leads us to the important conclusion that *any* system that aims to reconstruct faces need to take the greatest care in correctly applying texture to the model.

5.1.1 Quality differences between avatars created from different cameras

We have also shown that there is a distinct quality difference between avatars generated from low quality cameras and those generated from high quality cameras. This may not be a very surprising result but the interesting part is that the difference may not matter in some cases such as when an avatar is to be mostly viewed from a distance of a few meters or more. This is probably the case in most computer games where it is more common to view other characters at some distance. For other types of software, e.g. virtual reality meetings or applications for online shopping of clothes closer viewing distances should be more likely. The important conclusion here is that the developer needs to analyze the application and determine what the lowest acceptable quality standard will be for her specific application and determine what the input image quality requirements based on it.

5.1.2 User interaction with the algorithm

We have shown that it is feasible to let the user guide the process if necessary in order to produce better results. It does not *entirely* prohibit the creation of monstrous faces as described by Kotaku in [3], but at least they can be more easily corrected in a post processing step. From a user experience point of view, it is a hard balance to strike between automating the process which will make it less tedious for the user and to allow or require user interaction in order to produce better results. Finding that exact balance is well outside the scope of this thesis and may be the basis for future research, especially now that we have shown it is possible to include the user more in the process than some existing systems like EA Sports NBA 2015 do.

5.2 Future work

In this section we consider possible features that could be added to the system in order to increase its quality.

Aside from adding entire new features the performance of each individual step could also be improved. The segmentation could be improved by training it with more samples or using a more advanced segmentation algorithm like Active Appearance Models described by Cootes, Edwards and Taylor in [15].

Using a bone hierarchy for editing a face based on eigenvector shape variations may not be an optimal solution. Instead using one blendshape, described by Lewis and Anjyo in [16], per eigenvector and weighting them together using eigenspace coordinates would be a more direct way to describe 3D adjustments. This would improve the 3D model adjustment step.

When projecting pixel values from source images we only use one source image from the side. This means that any features that are non-symmetrical, e.g. scars, birthmarks etc., will be present on both sides of the face. We could improve this by allowing the user to submit one photo of the other side and in that case use that photo as source image for texturing the other side of the face.

5.2.1 Add hair with volume

As was mentioned in the previous section the current model cannot properly handle hairstyles that have volume. Adding such a capability would likely do very much to bridge the quality gap between male and female avatars. We propose three possible solutions to this problem.

The simplest way to generate volume for hair is to move the vertices of the 3D model outwards. This could be accomplished using bones just as we currently do for parts of the face. The main drawback of this solution is that it will only be able to feature hairstyles where the vertices are moved outwards and will not be able to do e.g. a pony tail or braids unless they are modeled separately. It also does not improve the "flat look" of the hair as the texture and the material is just as flat.

A more intricate solution would be to construct a pixel displacement shader program where individual hair pixels are moved based on a constructed height map. Such a height map may be constructed using hair pixel intensity values or some other constructed heuristic. It would require the algorithm to be able to determine which pixels are hair and which are skin. We can generate statistical knowledge of the properties of both hair and skin using the individual triangles of the source shape and prior knowledge of their likelihood to contain hair. E.g. a triangle in the forehead has a low likelihood to contain hair and a triangle on the side of the head has a high likelihood. It should be possible to generate a Bayesian model from this data to determine if individual pixels have hair or not. The advantage of such a model is that not only can it generate some volume of the hair but it also would give the hair a non-flat surface which would increase the graphical fidelity.

An even more advanced hair model would be to perform a full hair strand simulation in real time. In previous years such full scale simulations would not have been possible in real time and instead so called *fur shaders* as described by Lengyel, Praun, Finkelstein and Hoppe in [17] would have had to be used. Recent research by Guan, Sigal, Reznitskaya and Hodgins [18] suggests that real time full hair strand simulation is possible though, and if we could implement that in an avatar system like *Selfie to Avatar* and have avatars with properly modeled hair then the recognition rate would probably improve dramatically, especially for the female avatars who now suffers from the poorly modeled hair.

5.2.2 3D reconstruction through a statistical method

Instead of training the model using only 2D images we could train a model with 2D images and a ground truth 3D data measure using e.g. a depth sensor or laser scanning. A statistical model could in that case be built calculating the most probable 3D shape based on given segmentation of 2D images. This would remove the need for manual settings of effects of eigenspace coordinates and automate that step instead.

5.2.3 3D reconstruction through structure from motion

As was described by Hogue et al in [4] an avatar could be constructed through structure from motion rather than segmenting a set of images. In their method they do not, however, address the problem of creating a rigged mesh which is a requirement if the model is to be used in an interactive environment. This would require pose estimation and a way to segment the 3D data to determine which vertices belong to which facial part. This would probably be a very important way to improve on the *Selfie to Avatar* system and would potentially generate extremely lifelike avatars, at the cost of requiring a few more photos. Note that this would also automatically handle vertex displacement to take hair volume into account.

5.2.4 Image processing

The lighting conditions when taking the photos has a large effect on the final result. If the user could manually adjust the image hue, saturation and value this may help to alleviate such problems. It might even be possible to automatically adjust the images so that their face pixels share similar color histogram properties. As no image preprocessing is performed in the system at all this is probably an area that could result in vast improvements of the quality of the final texture.

5.2.5 Eigenfaces to describe face texture

Right now we use a texture that is 1024 by 2048 pixels. This results in textures that are about 2-3MB in size using the lossless png-format. This is not a problem when displaying a single avatar on modern devices but it would not work in a massive multiuser environment like an MMO-game where there could be tens or hundreds of users in a scene. Streaming all that data to every user would simply take up too much bandwidth. A possible solution might be to calculate eigenfaces, first described by Sirovich and Kirby in [19], of the textures and only stream the coordinates in that eigenspace, using pre-distributed Eigen face textures to calculate face textures client side.

Face, hair and beard may need to be modeled independently to reduce the dimensionality of the problem as 1024 x 2048 x 3 (in the case of rgb color format) might be too high dimensional to calculate in reasonable time, even with dimensional reduction such as described by Cootes in [6]. Procedural texturing of hair and beard based on statistical models may be a requirement in this case as Eigen faces are unlikely to be able to feature individual strands of hair.

6 Critique

It is not entirely obvious how to measure the quality of an avatar. Whether an avatar is recognized among of a set of other avatars may not be the best way to determine its quality as the real quality is in how realistically a user feels an avatar represents her. Measuring this highly subjective appreciation is not a trivial problem though. If more avatar creation systems enter the market it might make sense to establish a standardized test somehow in order to properly compare their quality.

In the beginning of the system performance test it is stated: "*Note that this is not a test of your abilities but a test of the quality of the system. Don't spend too much time trying to figure out who is who, if you do not instantly recognize a person then state that fact.*" Still, test respondents tend to diverge from testing the system and start testing themselves. Many respondents also asked for "the correct answers" to "see how they did" which supports this observation. Thereby there is a risk that they try to use a method of exclusion to figure out which avatar belongs to which photo in order to get a 100% score. Hence the recognition rate is probably slightly overestimated. It might be better to test 1 avatar against 20 photos in 20 different questions rather than show all 20 at once to reduce that risk.

The comparisons to existing systems and between different cameras only consist of a single avatar each. With such a small sample they can be seen only as indicator and not proof of any statement. Given more time and resources it would be very interesting to see how different systems measure up against each other in a test.

7 Bibliography

[1] B. Senftner, "3D Avatar Store," 3D Avatar Store LLC, 07 01 2016. [Online]. Available: Automated Avatar Creation for 3D Games. [Accessed 07 01 2016].

[2] L. Plunkett, "Kotaku," Gawker Media, 14 09 2014. [Online]. Available: http://kotaku.com/i-wish-skyrim-had-face-tech-like-this-sports-game-1636127917. [Accessed 07 01 2016].

[3] E. Narcisse, "Kotaku," Gawker Media, 07 10 2014. [Online]. Available: http://kotaku.com/nba-2k15s-face-technology-fails-miserably-creates-mons-1643490237. [Accessed 07 01 2016].

[4] S. G. M. J. Andrew Hogue, "Automated Avatar Creation for 3D Games," York University, Toronto, 2007.

[5] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active Shape Models - Their Training and Application," Academic Press, Inc, Manchester, 1992.

[6] T. Cootes, "An Introduction to Active Shape Models," Oxford University Press, Oxford, 2000.

[7] C. Goodall, "Procrustes Methods in the Statistical Analysis of Shape," *Journal of the Royal Statistical Society. Series B (Methodological),* vol. 53, no. 2, pp. 285-339, 1991.

[8] E. Angel, Interactive Computer Graphics - A top down approach using OpenGL, Boston: Pearson Education, 2009.

[9] D. H. Eberly, 3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic, San Fransisco: Morgan Kaufmann Publishers, 2005.

[10] S. Milborrow, J. Morkel and F. Nicolls, "The MUCT Landmarked Face Database," *Pattern Recognition Association of South Africa,* 2010.

[11] M. De Simone, A. Bottino, A. Laurentini and T. Vieira, "Detecting Siblings in Image Pairs," *The Visual Computer,* vol. 30, no. 12, pp. 1333-1345, 2014.

[12] P. Viola and M. Jones, "Robust Real-Time Object Detection," *International Journal of Computer Vision,* 2001.

[13] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Applied Computational Geometry: Towards Geometric Engineering,* vol. 1148, pp. 203-222, 1996.

[14] C. Beumier, "3D_RMA : 3D database," Signal and Image Centre (SIC) - Elec Dpt, Royal Military Academy of Belgium, [Online]. Available: http://www.sic.rma.ac.be/~beumier/DB/3d_rma.html. [Accessed 12 January 2016].

[15] T. F. Cootes, G. J. Edwards and C. J. Taylor, "Active Appearance Models," *Proc. European Conference on Computer Vision,* vol. 2, pp. 484-498, 1998.

[16] J. P. Lewis and K. Anjyo, "Direct-Manipulation of Blendshapes," *Engineering in Medicine and Biology,* vol. 30, no. 4, pp. 42-50, 2010.

[17] J. Lengyel, E. Praun, A. Finkelstein and H. Hoppe, "Real-Time Fur over Arbitrary Surfaces," *Proceedings of the 2001 Symposium on Interactive 3D Graphics,* pp. 227-232, 2001.

[18] P. Guang, L. Sigal, V. Reznitskaya and J. K. Hodgins, "Multi-linear Data-driven Dynamic Hair Model with Efficient Hair-body Collision Handling," *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation,* pp. 295-304, 2012.

[19] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America A,* vol. 4, no. 3, pp. 519-524, 1987.

8 Appendix A- Face match test

This was the test that was posted online in order to test the system performance.

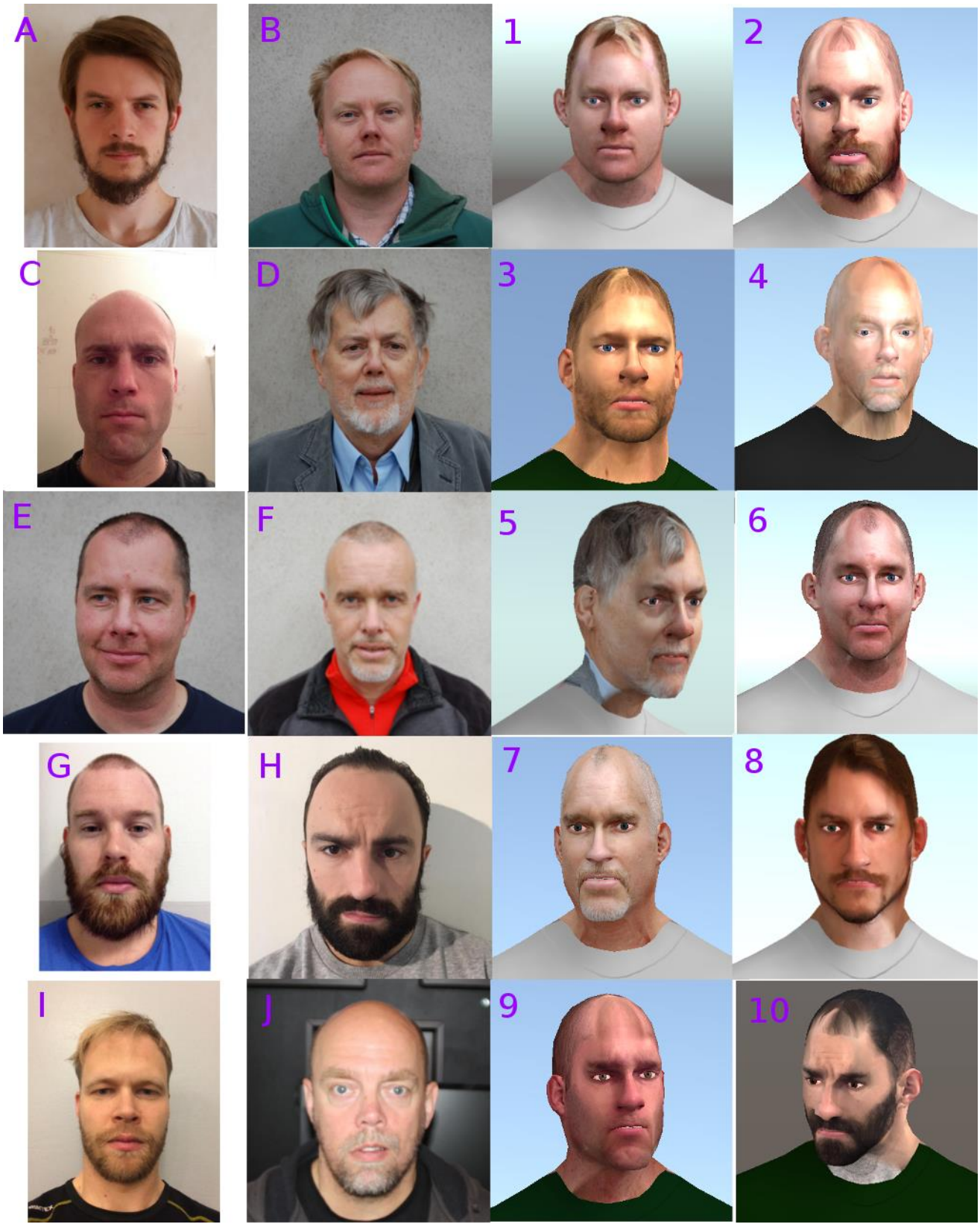
Face match test

I need help with my master thesis! I have created a system that generates avatars based on photos. In order to test the quality of the system I need to see at what rate the generated avatars are recognized as the person in the photo. Try to figure out which real world face belongs to each generated avatar!

Note that this is not a test of your abilities but a test of the quality of the system. Don't spend too much time trying to figure out who is who, if you do not instantly recognize a person then state that fact.

For more info: http://forum.unity3d.com/threads/wip-uma-master-thesis-player-avatar-creator.343331/#post-2422671


A
B
1
2
C
D
3
4
E
F
5
6
G
H
7
8
I
J
9
10


A
B
1
2
C
D
3
4
E
F
5
6
G
H
7
8
I
J
9
10

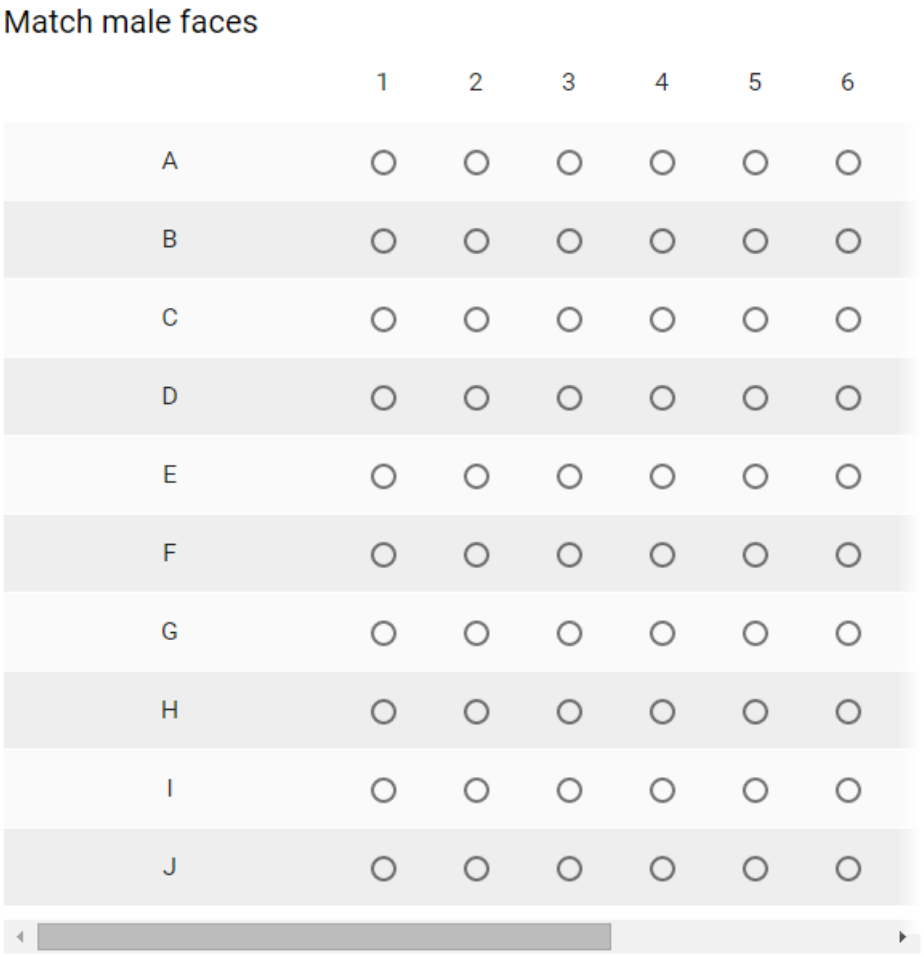


Figure 43 The matching tool in the test