A Physically-Based Muscle and Skin Model for Facial Animation

Submitted to the University of Glasgow

for the degree of Doctor of Philosophy

Alasdair D. Coull

January 2006

Department of Computing Science

University of Glasgow

Glasgow, UK

Abstract

Facial animation is a popular area of research which has been around for over thirty years, but even with this long time scale, automatically creating realistic facial expressions is still an unsolved goal. This work furthers the state of the art in computer facial animation by introducing a new muscle and skin model and a method of easily transferring a full muscle and bone animation setup from one head mesh to another with very little user input.

The developed muscle model allows muscles of any shape to be accurately simulated, preserving volume during contraction and interacting with surrounding muscles and skin in a lifelike manner. The muscles can drive a rigid body model of a jaw, giving realistic physically-based movement to all areas of the face.

The skin model has multiple layers, mimicking the natural structure of skin and it connects onto the muscle model and is deformed realistically by the movements of the muscles and underlying bones. The skin smoothly transfers underlying movements into skin surface movements and propagates forces smoothly across the face.

Once a head model has been set up with muscles and bones, moving

this muscle and bone set to another head is a simple matter using the developed techniques. The developed software employs principles from forensic reconstruction, using specific landmarks on the head to map the bone and muscles to the new head model and once the muscles and skull have been quickly transferred, they provide animation capabilities on the new mesh within minutes.

Contents

List of Figures

Acknowledgements

I would like to thank my two supervisors who kept me on course throughout this work, Dr. John Patterson and Dr. Paul Siebert.

I would also like to give my deepest thanks to my family and friends to whom I am indebted for their assistance and support: David Coull, Gwendoline Coull, Fiona Hitchman, Michelle Coull, Christopher Coull, Ciaran Wills, Stuart Ford, Joshua Hale, Duncan Jarvies, Sylvain Brugnot, Donald MacVicar and Xiangyang Yu.

Alasdair D. Coull

January 2006

Declaration of Authorship

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Alasdair D. Coull

January 2006

1. Introduction

The face is perhaps the most examined feature of the entire human body. When speaking to someone we are continually studying their face. Even when we do not realise, we are continually watching the minute changes in its surface. The flick of the eyes away during speech may show shyness, the lack of creasing under the eyes during a smile seems to indicate it's not entirely genuine and the furrowed lines between the eyebrows are definitely showing anger. These non-verbal signs of emotion are simple examples of how humans use their faces to show their meaning during speech. If they were taken away from a person, then that person would seem unreal, insincere, even robot-like. This intense scrutiny that people place on facial movements is one of the huge problems of facial animation. If the results are in any way unreal, it will be glaringly obvious to an observer.

When facial animation is mentioned, most people think of animated speech, but speech is only half of the issue. Humans can pick out the slightest movement or expression in a face; this is how subtle emotions are conveyed. Simulating the movements of the face on computer is a

complex task but when the subtleties of expression are added it becomes a very lofty goal indeed.

Facial animation is entering into everyday life more and more. The entertainment industry has been using facial animation as long as the industry has existed. As computers became more powerful and accessible, animation in general and thus facial animation moved on to computers too. In 2001 Square created Final Fantasy [Squ01] the first film to have a fully computer graphics human cast. It was met with a mixed response but even with a budget of millions the facial animation did not impress critics.

Possibly the most successful example of facial animation in film to date is not actually human, but rather the creature Gollum from the Lord of The Rings trilogy [Jac02]. The facial animation of Gollum has been widely hailed as excellent and set a new standard for film facial animation. However, Gollum was not a breakthrough for facial animation research but rather the painstaking effort of many animators and one actor. Every scene Gollum was in was acted out by an actor and the artists could then copy his every facial movement and expression to inject life into the computer graphics (CG) model.

Like the film industry, the computer games industry has similarly embraced facial animation. Traditionally a lack of processing power meant

that the facial animation in games either did not exist, was done offline for pre-rendered scenes, or was of very low quality. In recent years there has been a trend to improve things, but so far the results have been less than impressive. This is most likely because facial animation is considered secondary in games and not key to the gameplay. This could soon change, as large amounts of money are being spent to capture the likeness of movie actors to make their game counterpart seem realistic. On the whole, this effect is ruined when the character starts to speak as the excellent CG likeness and the CD quality recorded speech are synced to an unrealistically moving mouth.

The entertainment industry is not the only area to have embraced facial animation. The medical community has long been trying to use it to aid in surgical planning, training and simulation [DWD+83], [KAA83], [Pie89], [KGPG96]. A realistic model of the physiology of the face is an excellent tool for examining the effects of surgery. Surgeons can visualise what a person's face will look like after a surgical procedure and decide with the patient whether to go ahead with the operation or not. Facial simulation and animation can also be a useful tool for teaching. Medical students can see the structures of the face working and surgeons can practice procedures on computer models with no danger to the patient.

Facial animation is obviously an important research topic with wide

ranging applications, however so far no perfect facial animation system exists and the field is still an area of intensive research.

This work furthers the state of the art by describing a new muscle model for facial animation which allows muscles of any shape to be contracted whilst bulging out to preserve volume. The muscles also automatically interconnect to realistically simulate the intertwining of human facial musculature and the contraction of a single muscle will smoothly affect neighbouring muscles. The muscles drive a rigid body mandible which moves realistically and pulls on a new model of human skin which behaves in a manner very like that of human skin. The skin consists of multiple layers which are automatically created and connected to the muscles from the input of a polygon skin surface mesh and during muscle contraction the skin moves smoothly and realistically. Once a set of muscles have been created and are working for animation of a face mesh, they can be easily transferred to another face mesh. This is performed using a novel technique requiring that only a few points are selected by a user, and the muscles and skull are then automatically warped to fit the new model.

This thesis will proceed as follows:

Chapter Two

Chapter two will outline the relevant previous works in the area of

facial animation. There has been an extensive amount of work done in the area and this chapter will single out the most important and relevant, and show where there are gaps in the current knowledge.

Chapter Three

Chapter three gives an overview of the developed software. It explains how the parts come together, what each module does and how data flows through the system.

Chapter Four

Chapter four explains the muscles. It explains how the muscle model works and how the muscles interact. It then covers how the muscles drive the mandible and how the skull and muscles can be easily adapted to fit any surface mesh desired.

Chapter Five

Chapter five covers the skin model developed. It starts with a brief overview of the physiology of skin and then explains in detail the methods used to simulate this complex anatomy.

Chapter Six

Chapter six details the experimental work done with the system and covers the tests used to gauge user opinion. It then shows the results gained from these experiments and what they say about the system.

Chapter Seven

This chapter summarises the work done and details the conclusions drawn from the experiments and then covers what future work could be done using this system as a starting point.

1.0.1 Medical Terminology

Medical terminology has been kept to an absolute minimum throughout this work, but it is used in one or two places where it simplifies the description of an action. For this reason it is worth noting the meaning of a few words that are used.

Contralateral - The opposite side of the body.

Ipsilateral - The same side of the body.

Superior - Above something. For example the lips are superior to the chin.

Inferior - Below something. For example, the lips are inferior to the eyes.

Posterior - Generally refers to the back of the body.

Anterior - Generally refers to the front of the body.

2. Literature review

Realistic computer facial animation has long been a goal in computer graphics research, with the first work in the area being published over thirty years ago [Par72]. Over the years, much research has been done in the area, but results still vary greatly in quality and realism.

Facial animation research has gradually expanded into four broad areas, each defined by the method used to create animation. These methods are: *parametric* methods, *geometric muscle* methods, *physics based* methods and *image based* methods. Each of these areas takes a different approach to the problem. Parametric methods generally ignore the physiology of the face and use simple deformations to move a face mesh. Geometric muscle methods use very simple muscle models to try and move the mesh. These muscle models are usually unrelated to real muscle physiology and usually move the surface mesh by simple geometric deformations. Physics based models try and simulate the actual behaviour of skin by modeling its structure so that it behaves in a manner similar to real skin, however these models often ignore the muscle size and shape and use simplified models. Image based methods try

and recreate video of facial animation, avoiding the problems created by trying to model such a complex structure in three dimensions.

The four classes of model will be considered in order of their complexity and realism, that is, parametric, geometric, image-based and finally physically-based. For completeness direct interpolation based methods are briefly considered before other methods as their development heralded the start of the whole area of computer based facial animation.

No matter which method is used to create facial animation, a common problem is how to describe the movements of different parts of the face uniquely. For example, in an animation system there might be a need to define exactly what a frown or a smile is. The human face can be manipulated into a large number of different expressions and defining how each part has moved requires a detailed set of rules. Expression coding systems were developed for just this purpose. They aim to describe the movements of the face in more general terms than the movement of muscles but with enough accuracy to describe any visible movement in detail. There are several expression coding systems in use and each tries to encode all of the possible physical movements of the face into a small set of parameters. The main expression coding systems were not designed for use in facial animation, but over the years some have been adopted by researchers to help them describe the motions of their systems. These

will be considered first before moving onto the previous works in facial animation.

2.1 Expression Coding Systems

The Facial Action Coding System (FACS) [EF78] is extensively used in facial animation research. It was developed by Ekman and Friesen as a way to describe the most basic facial muscle actions and their effect on facial expression. The FACS system was designed to describe every possible *visual* facial movement, which in FACS terminology are called Action Units (AUs). Movements that are too subtle to notice or would not be visible to an observer are not included. The FACS action units are based on all of the facial muscles that can be contracted voluntarily. Not all muscles have a corresponding Action Unit. Where more than one muscle produces the same visual surface movement only one action unit is defined. One muscle may also be associated with more than one action unit such as the Levator Labii Superioris which is involved in the *Nose Wrinkler* and *Upper-Lip Raiser* AUs. FACS is complete for distinguishing actions of the upper face however it cannot fully describe the movements of the lower face. The moveable jaw and flexible lips allow a huge number of actions and defining their movements in a finite

set of Action Units would lead to a very high number of AUs which would most likely be too large a set to be useful. In 2002 the FACS manual was updated [EFH02] to make it easier to understand and include video clips of Action Units. However so far there seems to be no reference to this new version in any work.

In recent years, the MPEG-4 standard [Gro00] has been adopted to some degree but has not been nearly as widely used in the literature. MPEG-4 specifies a set of face animation parameters (FAPs), each corresponding to a particular facial action deforming a face model from its neutral state. These FAPs are much like the action units of FACs in that they define the set of minimal perceptible actions of the face. To create a facial animation sequence using FAPs, the face model is deformed from its neutral state according to a stream of FAP values which are specified along with a time code defining when each FAP should be executed. These FAP values can be transmitted across some medium allowing animation to be produced between a distant sender and receiver.

Magnenat-Thalmann et al. [NMTT88] propose the idea of Abstract Muscle Action (AMA) procedures. Each AMA procedure corresponds roughly to a muscle or bone structure. The FACS action units were used as a guide for constructing the AMA procedures and each AMA is designed to mimic one or more facial muscles. There are thirty AMA

procedures and they are not independent, so the ordering of the actions is important. AMA procedures have not proved popular and have not been used extensively in the literature.

2.2 Non-Physically-Based Methods

2.2.1 Interpolation

For his Masters thesis Fred Parke [Par72] created a mesh of his wife by hand digitizing her face and using photogrammetry techniques to create models of her in various expressions. Each model was digitized into the same mesh and thus each model had the same set of vertices and connectivity. This allowed Parke to create animation by simply interpolating the vertex positions from one pose to the next. The obvious problem with this method is that the human face can deform into an almost infinite number of expressions and so producing the keyframe meshes for anything more than a small number of face poses is very time consuming and labour intensive.

Even with these drawbacks, direct interpolation methods have continued to have some popularity, particularly in image based facial animation. Pighin et al. [PAL+97], [PHL+98] conform a 3D face model to fit the

landmarks on a face captured by several cameras using scattered data interpolation [LS89] to map all surface points. Textures are extracted from the photos and combined into one image using view dependent texture mapping [Deb96]. The actual animation is accomplished by linear interpolation of the mesh points from one captured key frame to the next. The textures are interpolated and blended together at each frame using the view dependent texture mapping again.

2.2.2 Parametric Methods

The obvious problems with direct interpolation methods led Parke to extend his work to a more generalised parameter based animation system [Par74], [Par75]. He created a model that was capable of representing different individuals via conformation parameters and different expression using expression parameters. Facial features are defined by altering conformation parameters such as lip width, forehead size and lower nose width. Animation between poses can be produced by interpolating the expression parameters which will animate the features. This makes creating animation a far quicker and easier task than having to set and interpolate every single vertex. Parke assumed symmetry in the parameters and felt that he needed just ten parameters to produce animated

speech: jaw rotation, mouth width, mouth expression, upper lip position, eyebrow arch, eyebrow separation, eyelid opening, pupil size and eye tracking. Later Parke [Par82] improved the model by adding more parameters and allowing the parameters to be altered on each side of the face seperately.

Lewis and Parke [LP87] used linear prediction to process recorded speech, determining the phonemes. These are turned into keyframes of a modified version of Parke's model. Pearce et al. [PWWH86] modify Parke's model and use a system which lets them use phoneme information to generate synchronised speech and facial animation by sending the phonemes to a speech synthesizer along with the facial animation model simultaneously. Their system also has a scripting interface which allows a library of expressions and phonemes to be created and accessed by an animator giving high level control over the animation. Cohen and Massaro [CM93] use the work of Pearce et al. and extend it by adding a tongue. Their work was psychological in nature and they were mostly interested in examining how speech can be understood from the visual cues of lip and tongue movements.

In 1991, DiPaola [Dip91] extended Parke's work to allow a far greater range of face shapes to be produced. DiPaola extends the work to include approximately one hundred parameters to control the face, head shape

and facial hair. He also added squash and stretch parameters to the system, allowing the model to be be scaled easily. Ellipsoidal warping volumes were also added. Using these, facial vertices which lie within the ellipsoid are warped when the ellipsoid is moved or changed. The warp function decays smoothly to zero so that the warps have local influence over the model. These warping volumes provided a simple way to make general changes to the model shape rather than having to add specific parameters for items such as ear shape.

2.2.3 Geometric Muscle Models

Parameterised models allow animators to create animation using a small number of controls, and are quick and easy to use. These models suffer from the problem that unless the parameters are created with great care then unrealistic motion and shapes can be produced during animation. To improve the realism of animation, research started to move towards using parameters based on the actual anatomy of the human face.

Platt and Badler [PB81] introduced a system using a very simple muscle model. They created a system where several of their simple muscles would be combined to produce the movement of each Action Unit from the FACS [EF78] work on facial expression. Their muscle model con-

sists of several *muscle fibres* which consist of three points: A bone point where the muscle connects to the skull, a muscle point where the muscle force is applied and one or more skin points simulating the head of the muscle, where it connects into skin. There is no real model of a skull in this model. The skull only exists at the points where the muscle origin is fixed. During muscle contraction, a force is applied to the muscle point in the direction of its bone point. The same force is then reflected along all the skin points for this muscle where it is applied to all the vertices connected to the muscle surface points. The force applied depends on the stretching factor of the skin which can be altered to produce varying deformations.

Free form deformations (FFDs) were introduced by Soderberg and Parry [SP86] as a general technique for deforming objects by warping the volume in which they are embedded. A parallelepiped lattice of control points defining a trivariate Bézier volume is used to embed the object to be deformed. By manipulating the control points, the Bézier volume is deformed and so is the object defined in it. Coquillart [Coq90] extended this to allow more complex shapes by joining several parallelepiped lattices together to create volumes which better resemble the object being deformed. Chadwick et al. [CHP89] used FFDs to deform the skin surface, trying to mimic the effect of muscles on the skin. Kalra et al.

[KMMTD92] improve on this model by using rational FFDs (RFFDs) as pseudomuscles. RFFDs are an extension to FFDs which incorporate weightings for each control point of the parallelepiped control lattice. These weights provide additional degrees of freedom when manipulating the lattice. When all of the weights are unity, the RFFD is equivalent to a standard FFD. To simulate muscle action on the face, regions are defined on the mesh which correspond to the area in which a muscle action is desired. The muscle deformations are simulated by displacing the control points and changing the weights of the RFFD. A simple method of simulating physical properties by scaling the deformation by some scale factor is also used.

In 1987, Keith Waters [Wat87] presented a more advanced geometric muscle model for facial animation. His work was later extended to improve a sheet muscle model [Wat89] and was presented slightly updated in his book with Fred Parke [PW96]. In his work the skin is modeled as a polygon mesh, where each node has a finite degree of mobility. The model includes linear, sheet and sphincter muscles. Linear and sheet muscles are modeled as vectors which follow the general direction of the many muscle fibres in the equivalent facial muscle. Each muscle vector points towards the muscle's origin and has a zone of influence between 15 and 160 degrees governing which mesh nodes it affects. The muscle

vector model works by displacing mesh nodes an amount dependent on the distance from the muscle and the amount the muscle is contracted. Maximum displacement occurs at the point where the muscle connects to skin and zero displacement where it would attach to bone. By using a non-linear function to control the rate at which the muscle movements fall off across the mesh, the elasticity of the skin can be modelled. Varying this fall-off rate and thus the amount of mesh movement allows differing skin elasticities to be modelled, giving simple control over different skin ages and conditions. Waters also includes sphincter muscles in his model. Unlike linear muscles, sphincter muscles are elliptical and contract towards a point at their centre. This draws the skin surface with it like the tightening of the material when closing a draw-string bag. Waters models sphincter muscles as parametric ellipsoids with a major axis in the longest direction and a minor axis in the shorter one. The displacement of a skin mesh vertex is then calculated proportional to the distance of each point from the centre of the muscle. Sheet muscles are modelled in a manner similar to linear muscles, but the radial displacement is replaced by a displacement based on the movement of the closest point on the muscle fibre. Waters modelled several emotions including anger, surprise, fear and happiness by contracting the muscles of his face to produce poses.

A muscle model very similar to the Waters model was used in the Pixar animation short *Tin Toy* [Stu90]. The baby's head was constructed from four sided Catmull-Rom patches which gave around six thousand three-dimensional node vectors. Around fifty muscle controls were implanted into the model and *macro* controls were created to simplify the system further. These macro muscle controls would move multiple low level muscles with a single parameter, making the model much simpler for the animators to control. Even with such extensive controls, the model showed its limitations in this animation as the skin often moved in an unrealistic manner.

Waters and Levergood [WL93] use Waters' model to create DECface, a system which generates lip-synchronised facial animation from textual input. Keyframes are associated with phonemes and animation is produced by interpolating between them. They use simple cosine based acceleration and deceleration to increase the realism of the movements.

An improved version of Waters' system which had the ability to produce the shapes required for speech was developed by Edge and Maddock [EM01] in 2001. Their work improves on Waters' model by introducing a new model for the Orbicularis Oris to allow independent control over the upper and lower lips which is important for the speech.

2.2.4 Image-Based Methods

Due to the complexities of modeling the physical structures of the face realistically, an alternative approach to facial animation using image-based techniques has become popular in recent years [SKW+94], [BCS97],[CG98], [EP00], [GCE00], [EGP02].

Image-based facial animation uses a collection of images captured of a human subject. These methods have the obvious advantage that they are blending real video footage of speech so they do not need to consider the physical structures in detail and can concentrate on combining images to produce a realistic effect. Image-based methods use purely two dimensional data to create the resulting video. These methods often try to apply to images the same concepts behind text to speech synthesizers where short sections of speech are concatenated together to produce fluid speech.

Scott et al. [SKW+94] use a set of forty to fifty visemes to animate a face. A 2D morphing algorithm is used to smoothly blend between the mouth shape images however the morphing algorithm requires considerable user input to produce good results.

Bregler et al. [BCS97] describe a system in which the output talking

head video is produced by concatenating triphone[1] video segments from a large database of segments to produce video. The segments are taken from a huge library of possible triphone segments created automatically from video clips of the subject speaking. To produce a new video, the new audio track has its phonemes labeled and the relevant mouth images in the training footage are then selected and reordered to match the phoneme sequence of the audio track. When particular phonemes are unavailable in the training footage the closest approximations are selected. The resulting sequence of mouth images is stitched into the background footage. To cope with all possible triphone combinations the video segment database has to be very large; around several thousand triphones must be stored. Cosatto and Graf [CG98] developed a method to reduce the amount of necessary stored data. Rather than storing lip images based on the sound they are producing, the images are stored using labels based on measurements such as lip width, jaw rotation and lip opening. Of course, as the cost of disk space and computer power comes down, having a smaller amount of information to store and search becomes less of a reason to use one method over another.

Ezzat and Poggio [EP00] produce video speech by morphing visemes. A subject is first captured uttering 40-50 words, each chosen so that a spe-

[1] A triphone is three subsequent phonemes

cific phoneme can be extracted from the video. The phonemes are then manually extracted and stored. Using optical flow, a correspondence between each viseme image and every other is found. This correspondence is used as the input to a morphing algorithm [BN92] which smoothly blends between any two images. Speech animation is then produced by morphing between the visemes for any word required.

Ezzat et al. [EGP02] describe a nearly fully automatic method of producing video realistic facial animation. A fifteen minute recording of a subject speaking is analysed, using user defined image masks to define the areas of mouth, eyes and head outline. Then a set of 46 images is produced automatically in a process taking two days. To create novel speech video, a *multidimensional morphable model* is used to morph between multiple images in the bank which correspond to the phonemes uttered by a recorded speech file or a text to speech system with phoneme timings.

Pighin et al. [PALS97, PHL+98] developed a technique to extract geometry and texture information from photographs using image processing techniques. A face mesh and texture is then created for each keyframe and animation is produced by interpolating between them. The geometry is linearly interpolated and the textures are blended together to produce a smooth change. To smooth this blending, the textures are warped us-

ing optical flow, bringing them closer to the texture being blended with, thereby reducing discontinuities.

2.3 Physically-Based Models

2.3.1 Finite Element Method Models

Geometric models limit the realism of animation because they ignore the physiology and biomechanics of facial structures. They usually approximate the skin as an infinitesimally thin surface with no underlying structure. Geometrically distorting such a simple approximation of the face does not produce realistic deformations or the subtle but essential wrinkles of real skin deformation.

Surgical simulations require as exact a model of the facial musculature as possible. The finite element method has proved popular amongst surgical researchers trying for ultimate realism [KGPG96] [NvdS00]. The finite element method (FEM) is more common in structural engineering and is a method of calculating the energy and forces in a system using piecewise approximations of a continuous functions defined on polygons. It reduces the problem of finding the solution at the vertices of the polygons to that of solving a set of linear equations. It is a very accurate

method of analysing a system, however it is very computationally expensive.

Larabee [Jr.86a], [Jr.86b], [Jr.86c] was the first to propose a truly physically based model of skin. By analysing animal skin he developed a finite element method model of skin which behaved in a similar manner to that observed in real life. The Finite Element Method has since become popular in computer based surgery simulation as it allows the very accurate simulation of the physical properties of skin. However it is very computationally expensive and thus is not an obvious choice for facial animation. In facial animation, small deformations are important and this means the FEM mesh must be very fine, leading to very long simulation times.

Koch et al. [RMKB98] describe the prototype of a facial expression editor. Their work builds upon [KGC+96] where they develop a 2D finite element mesh of the skin surface and use a mass spring model for the underlying structure. They create a FEM model of the facial structures based on a Computer Tomography scan if available. The CT scan is used to derive the skull geometry and tissue stiffness from a tissue segmentation of the volume data. If a CT scan is not available, a default skull is non-proportionally scaled to approximately fit the surface and the tissue stiffness values are obtained from a template model. An interac-

tive muscle editing process is performed to allow the user to edit muscle placements, origins and insertions. The user can also create weighted sums of muscle contractions to produce action units (AU). These action units are more general deformations than muscle contractions. The FEM engine precomputes the facial appearance of each action unit in order to obtain displacement fields for the individual face. Each muscle consists of approx 50 fibres that increase its influence area on the face surface but during editing the user only defines the central fibres. To determine which nodes are affected by muscles, the method of Lee et al. [YLW93] is used. To overcome the time required to simulate FEM models, the effect of pulling muscles is precomputed using the FEM system and then the deformations are stored as displacement fields that are used in their emotion editor for real time animation. This method assumes a linear deformation of the skin which is only true for minor skin deformations.

Basu et al. [BOP98a], [BOP98b], [BOP98c] use the FEM to model human lip motions. Unlike most work, they do not use the FEM to mimic the actual physiological structure of human lips, rather they use the FEM to model the lip surface as if it were made from a generic rubber sheet. A subject has their lips marked with 16 landmarks and 150 frames of video footage of them uttering phrases is captured from two views. These landmarks are used to position the 3D lip model and

the FEM is used to move the rest of the lip mesh into the position of least strain, giving the final lip position. Using Principal Components Analysis, these 150 positions are reduced down to the 10 which account for the greatest variance in the input data.

2.3.2 Mass Spring Models

Terzopoulos and Waters [WT90], [TW90] were the first to use a multi-layer mass spring system to simulate the physical properties of facial tissues to produce facial animation. They introduce a model of skin which has several layers connected with springs. The tissue is modelled as a set of connected triangular prism elements, with three layers representing the makeup of skin and bone. Simple muscles modelled as line vectors are attached to the middle layer and deformation on the surface layer due to muscle contraction is propagated through the mesh of springs. Muscle parameters are estimated from video by applying makeup to a subject and using snakes [KWT88] to track the movement.

In 1991, Waters and Terzopoulos [WT91] add to the multi-layer system by describing a method to adapt a generic mesh to fit laser scanned face data which can then be used for animation. Waters [Wat92] later developed a method using marching cubes on computer tomography data

to create a polygon mesh of the skull and skin of patients. He creates the muscle layer mid-way between the skull and skin of the multi-layer skin model.

Lee et al. [YLW93], [LTW95] improve upon [WT91] by introducing a nearly-automatic method of conforming a generic model to a scanned mesh. They also improve the model by introducing a more accurate model of spring stiffness to represent skin, and updating the muscle model. The force acting on skin nodes due to muscle contraction is now calculated along the muscle length and distance to the closest point on the muscle vector. Their new system is also capable of synthesizing functional eyes, eyelids, teeth, and a neck, and fit them to the final model.

Kolja Kähler et al. [KHS01] developed a physics based model similar to the work of Lee et al. Muscles are created interactively by the user who draws the area on the face where the muscle should lie, and ellipsoid quadrics are used to cover the area. Their muscle model is based on a piecewise linear representation similar to the one developed by LEE et al. [LTW95]], where contraction is expressed by shortening the linear quadric segments. A muscle can either contract towards the end attached to the skull (linear muscle) or towards a point (circular muscle). Like real muscles, during contraction their muscles expand, the expansion is unrelated to the volume of the muscle however, and all muscles

simply have their height scaled up to a value of three upon maximum contraction. A mass spring network is used to simulate the layers of skin but unlike Waters et al. [TW90],[YLW93],[LTW95] they use a greatly simplified model combining volume preservation and interpenetration of skin and skull avoidance into one. They accomplish this by adding an extra spring to each mesh node that pulls the node outwards, mirroring the spring that attaches it to the bone layer. They explain this as a model of the internal pressure of a skin cell. Similar springs are added to mirror muscle attached nodes. This technique reduces the number of interpenetrations during simulation but does not stop them.

A generic mesh is pre-fitted with muscles and they adapt this to fit onto a scanned head. Thin plate splines are used to conform the generic mesh to the target mesh using many landmarks. They use 60 skin landmarks and 22 skull landmarks. The skin is deformed by direct application of the conformation method. Landmarks on the skull mesh are related to the skin mesh by vectors and when the corresponding skin vertex moves the skull landmark is moved too. The same warp is then used to conform the skull vertices. The muscles are transferred by applying the warping function to the grid vertices specifying their position and then rebuilding the muscle shapes. Albrecht et al. [AHK+02], [AH02] take this model and incorporate it in a system to create a text to speech which can gen-

erate sound and synchronised animation from text. As this model is the closest model to the one developed here, it is compared in detail and the differences highlighted in chapter 7.

2.4 Performance-Based Animation

The above sections cover the differing ways in which facial animation can be produced. Each of these methods is a tool for producing facial animation but they need some form of input to drive their movements in a realistic fashion.

The simplest input to a facial animation system is a user manipulating controls to get the desired animated result. This can be time consuming, and so often some form of automated system is used which will take text or recorded speech and turn it into a set of parameters and timings which will be fed into the animation system and an animated face will result.

The best results in facial animation have so far been from *performance-based* systems. These take as input some form of movements recorded from a real person and use this to drive the animation.

An early example of performance-based facial animation is seen in the short film *Tony de Peltrie* by Bergeron and Pachapelle [BL85]. To capture the movement of the actor, a polygon topology was painted onto his

face which was then photometrically digitized in a variety of expressions. The expressions were used to create a database of facial postures and a mapping was then created between the actor's face and the exaggerated caricature face.

Terzopoulos and Waters [TW93] used snakes [KWT88] to track facial features such as the eyebrows and lips in image sequences. From this data they estimated muscle contraction parameters and used this to drive their facial animation system.

The creature Gollum mention in the previous chapter [Jac02] is an example of the good results possible with performance based animation. It was however a completely manual process. Each scene was acted out by an actor and the footage was then used by animators to painstakingly manually manipulate a model of Gollum into the same expressions and mouth shapes as were recorded.

Performance based animation is becoming more and more common. The film *The Polar Express* [Zem04] used a similar technique as was used for Gollum but motion tracking markers were used to automatically pick up some of the animation information.

Using an actor as a basis for computer facial animation may be becoming popular but the real problems is still the quality of the facial animation system used to produce the final results. If the animation sys-

tem is not up to the task of capturing the actors subtle facial movements in a realistic manner then the effort that went into the performance has been wasted as it will not be visible in the final result. For this reason, performance-based techniques are a very useful tool for bringing a character's performance into a facial animation system but the quality of the animation system itself is the key to lifelike computer facial animation.

2.5 Extending Model Realism With Wrinkles

Facial animation systems generally deal with modeling the structures of the face including the skin but generally do not deal with the fine lines and wrinkles of the skin during muscle movement. For this reason, it is worth briefly covering the work that has been done in bringing that extra layer of realism to a model by adding wrinkles to the skin surface.

In 1992 Viaud and Yahia [VY92] presented a method for simulating wrinkles in facial animation and transferring the wrinkle system to any face model. They start with a cardinal spline surface which the Isolines are aligned with all possible expressive wrinkles in the face. This mask is then flattened onto a plane using cylindrical projection. To fit this

generic mask to any model, the target model has reference points such as the corners of the eyes and mouth marked. These points are then cylindrically projected onto a plane. The planar projected spline mask then has these points moved to match the characteristic points and the rest of the mesh is moved to fit around these points using a relaxation method. The control points of the planar spline mask are connected with springs whose rest length is determined by the natural location of the control points. Due to some of the points being moved to incorporate the position of the target characteristic features, some springs will be stretched or compressed. As these springs move towards their rest lengths they force the rest of the mesh to relax in a around the fixed characteristic points. The spline surface can then be projected onto the surface of the target face. Wrinkles are created by bulging the surface along the Isolines where the skin is contracting.

Boissieux et al. [BKMTK00] detailed three different methods to simulate wrinkles on the subjects. The first was an image based process. This process involved altering the face texture by means of darkening the image along the lines where wrinkles would lie. The location of these lines was determined from one of 8 masks which were defined by the type of face being simulated. The criteria for matching a mask to a face is based on gender, face shape and whether the person smiles a lot or not.

The age of the person is taken into account and the older the person, the darker the wrinkle lines.

The second method outlined in the paper employs bump or displacement mapping to show wrinkles in the skin. To define the size of the wrinkle displacement, their system tracks the shrinking of the skin along predefined muscle lines. As the skin area reduces perpendicular to the line, the amplitude of the wrinkle increases.

In the same paper, Boissieux et al. detail a method for actually altering a skin model to simulate wrinkles. Their results are demonstrated on an abstract simplified piece of skin rather than an actual face simulation. Using a simplification of muscles as two end points being moved. This movement causes the FEM skin model to bulge up or compress down as the the end points move further or closer together. By using a sinusoidal interface between the two simulated skin layers similar to real life, more realistic wrinkles form in the surface of the FEM model. The model also has the concept to permanent wrinkles, where the tissue has memory and plastic effects are introduced where the skin slowly adapts to deformations and maintains some part of the deformation after the compression force has been removed.

Bando et al. [BKN02] simulate fine scale and large scale wrinkles over the whole body. Fine scale wrinkles of the type that cover the whole hu-

man body are created from a user defined direction field. The mesh is projected onto a flat plane and the user selects various vertices and defines the direction that wrinkle will flow. The direction field is created from interpolation from the vertices the user specified to all other vertices so that each vertex has a direction associated with it. Fine scale wrinkles are created across the mews from any point in the direction of this field. These fine scale wrinkles can connect and merge or terminate when the meet each other just like the fine wrinkles in real skin. Large scale wrinkles are more important for facial animation and Bando et. al's model creates these in a different manner. The user must specify the wrinkle locations on the mesh. Wrinkles are defined as Bézier curves on the surface of the model. The cross sectional wrinkle shape is defined by a simple function that has parameters for width and height. To actually cause the vertices to move into the wrinkle shape, the vertices are displaced along their Normal vectors according to the wrinkle shape function. If the mesh is not fine enough to model the wrinkle then the mesh is adaptively refined in the area of the wrinkle. The depth of the wrinkles is dynamically modulated by the area of the triangles in which the wrinkle lies. Large amounts of shrinkage in the perpendicular direction to the wrinkle causes the wrinkle to furrow deeply.

2.6 Conclusions and Discussion

There has been a wealth of research in the area of facial animation. Realism and quality of results varies greatly between the different approaches. Image based methods produce very *video real* results but are restricted in that the view point can not be changed and every subject requires considerable data to be stored for animation. Multi-layer mass spring models have so far produced the most realistic results in three dimensions. These models create a full 3D model which can be manipulated, deformed and transferred into any situation imagined. Even the most complex of these models is still a long way from perfect, they are all still clearly computer generated.

This thesis extends the previous works by introducing a new muscle model which preserves volume during contraction and manipulates a rigid body jaw which can realistically handle chewing. The muscles and skull can be automatically transferred to any new skin surface model using a small number of sample points selected on the skin by the user. The skin is simulated using a new multi-layer mass-spring model which attaches automatically to the muscles underneath and deforms as the muscles and jaw move.

The next chapter will give an overview of the developed system, ex-

plaining what each part does and how the different parts relate to each other.

3. System Overview

Several software programs were developed during this research. These programs form a suite of software which allows a user to easily build up a network of muscles on any facial mesh or to take pre-defined muscles on a generic mesh and fit them to a new mesh. These muscles drive an anatomically based skin model and the software can output animation or export the resulting animated meshes to other software packages.

The workings of these programs will be explained in detail in the following chapters, but a brief overview of the whole system will be presented here to allow the reader to understand the flow of data through the software as it is explained later.

The developed software can be broken down into four main components.

- The muscle builder
- The muscle simulation system
- The skin simulation system
- The model adaption system

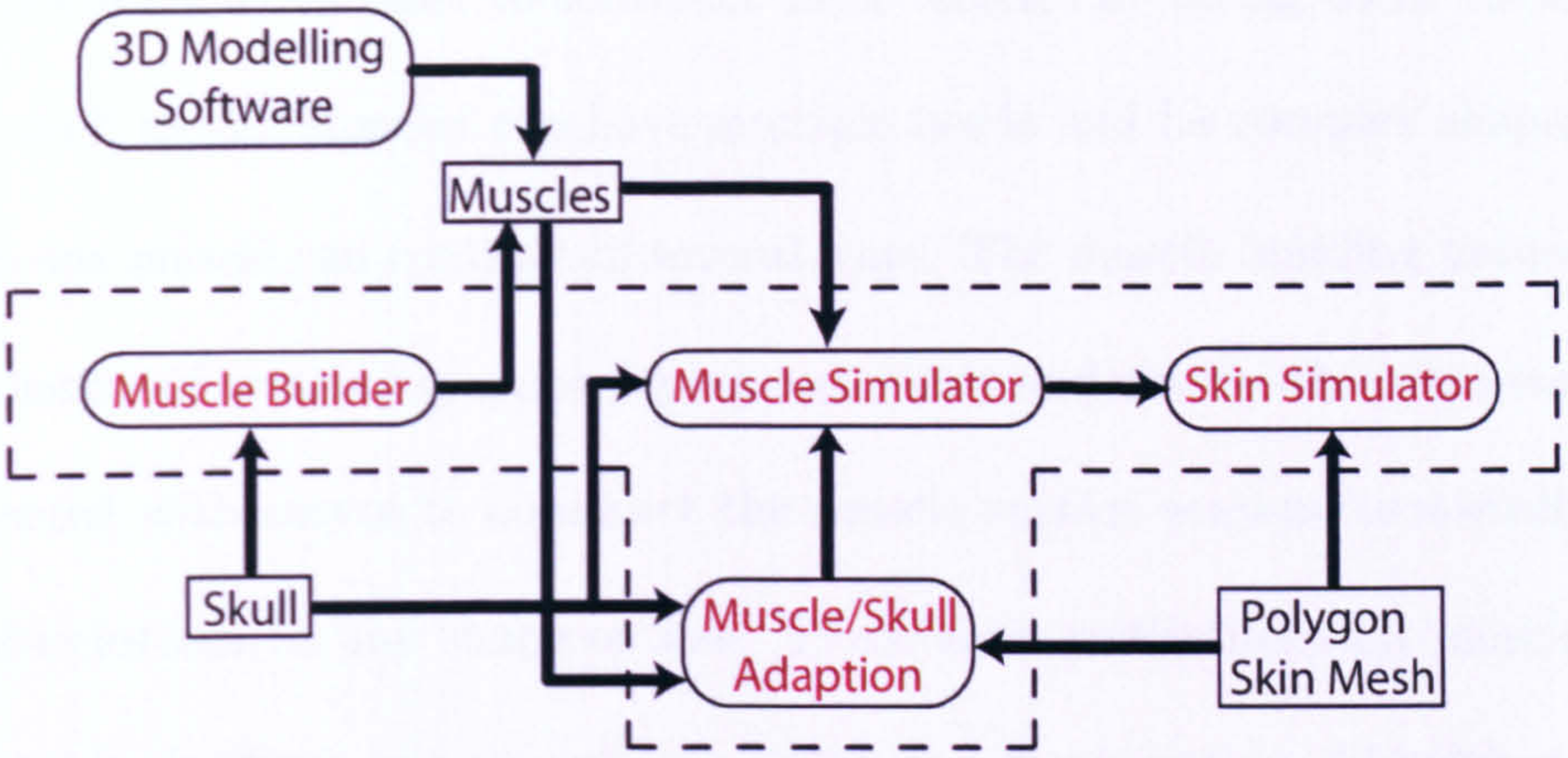


Fig. 3.1: High level overview of the simulation system

The developed software comprises four modules which are shown in red in the diagram above. The four parts and what each actually does are explained next. The muscle and skin simulation systems are actually part of the same piece of software but have been separated here so as to clearly explain the function of each.

3.0.1 Muscle Builder

Input: - A skull

Output: - A set of muscles and paths showing their contraction direction.

Muscles are defined in the system as polygon meshes with a path describing the direction each muscle contracts in. The muscle builder

system allows the user to construct these muscles by laying them out on a skull mesh. Muscles can have multiple heads and be complex shapes, so one muscle can contract in several ways. The muscle building process consists of creating muscles, laying out points and circles which are connected with curves to construct the muscle surface semi-automatically. Muscles can be any shape or size. There is no restriction that muscles need be fusiform or have only one origin and one insertion. Muscles can be defined which closely match those of the human body or would fit inside an imaginary alien creature. If the user prefers to create muscles in a modelling package, muscle surfaces can be imported into the muscle simulation system and the muscle's contraction path can be added in the muscle builder or imported along with the surface.

3.0.2 The muscle simulation system

Input - Muscles and skull

Output - Muscle and jaw animation

The muscle simulation system takes the muscles from the muscle modeller or an external package and simulates their contraction and interaction. The user can contract the muscles which will automatically preserve their volume and expand in cross section as they shrink in length. The

muscle simulation system handles all the low level details of muscle movements. It automatically connects up nearby muscles so that they move in unison, creating a realistic movement of the facial substructure. The muscle simulation system is also responsible for manipulating the jaw. As the muscles of mastication are contracted, the muscle simulation system applies forces to the jaw and makes it move in a realistic manner.

3.0.3 The skin simulation system

Input - Muscles and skull from muscle simulator and a skin mesh.

Output - Animation of the skin mesh.

The skin simulation is based on a multi-layer mass spring model which simulates the multiple layers of skin and their interactions with each other. It will automatically build up layers of connections between the skin and the underlying muscles and bones so that as the muscle and bones move and the skin moves correctly over the top. The multiple layers of skin each have individual properties and the movement of muscles pulls on these layers and the force is propagated through the skin layers to produce movement on the skin surface.

3.0.4 The muscle/skull adaption system

Input - Muscles, skull and skin mesh

Output - Muscles and Skull fitted into skin mesh ready for muscle animation.

In the adaption system, a skull and muscles that have been set up for animation can be easily fitted into any other head or face model. This means that once a set of muscles has been defined on a skull, there is no need to repeat the process as this muscle and skull set can be moved by selecting only a few key points on the surface of a new mesh. The system works by the user selecting specific points on the face surface which correspond to known average depths of facial tissue. The system then automatically warps the skull, and muscles to lie in the correct size and position under the mesh.

3.1 Conclusions and Discussion

This short chapter provided a very brief overview of the developed simulation system. It should have provided the reader with enough information to understand the data flow between different parts of the system and what function each part of the simulation system performs. The technical details will be covered in subsequent chapters starting with the muscle

system in the next chapter which covers the construction of muscles in a generic mesh, how these muscles are simulated during contraction and then how the muscles and skull can be easily transferred to a new skin mesh.

4. Muscle and Bone

Muscle and bone comprise the main parts of the face. When a person speaks or shows expression, the changes in the surface of the face are created by the movements of the underlying muscles and bony structures. When a person smiles, a group of muscles led by the Zygomaticus Major pulls the corners of the mouth diagonally up and to the side. When the mouth is opened, it is accomplished by the muscles of mastication pulling the mandible down and causing it to rotate around the temporomandibular joint, pulling the skin with it. Correctly modelling the muscles and bone of the face is key to creating believable animation. By modelling these structures and their movements correctly, the overlying skin should move accurately. This chapter details the muscle model developed to simulate these complex features.

4.1 The Muscle System

The human face can produce an almost infinite number of expressions and as such, finding a computer model to successfully simulate them has proved problematic to researchers for over thirty years. Facial movements

are produced by the muscles of facial expression and mastication pulling a multi-layered skin over a bony subsurface. The movements of the muscles themselves can be quite complex; facial muscles can be intertwined and interconnected so that the movement of one muscle can affect several others.

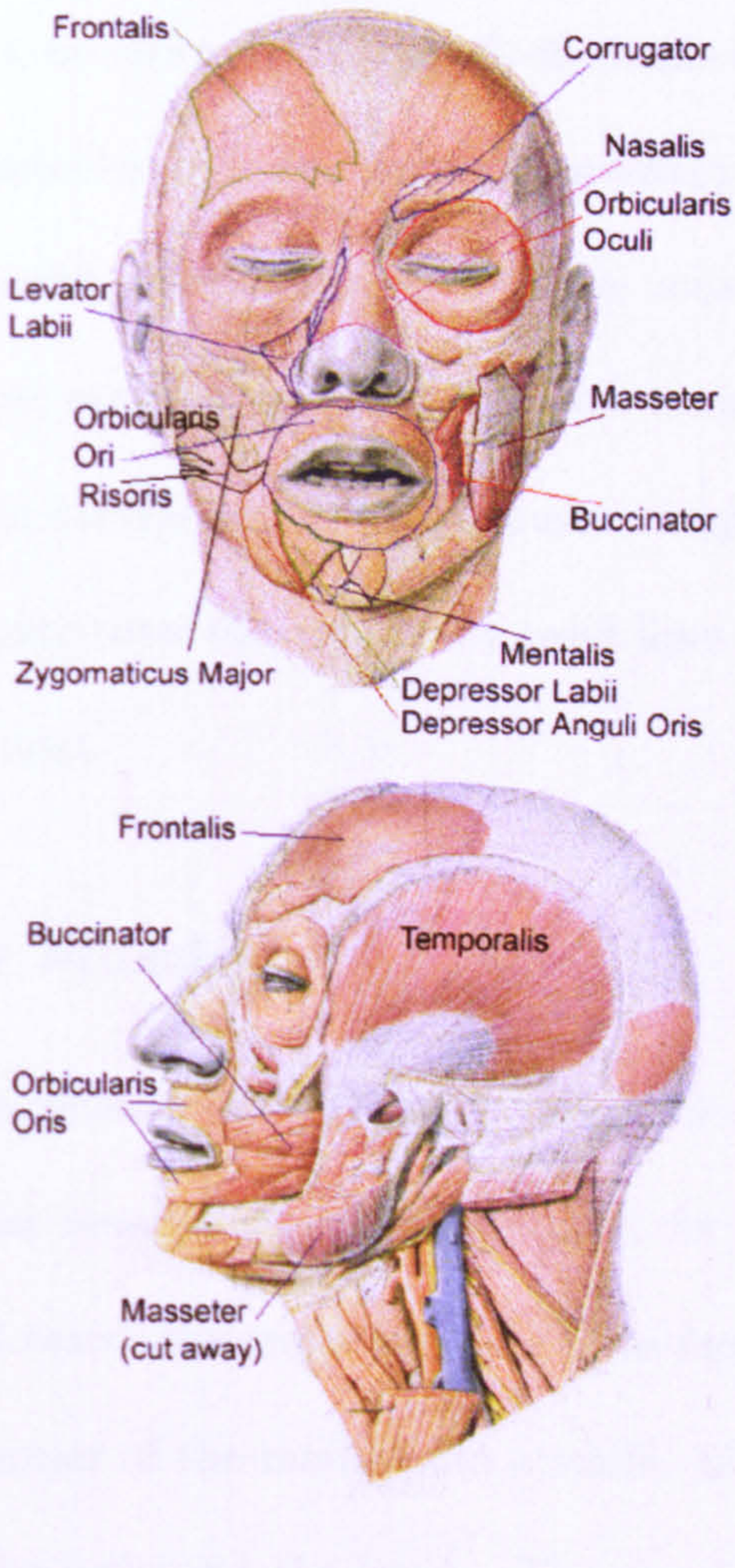


Fig. 4.1: The muscles of the face.

The muscles of the face comprise a large range of shapes and sizes as can be seen in figure 4.1. Previous facial animation work has often used greatly simplified shapes and contraction models to mimic the complex movements and interactions of the facial muscles. Most facial animation research models muscles as line segments [Wat87][PW96], ellipsoids [Dip91] or, at best, quadrics [KHYS02]. Such simple shapes cannot represent real facial muscles with any degree of accuracy.

The muscle model described here simulates muscles accurately including their shape, position and change in size during contraction. To model the realistic contraction of these muscles requires a multi-stage model more complex than the commonly used lines or quadrics. This model is detailed next.

4.1.1 Linear Muscles

In the human body, muscles can be broadly classed into linear and sphincter muscles. Linear muscles are pulling muscles, for example the bicep pulls the forearm towards the upper arm and in the face, the Zygomaticus Major pulls the corner of the mouth into a smile. Linear muscles comprise two parts, the belly and the heads. The muscle belly is the large section which expands out and shortens in length during contraction.

The *heads* of a linear muscle are the parts which attach onto something. Muscles can have multiple heads, for example the *tricep* has three heads and the *bicep* has two. The *origin* of a muscle is the end that the muscle contracts towards, the other end which is usually pulling bone or skin is the *insertion.* Sphincter muscles are circular and contract towards a central point. The muscles around the mouth and eyes are examples of sphincter muscles.

Real human muscles come in a number of shapes and sizes and often follow curved paths as they flow over and under bony structures. Upon contraction, a muscle may cause a rotation in a joint and thus need to follow the rotational path of the attached bone. This path cannot be faithfully represented with a line or line segments. In the face, as a muscle contracts in a curve over bone, it draws the skin along this curved path. The skin, which is what an observer actually sees, is also greatly affected by the size and shape of the underlying muscles. Systems which model muscles as simple Line segments cannot hope to capture this way in which muscles move and change in shape as they contract.

The muscles developed here overcome all of these problems. These muscles have three components:

- A polygon mesh defining the muscle surface.

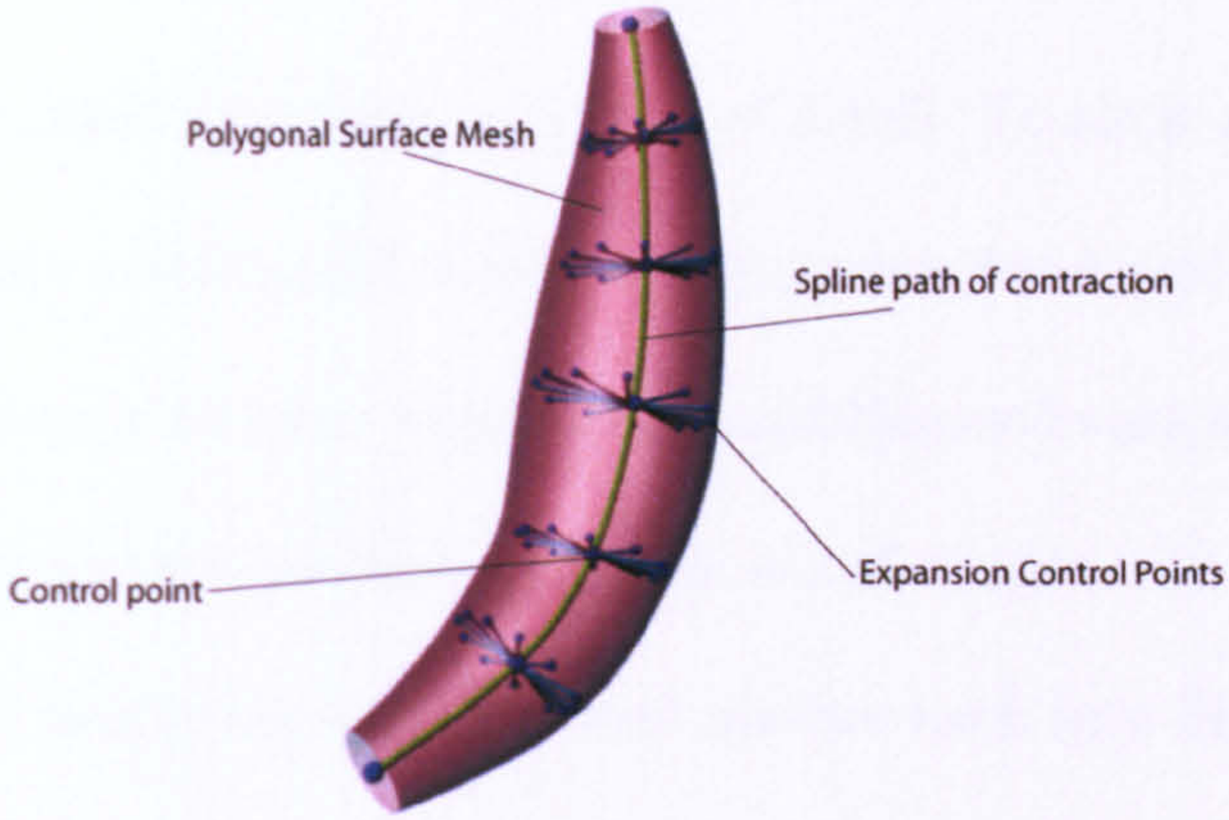


Fig. 4.2: A muscle and its fibre

- A set of control points which move the surface during contraction.
- A Spline path describing the path of contraction of the muscle.

These three muscle parts can be seen in figure 4.2. The construction of these muscles will be explained first, before detailing how muscle contraction is performed.

The muscles developed here are not modelled using a volume representation, but they act like a solid object in the way they are deformed during muscle contraction and during skin muscle interaction. The polygon mesh comprising the muscle surface can be created interactively in the developed software or in any 3D modelling program. The only restriction on the mesh is that it should be polygonal in nature and the

faces should be triangular. This is a minor restriction as any 3D model can be triangulated with little or no loss of detail. To allow muscle models to be quickly constructed, a set of scripts were developed which allow models to be exported from Alias' Maya modelling software to the muscle design and simulation package. Another set of scripts was also created to import the animation and deformed meshes back into Maya, so they could be incorporated into larger scenes or rendered out using a high quality renderer such as Mental Ray or Photo Realistic Renderman. An interactive muscle editor was also developed. Its usage will be detailed after the muscle contraction graph as it is closely tied to the construction of these graphs.

The Muscle Contraction Graph

The many different shapes of muscle in the human face can contract in any number of possible ways. If a muscle is not a simple fusiform shape then it is not obvious how that muscle should shorten in length.

Figure 4.3 shows the muscles of facial expression. These muscles are those which are primarily responsible for moving the upper parts of the face when creating the expressions of emotion and speech [Mus02], [MHL95], [GHH+96]. The lower parts of the face are moved by the muscles of mastication and will be dealt with later. The diagram clearly

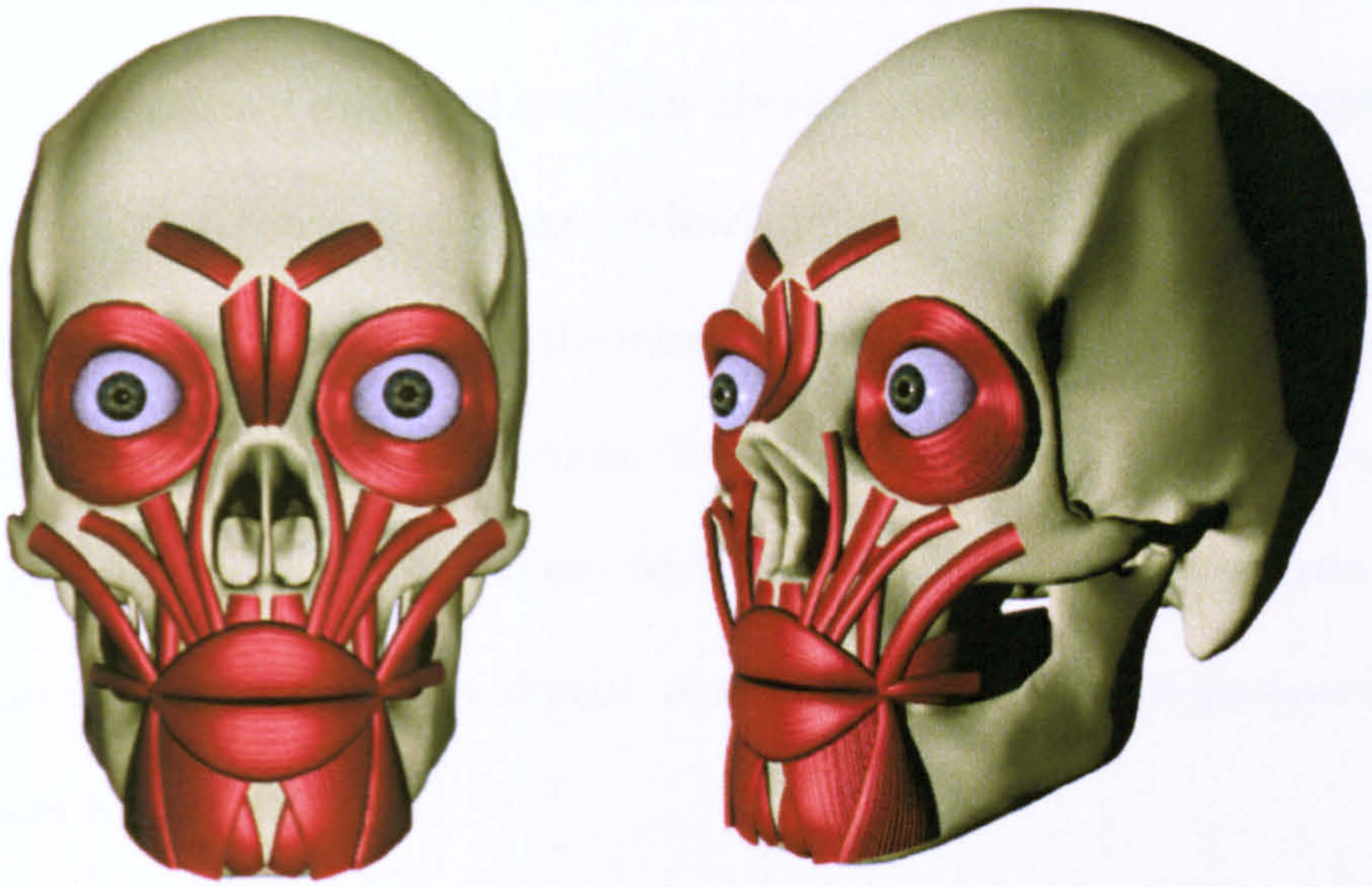


Fig. 4.3: The muscles of facial expression

shows the variety in size and shape of the muscles of the face. How each of these muscles would shorten during contraction is not clear. As muscles contract, they also expand in cross section, giving a familiar bulge. Many of these muscles have shapes which do not have an obvious direction in which their cross-section would expand during contraction. As was mentioned earlier, several muscles of the body have more than one head and so connect to bone or skin at more than one point. This produces a complex muscle shape which contracts from and towards more than one place, this is commonly ignored in muscle animation. In most work, muscle shapes are usually simplified by choosing rough mid-points

for the origin and insertion of the muscle somewhere between the several heads, and then a simplified muscle is created around this line. The entire muscle is then contracted along the line's length. This simplification loses important information about the muscles' movements and how the muscle changes shape during contraction. To allow the simulation of muscles as complex as those of the human body, and to allow artists free rein in their creature and muscle design, a more advanced muscle contraction model is required.

In the developed system, to specify the path a muscle follows during contraction, each simulated muscle has a group of spline curves which show the general direction that all areas of the muscle contract in. These paths can be constructed in a modeling program along with the muscle mesh or can be interactively generated in the software by a user. To construct the paths, a set of control points (CPs) is created within the polygon mesh of the muscle. All but one of the control points in each path is assigned a parent control point. The connections between control point and parent form a graph structure which, when connected, form piecewise line segment contraction paths. To create smooth contraction paths, the control points are duplicated and the duplicated points become the knots in Catmull-Rom interpolating splines [CR74].

The splines are created starting at control points with no children

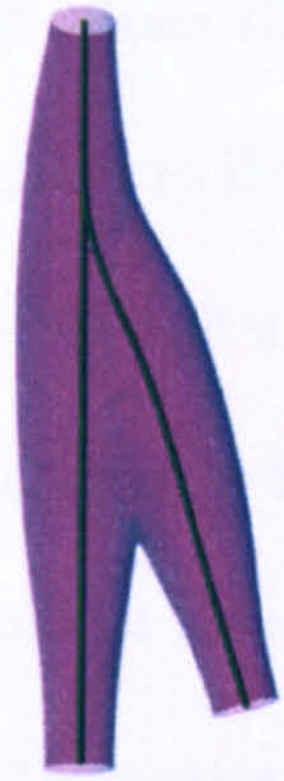


Fig. 4.4: A muscle with multiple Spline paths

working their way up to a control point that has no parents. This can lead to splines which overlap such as in figure 4.4, however this is not a problem as the splines share knots where there is overlap and so any difference in spline paths is very small.

At this point, the control points which were duplicated to produce the knots are redefined in parametric terms of the spline curves. Each control point is defined initially by the knot that it produced, the and distance from that knot towards the next knot (0..1) which is initially zero. This has the effect that when the splines are moved or reshaped by moving knots then the control points are moved automatically.

The movement of the contraction CPs is what actually drives the movement of the muscle surface and so this redefinition is an important

feature of the muscle model. It allows the spline curves to be affected by the bones they are attached to and so if one muscle affects the movement of a bone then all the spline paths of all muscles attached to that bone will automatically move and may change in length. As these CPs are defined on the curves, they will spread out or squash together automatically as the curve changes length and thus cause the muscle to change in size.

It is worth noting at this point that each muscle has a second set of control points which deal with the expansion of a muscle's cross section during contraction. These CPs are those which radiate out perpendicular to the main CPs in figure 4.2, and are what cause the muscle to bulge out during contraction. They will be dealt with after the main muscle contraction has been explained fully, but are mentioned here so that the diagrams showing muscle contraction do not contain any unknown objects.

The spline paths detail the direction the muscle contracts in, however they do not define how the surface should move during contraction. This is defined by the relationship between the muscle control points and the surface mesh vertices. The control points are split into two categories, tendon and belly. Real muscles consist of a belly section and connecting tendons. Muscles attach to bone through tendons which do not change length during contraction. During a muscle contraction, the muscle belly

shortens which pulls the tendon which in turn causes a bone to move or rotate. Face muscles do not have tendons as such and just merge into skin and attach to bone, but this muscle model is general enough to simulate any muscle in the body and as such tendons can be modeled.

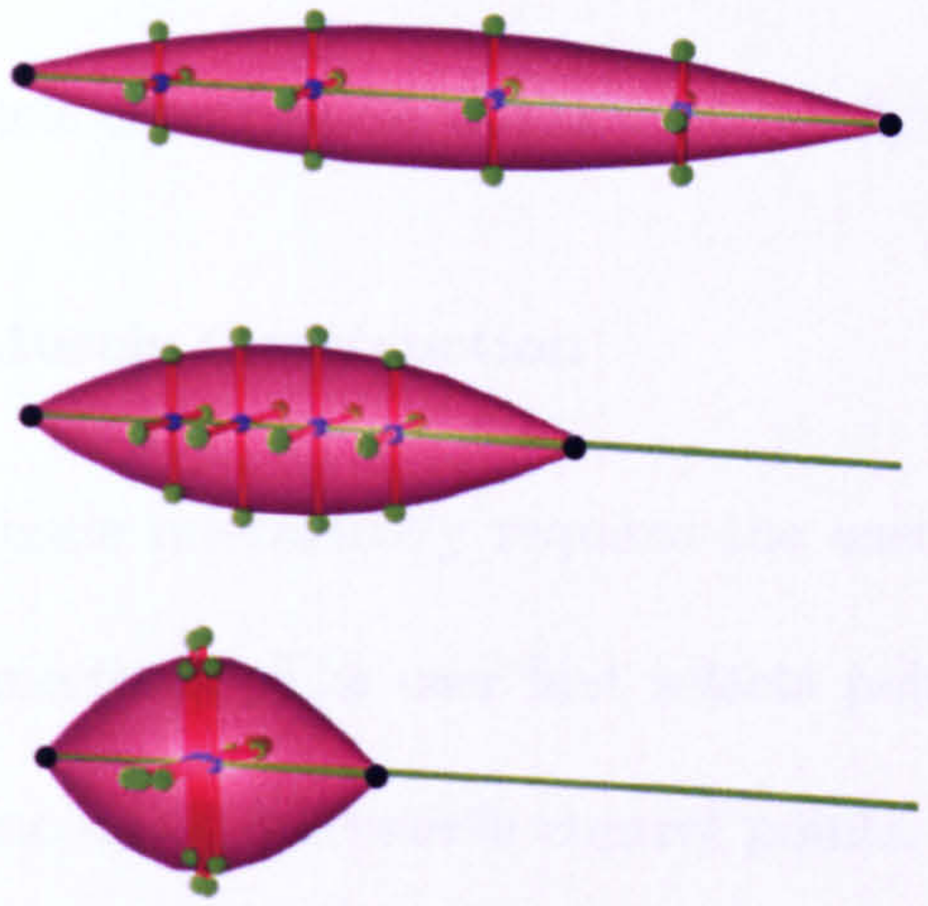


Fig. 4.5: A muscle contracting and expanding in cross-section to preserve volume. The control points are shown as oversized spheres for clarity. The black spheres signify tendon and thus these sections do not change in length.

During a muscle contraction, the spline paths do not move. Instead all of the muscle control belly points contract. When ordered to contract, these control points move up the spline curve in the direction of their

corresponding spline control knot's parent. When the muscle is again relaxed, the belly points move down the curves toward their original position and even beyond if the muscle is stretched. This contraction process is shown in figure 4.5.

With the muscle paths and control points defined, the interactive method of muscle construction will now be briefly explained.

Interactive Muscle Construction

To create a muscle interactively requires the user to lay out its rough shape and connections. The user first selects points on the bony skull surface to define all of the muscle control points. The user selects the parent point for each control point as the path is constructed. By connecting each point to its parent up to the end of the path, one or more control graphs will be constructed for the muscle. Once they have been created, all of the points can be selected and moved around in 3D using the mouse to position them roughly in the centre of what will be the muscle.

When these points are satisfactorily positioned, a set of circles is created by the system at each point entered previously. The circle for each point p is oriented perpendicular to the vector from p to p's parent. This stage can be seen in figure 4.6.

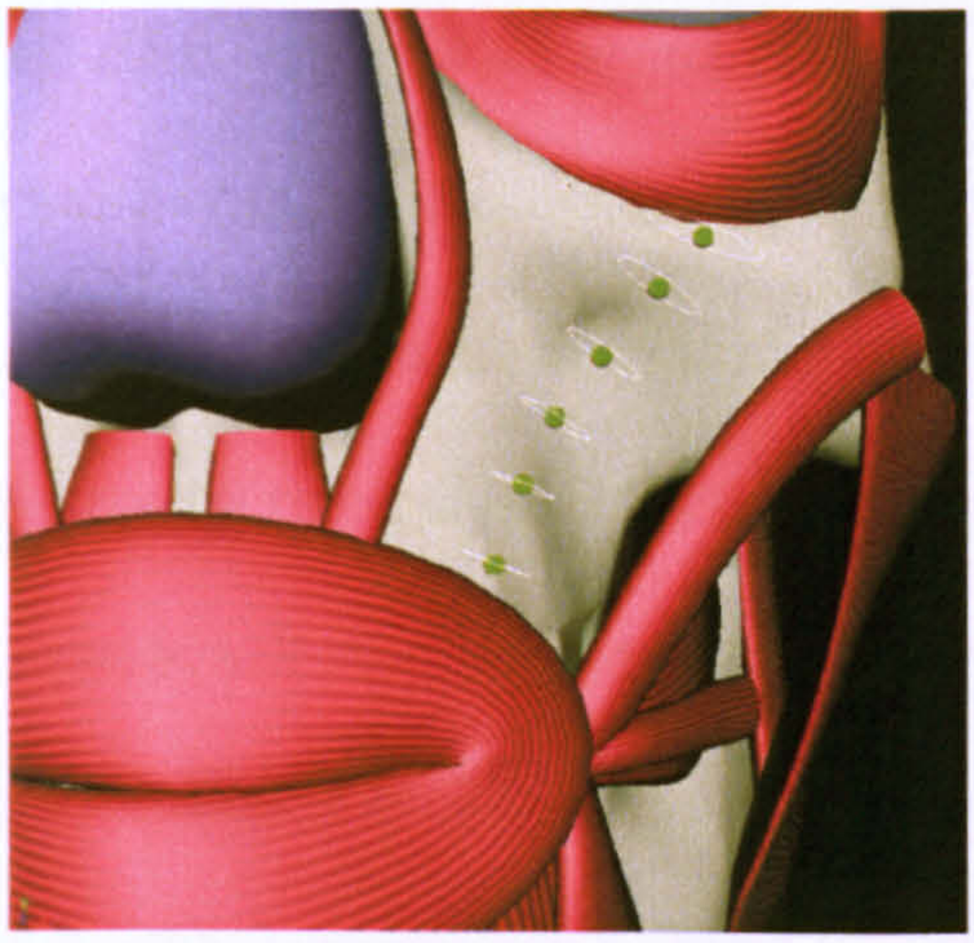


Fig. 4.6: Interactive muscle construction

The user can scale or move each of these circles to lay out a more precise shape for the muscle. More circles can be added between any two circles if required for more accuracy. When the user is satisfied with the muscle shape the circles are converted to piecewise line segments and the circles are all connected up to form a triangular mesh.

The constructed mesh can be quite coarse, which is undesirable for two reasons. Firstly, a rough mesh with sharp edges is very unlike a real muscle and will tend to poke the skin at the points where its vertices stick out. Secondly, the skin connects to the muscles by attaching into the triangles comprising the muscle surface. The fewer of these triangles there are, the more skin points connect to the same triangle, which means that large portions of the skin surface could be being moved by the same

underlying triangle. This can lead to visual artifacts when the muscle contracts, as large parts of skin will move in the same direction uniformly whereas the large area next to this may be moving in a different direction. The multi-layer skin model reduces this effect, but having more triangles in the underlying muscle will stop the problem from ever arising.

To create a smooth muscle model, the mesh is turned into a subdivision surface. Subdivision surfaces are a useful tool in defining smooth, continuous surfaces from meshes with arbitrary topology. To create a subdivision surface from the muscle mesh is a two stage process. The first phase, the *refinement phase*, creates new vertices and reconnects the mesh to create new, smaller triangles. The second *smoothing phase* computes new positions for some or all of the vertices in the mesh.

There are several subdivision schemes in existence and each has different ways of accomplishing the two phases based on differing sets of rules. Subdivision schemes can be characterised as approximating or interpolating. Approximating schemes move every vertex to a new position during the refinement stage; interpolating schemes only move vertices which were created during the last refinement phase.

The method used here is the modified butterfly scheme by Zorin et al. [ZSS96]. It is a modification of the butterfly scheme by Dyn et al. [DLG90]. This scheme is an interpolating scheme which is desirable

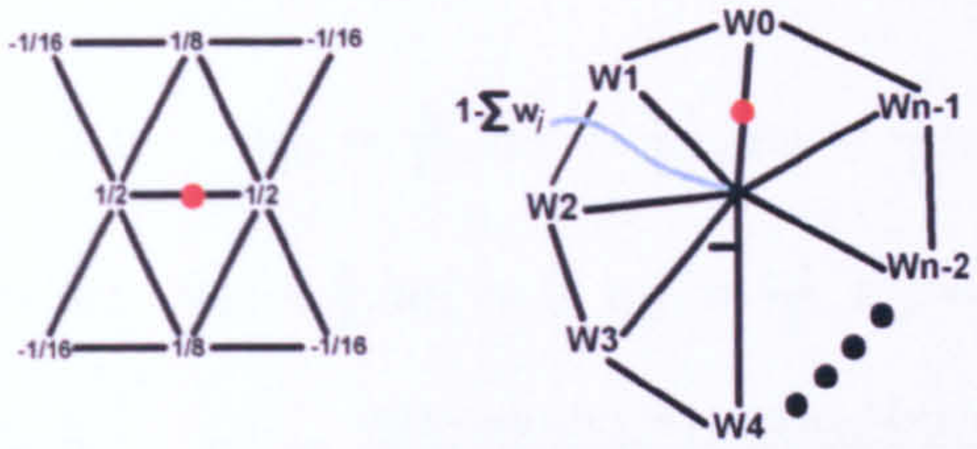


Fig. 4.7: The mask on the left is the butterfly mask used between two regular vertices and the mask on the right is used for a semiregular situation. The red circles indicate where a new vertex is being generated.

as once the user has set the initial mesh vertices, these should not be altered. The modified butterfly scheme uses four different rules to create the vertices of the next level of the mesh. These rules are described below and the corresponding masks are shown in figure 4.7.

Regular Setting: If generating a new vertex between two vertices with valence six, these vertices are regular and use the mask to the left of figure 4.7

Semiregular Setting: If one of the vertices being examined is regular and the other is irregular (i.e. valence $\neq 6$) then the following formula are used.

$$n = 3: \quad w_0 = \tfrac{5}{12}, w_1 = \tfrac{-1}{12}, w_2 = \tfrac{-1}{12},$$

$$n = 4: \quad w_0 = \tfrac{3}{8}, w_1 = 0, w_2 = \tfrac{-1}{8}, w_3 = 0, \tag{4.1}$$

$$n \geq 5: \quad w_j = \tfrac{0.25 + \cos(2\pi j/n) + 0.5\cos(4\pi j/n)}{n}$$

Irregular Setting: When both vertices of the current edge are irregular, a new vertex is temporarily created for both of the vertices using the formula from the semiregular setting. The average of these two new vertices is used as the new vertex. This will only happen at the first subdivision stage as after the first subdivision, the mesh will consist of only regular and semi-regular vertices.

Boundary Setting: At boundaries in the mesh where an edge has only one triangle connected to it, the weights below are used with the mask in figure 4.7.

$$w_{-1} = \frac{-1}{16}, w_0 = \frac{9}{16}, w_1 = \frac{9}{16}, w_2 = \frac{-1}{16} \tag{4.2}$$

Linear muscles are not the only kind of muscles in the face. Around the eyes and mouth are sphincter muscles, which are unlike linear muscles in that all points on the muscle contract towards a single point. These muscles and their simulation will be examined next.

4.1.2 Sphincter Muscles

Sphincter muscles are circular and contract towards a single point. In the face, sphincter muscles encircle the eyes and the mouth. For sphincter muscles, there is no need to define a full spline contraction path as every part of the muscle contracts towards the same point at the same time. A sphincter muscle is defined by a polygon mesh, a set of control points and a single contraction point.

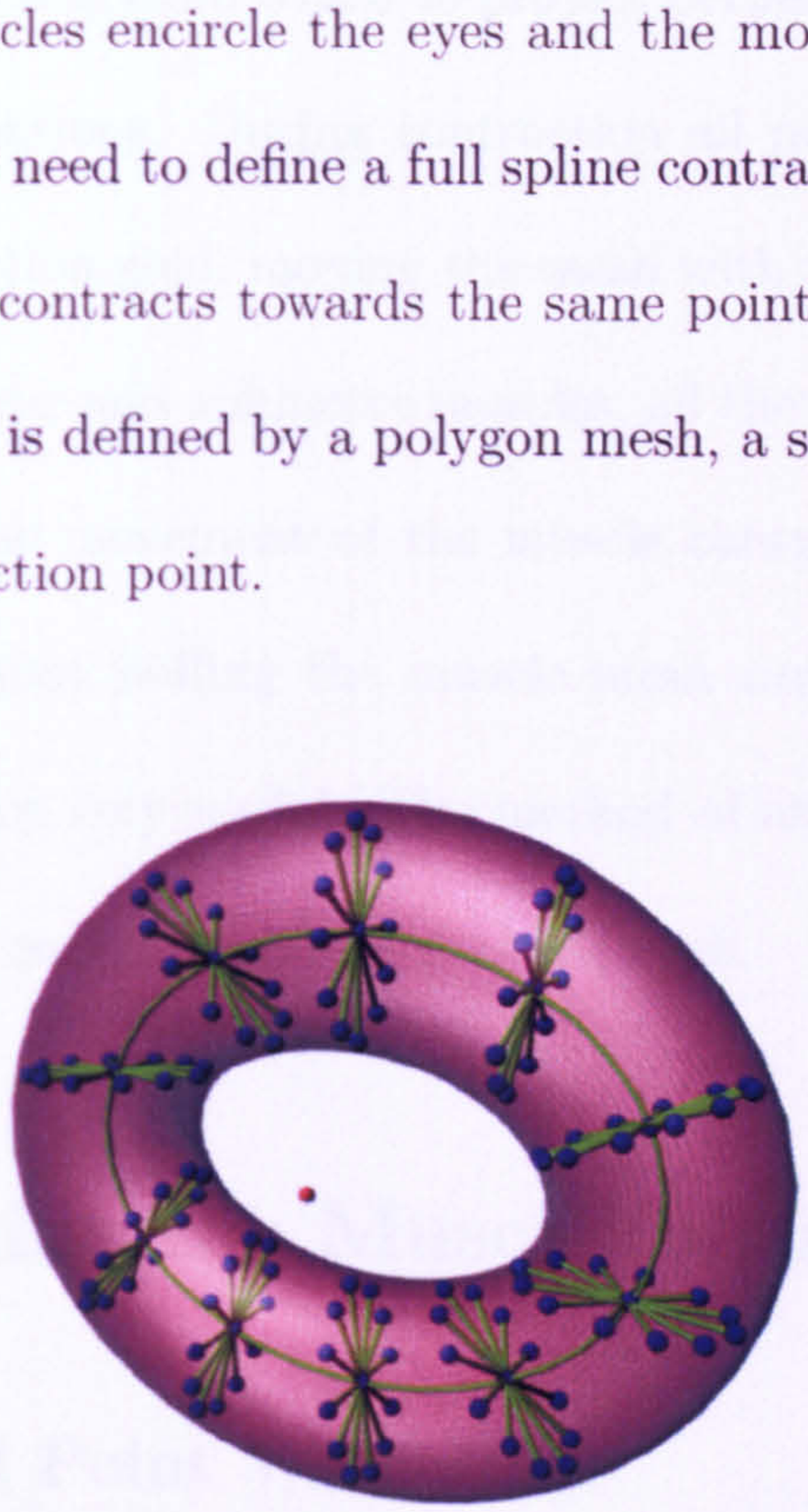


Fig. 4.8: A sphincter muscle, the red sphere is the contraction goal which all the CPs contract towards.

Figure 4.8 shows the parts of a sphincter muscle. Its makeup is similar to that of linear muscles. The circle of CPs is defined by the user and the expansion CPs are created by projecting radially, perpendicular from

the direction of a point to its parent. Although the points form a circle, there is still a construction order and each point has a parent. This is only used for a direction from which to project perpendicular to find the expansion point positions. During contraction all points move linearly towards the contraction goal, moving the mesh with them.

Through the linear and sphincter muscles, all the muscles of the face can be created. The movement of the muscle control points has been explained, but without pulling the muscle mesh surface along with it, these muscles are not very useful. The method of attaching the muscle mesh to the muscle control points is detailed next.

4.2 Attaching the Muscle Surface

4.2.1 Control Point Weightings

To make the muscle move along with the muscle control points, the muscle mesh must be attached to the fibre so that each vertex in the muscle surface must move with the underlying muscle control points. To achieve this, a mapping must be made between each vertex and the CPs so that the vertices of the mesh move smoothly with the CPs. To make the mesh move realistically as the muscle contracts, there needs to be some locality

in the mapping between mesh vertices and muscle control points. As all belly CPs contract simultaneously, considering CPs too far from the vertex position can cause undesirable movement in the mesh. For example, if a distant CP curves in a different direction from the the local CPs this could cause the mesh to twist possibly causing visual artifacts. Locality of influence is accomplished by giving each vertex a weight for every CP which specifies the amount which that CP will affect the movement of that vertex. This weighting is based on the distance from the vertex to the CP. The default setting is for all CPs to try and influence all vertices, but the user can specify a maximum number of CPs which should affect each vertex to improve the locality of deformations. If not all CPs affect every vertex, then the set of CPs S_i that will affect vertex i is selected before the distance weightings are assigned. This set is created based on distance in the CP graph to the vertex. The distance is calculated based on distance along the graph from the closest CP in each graph to vertex i. This is best understood by the example of an imaginary muscle in the shape of a hand where the CPs take up the bone positions. A finger should be bound to the closest CPs in the hierarchy, i.e. the CPs of that finger, not the closest CPs in world space which may include some of the CPs of adjacent fingers.

The effect a CP's movement has on a vertex falls off with the square

of the distance and is shown in equation 4.3 where $\omega^v_{S_i}$ is the weight for vertex v from control point i in the set of control points S which affect this vertex and $d_{S_i^v}$ is the distance from control point S_i to the vertex v.

$$\omega^v_{S_i} = \frac{1}{1 + {d_{S_i^v}}^2} \tag{4.3}$$

Various distance weighting factors were tested but the square of the distance gave the best results (this value is however configurable in the developed software). These weighting settings can also be altered for each muscle which allows muscles of greatly varying shape to have the best settings for each, rather than global settings that are perfect for none. Using this measure of weight, control points which are not quite close to a vertex tend to have a weighting of near zero which produces good results.

After the weightings on vertex v for each joint have been calculated the n weights are normalised by

$$\omega^v_{S_i} = \frac{\omega^v_{S_i}}{\sum_{j=1}^{n} \omega^v_{S_j}} \tag{4.4}$$

By normalising the weights no unwanted scaling is introduced during muscle contraction.

4.2.2 Moving the mesh during muscle contraction

To actually make the mesh move along with the muscle control points, the mesh vertices must be transformed by a weight scaled amount along with the control points as they move.

To move vertex V_w when control point p is moved, a local coordinate frame L_p is first defined at p. Vertex V_w is then transformed from world space into this local coordinate frame giving V_p. When p is moved the coordinate frame defined at it obviously changes and as V_p is defined in this frame, its position is altered automatically. When V_p is then transformed back into world space from this transformed local space, its final position has changed relative to the change of the frame at control point p. For each vertex, this process is carried out for all control points and the positions produced are weighted using the distance weighting factor described earlier. This produces a final position for vertex V moved by the weighted movements of all control points.

Creating the coordinate frames

To construct the local coordinate frame for each control point p, the local x-axis $\overrightarrow{L}_{p_x}$ is defined as the direction of the normalised vector from the position of p to its parent. Each control path has one CP which has

no parent and its x-axis is constructed as an average of the x-axes of all its children. The initial local orientation of the coordinate frame is irrelevant, however, it is essential that the orientation is found in the same way throughout so that vertices can be transformed into and out of the frame with no unwanted transformations.

To discover the y and z axes of the local frame, a matrix is created which will align the local x axis $\overrightarrow{L}_{p_x}$ with the world x axis $\overrightarrow{W}_x$. This matrix will hold the local coordinate frame for the control point in its columns.

To align $\overrightarrow{L}_{p_x}$ and $\overrightarrow{W}_x$ a rotation round the axis perpendicular to both must be performed. The axis $\overrightarrow{A}$ around which to rotate $\overrightarrow{L}_{p_x}$ is found by taking the cross product of $\overrightarrow{L}_{p_x}$ and $\overrightarrow{W}_x$. The angle θ to rotate by for alignment is then found using the dot product.

$$\overrightarrow{A} = \overrightarrow{L}_{p_x} \times \overrightarrow{W}_x \tag{4.5}$$

$$\theta = \arccos \frac{\overrightarrow{L}_{p_x} \cdot \overrightarrow{W}_x}{|\overrightarrow{L}_{p_x}| \cdot |\overrightarrow{W}_x|} \tag{4.6}$$

To create the matrix describing the local coordinate frame, a quaternion is first constructed from the axis A and angle θ. Quaternions are used in the construction of the transformation matrices as they allow a convenient notation for rotation around an arbitrary axis, however matrices are used to hold the final transformations as matrices compactly hold

the translation component as well as the coordinate frame being used, and are less computationally expensive than quaternions for transforming points.

A quaternion is created from the axis and angle using the standard quaternion definition below.

$$q = w + xi + yi + zk = \cos\frac{\theta}{2} + iA_x \sin\frac{\theta}{2} + jA_y \sin\frac{\theta}{2} + kA_z \sin\frac{\theta}{2} \quad (4.7)$$

Creation of the matrix is a matter of substitution. A unit quaternion can be expressed as $q = \cos\theta + \hat{u}\sin\theta$ where θ is the rotation angle, $\hat{u} = u_0 i + u_1 j + u_2 k$ and vector (u_0, u_1, u_2) has length 1. Rotation matrices take the form:

$$R = I + (\sin\theta)S + (1 - \cos\theta)S^2 \quad (4.8)$$

Using standard trigonometric identities the matrix below can be constructed which is used to create the transformation matrix from the quaternion created above.

$$T^o = \begin{pmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{pmatrix} \quad (4.9)$$

This produces the transformation matrix to take a vertex from world to local coordinate space. It does not include the position of the CP and so a 4×4 transformation matrix is constructed from the above 3×3

matrix by adding the translation of the CP which is multiplied by the rotation as this matrix is actually the inverse transformation matrix.

This produces a matrix which will transform a point from world space into the local space of a muscle control point. The local space of the control point in its original position will not change during muscle contraction and so the position of each mesh vertex V_w in local space for control point p can be precalculated:

$$V_p = V_w(T_p^o) \tag{4.10}$$

During muscle contraction, each control point p will move and as such the coordinate frame at that control point will move and so move V_p along with it. For each frame of the simulation, the final world position of each mesh vertex needs to be calculated to give the position of the muscle surface. To calculate this, the difference in position between the original position of V_p and the muscle contracted position of V_p must be applied back into world space.

To accomplish this, the coordinate frame for each control point p is recalculated each frame T_p^f. This 4×4 matrix will transform any point from world space into the local frame of control point p. So, using the inverse of this matrix returns any point in the local frame back into world space. As V_p is defined in local space, its position will change as

p's position changes and this change will be reflected in the movement of V in world space. The effect on vertex V by each control point p is weighted by the normalized weight ω_p for point p described earlier. This leads to the final position of a vertex V being calculated as:

$$V_f = \sum_{i=1}^{n} \omega_p (V_p (T_p^f)^{-1}) \tag{4.11}$$

This method of moving the mesh with control points allows the mesh to move smoothly as the individual control points change position.

This details the movement of individual muscles, but the muscles in the face cannot be handled completely separately as there is interconnection between muscles in the real face. This interconnection, where the movement of one muscle changes the position of muscles nearby, must be simulated to ensure realism of a model. This interconnection is explained next.

4.3 Connecting Muscles

4.3.1 Overview

The muscles of the face do not act independently of each other. Figure 4.9 shows three images of muscle contraction. The lower left image shows a muscle contracting independently of its neighbors and the lower

right image shows the same muscle contraction where it has influenced its neighbours' positions. The image on the left is completely unrealistic, only a few of the muscles in the human face can be activated independently and even these muscles when they pull the skin cause neighbouring muscles to move as a consequence.

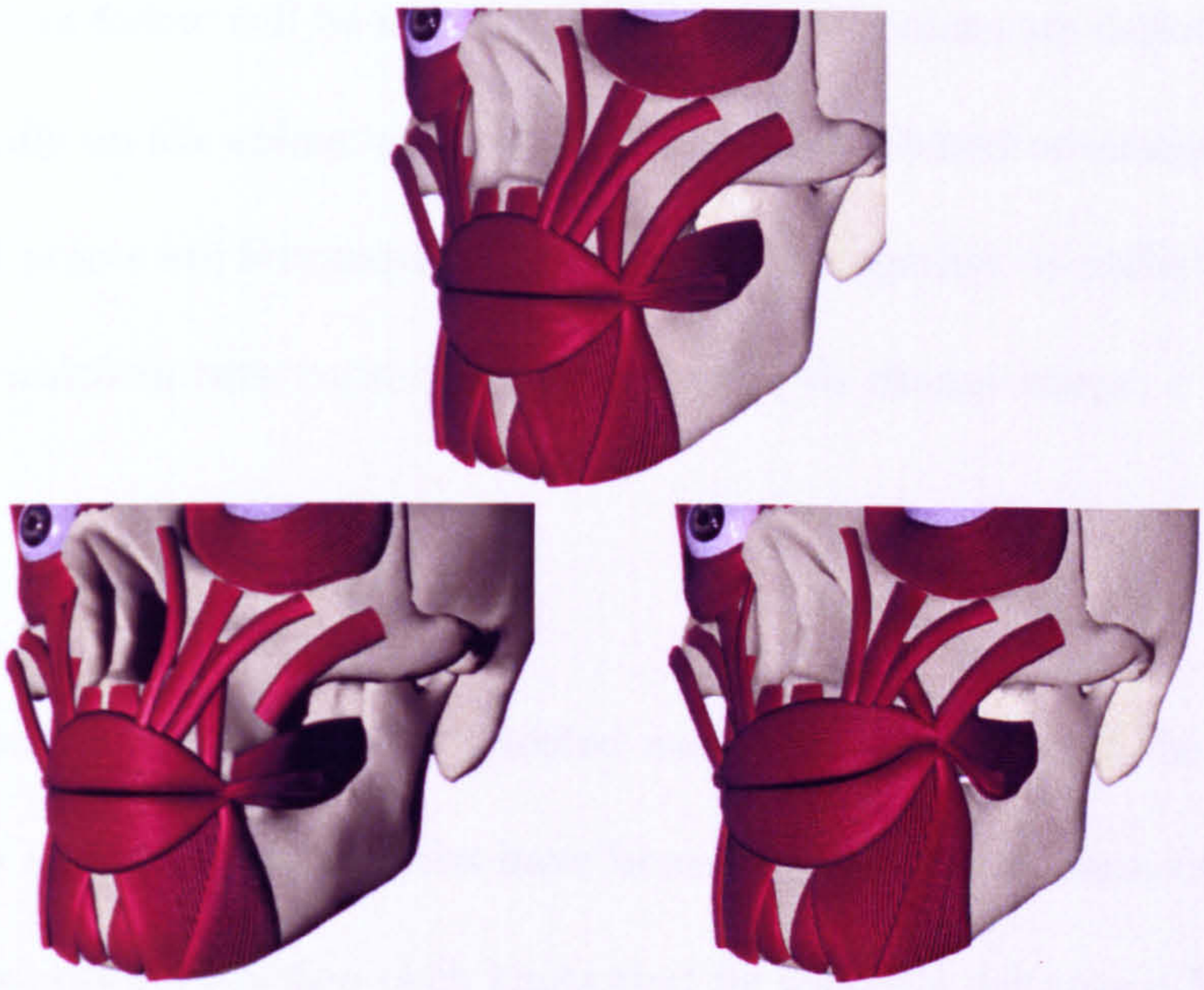


Fig. 4.9: Muscles at rest (top), a single muscle contracting - not connected to its neighbours (left) and connected to its neighbours (right)

To model this intertwined movement of muscles, springs are used to connect muscle contraction paths to neighbouring muscles. The control

points of each muscle are connected to the knots of nearby muscle spline control paths. When a muscle contracts, the extending spring will exert a force on the knot of connected muscles causing them to move. As the knots get pulled or pushed by spring forces, the spline flowing through them will remain smooth and thus the path of contraction that the control points follow will be smooth. As the control points are defined parametrically on the spline, as the spline path is shortened or elongated the control points are automatically pushed closer together or pulled further apart, which in turn causes the muscle mesh to change shape.

4.3.2 Implementation

Neighbouring muscles are connected automatically once all the muscle meshes and contraction paths have been defined. When connecting the muscles, any contraction path knots that lie within a distance d from the muscle head are specified as *connectable*.

All of the control points are then iterated over and any knots in the connectable set that are within distance c from each control point are then connected to that control point by a spring. Distances c and d are user specifiable and can vary from muscle to muscle depending on the muscle type and function. This process is shown in figure 4.10. These connecting

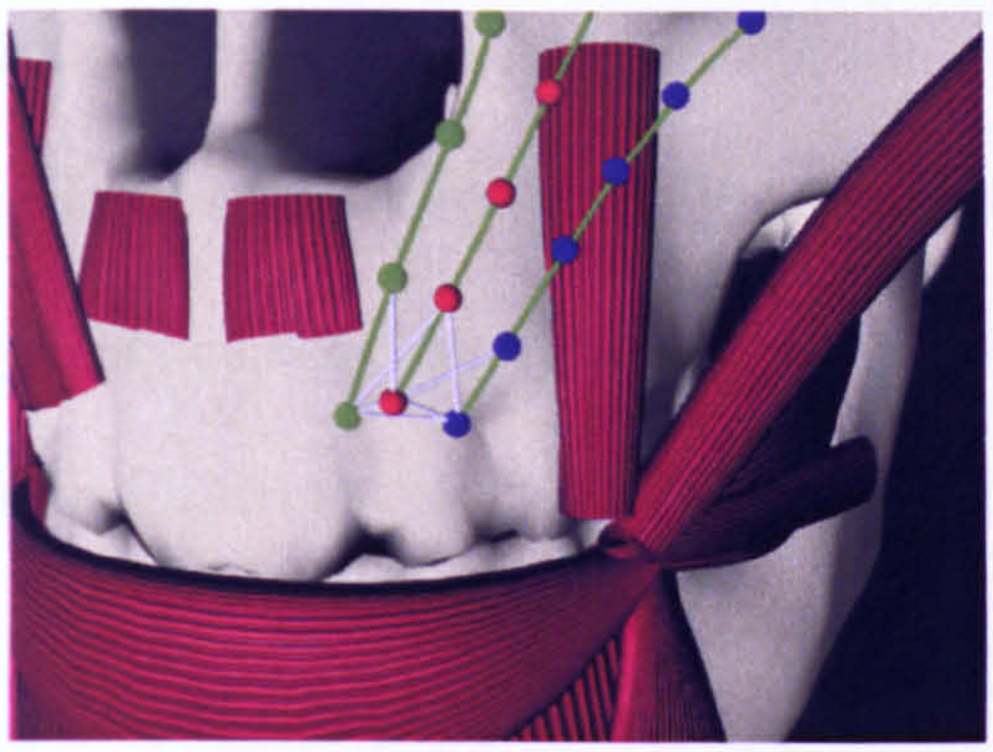


Fig. 4.10: Several muscles have been removed in this image to make it clearer. Three muscle contraction paths are shown in green, the control points are the coloured spheres and the grey lines represent springs connecting the ends of the muscles.

springs cause muscles to be dragged along when a neighbouring connected muscle is contracted.

The reason that only knots near the end of a muscle can be connected is so that the muscle model simulates real muscles correctly. Muscles in the human face sometimes slide over each other and lie close together, only where they merge at their skin end connections do they tend to actually move as one.

When a muscle contracts extensively it is possible that it could drag its neighbours far across the face, which in turn could pull other muscles excessively. This would lead to an unrealistic skewing of the facial muscles

where the whole face would appear to have slid in one direction. To alleviate this, each knot has a second spring attached to it controlling the pull of attached neighbour muscles. This second spring connects the knot to a point on the skull directly below its original position. As a muscle contracts, the force pulling the knot back to its rest position increases and counteracts the force pulling the knot towards its contracting neighbour. These rest position springs are emulating the natural connections of the muscle into the bone which help keep the muscle where it should be. These anchor points are found by tracing the y-axis of the local coordinate system at each control point down onto the skull. The triangle where it intersects becomes the anchor point. The world space position of this anchor point is described by the barycentric coordinates of the skull triangle the anchor lies in.

Springs cannot connect directly to knots as springs are a mathematical model of a real object and knots are simply points in space. Therefore, a point mass is created at the position of each knot which is to be connected with a spring. The knots are then attached to these point masses so that the knot's position in space is defined by the point mass. The actual mass of these point masses is defined by the muscle to which they belong and is determined by the muscle's volume. Small thin muscles should not pull neighbouring muscles nearly as much as large fat muscles.

Whereas large fat muscles should drastically affect their neighbourhood during contraction and so the larger the volume, the greater the mass of its knot point masses.

After the springs are set up, the masses are calculated using equation 4.12

$$m = 1 + \sqrt[3]{\frac{V}{S}} \tag{4.12}$$

The cube root of the volume is used to control the point mass of each knot in the muscle's control path. This value gives a good scaling for the masses which does not increase too rapidly as muscle volume gets larger. The division factor S is one by default and is user configurable. If the model being developed and simulated is exceptionally large or small, the volume of the muscles may be very high or small and thus the muscle masses may not be suitable for the simulation. The value of S is automatically estimated by averaging the muscle volumes and calculating a value of S to bring this average into the region where the simulation is known to perform well. The user can tweak this value if the system is not performing satisfactorily but this is rarely needed.

When a user wants to move a muscle from relaxed to significantly contracted, this could lead to moving the control points a large distance

in one step. This would cause the attached springs to change in length dramatically over a very short period of time causing huge forces to appear in the system possibly making it unstable. To counteract this, muscle contraction is done over a short period in a series of small steps. For each movement of the control points, multiple simulation steps are carried out for the mass-spring system. The maximum a point mass can move in one time step is controlled and at each step, positions, velocities and accelerations in each point mass and spring are calculated and distributed.

In the interests of continuity, the details of the physics used to model the springs are detailed in chapter 5 which explains the skin model. This is because the skin model makes extensive use of springs and it was felt that a thorough explanation of the workings of mass-spring forces fits better in that context.

The methods used to model, contract and connect the muscles of facial expression have been detailed now. The next muscle feature to deal with is expansion. As real muscles contract, they do not compress and so as their length shortens their cross sectional area must increase. The next section will explain how the expansion of muscles is accomplished.

4.4 Controlling Volume During Contraction

During muscle contraction, as a muscle shortens in length, it expands in width and height increasing its cross sectional area. This is often modeled in muscle simulation by a simple scaling in width and height.

As was shown earlier, muscles can be many shapes or sizes and simple fusiform or quadric models do not mimic the real thing well. The complex muscle shapes allowed by this muscle model will not preserve their volume by a simple scaling of their width and height.

4.4.1 Expansion points

To allow for muscle expansion, a new set of control points is introduced into each muscle which lie perpendicular to the main muscle control paths.

After the muscle shape has been constructed, at every control point a set of expansion points is created. These expansion CPs (ECPs) are created in a circle perpendicular to the direction of muscle contraction just below the surface of the muscle mesh. The first ECP is created by firing a ray directly along the y-axis of the local coordinate frame at each CP. This ray is checked for intersection with each triangle of the

surface mesh. When a point of intersection is found, the control point is positioned 95 percent of the distance to the surface. This control point is created parented to the path control point so that it is moved when the contraction path control points move. The distance of ninety-five percent was arrived at through testing and allows smooth scaling of the muscle shape. To create the rest of the circle of expansion control points for each CP, the y-axis vector is rotated around the local x-axis and an ECP is created every thirty degrees.

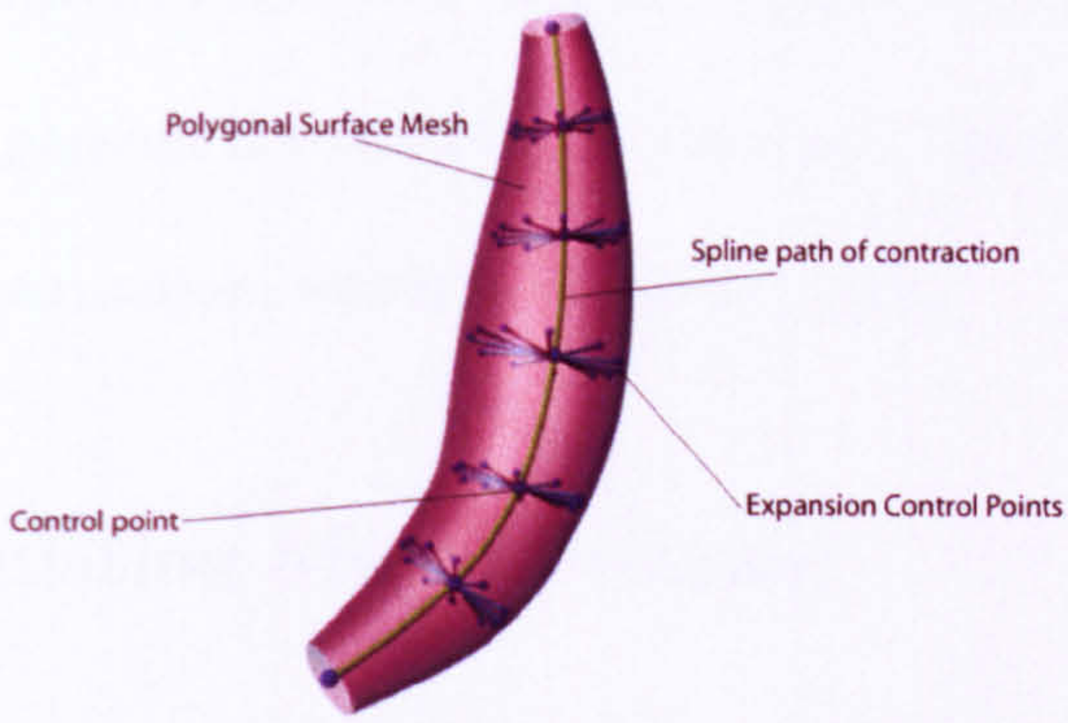


Fig. 4.11: A muscle and its control points.

This process is repeated for every contraction path CP that was designated belly by the user during muscle construction. After all expansion CPs have been generated, the control points for the entire muscle are in position, this is shown in figure 4.11

This ray-triangle intersection method of distributing expansion points can lead to some areas of the muscle receiving more points than others. For example, if the contraction path is closer to one side of the mesh than the other, or the mesh has very high curvature at one side, then the expansion points will be distributed unevenly. This is rarely a problem if the user specifies the contraction paths well, even if some areas are covered better than others it just means that the expansion in some areas will be more general than others. The number of expansion control points is user controlled and even very few points can produce excellent results as the expansion is controlled by the exact volume change in the muscle during contraction, which is explained next.

4.4.2 Calculating Muscle Volume

Calculating the volume of arbitrarily shaped muscles is not as simple as calculating that of simple geometric objects. The muscles here are modeled as polygonal models and calculating the volume of these objects requires integrating over the volume of the object. Calculating the volume, centre of mass and inertia tensor of an object are closely related and the latter two are required for simulation of the muscles of mastication described in section 4.5. The central definition of all three properties is

the three dimensional geometric moment: **Three-Dimensional Cartesian Geometric Moment** *The three dimensional Cartesian geometric moment* m_{pqr} *with order* $p+q+r$*, of a volume* V *with density function* g*:* $\mathbb{R}^3 \Rightarrow \mathbb{R}$ *is:*

$$m_{pqr} = \int_V x^p y^q z^r g(x, y, z) dv \tag{4.13}$$

To calculate the volume of a body requires calculating the zero order moment m_{000} with a unit density function. The centre of mass and inertia tensor are calculated from the first and second order moments and are detailed in the later section on mastication.

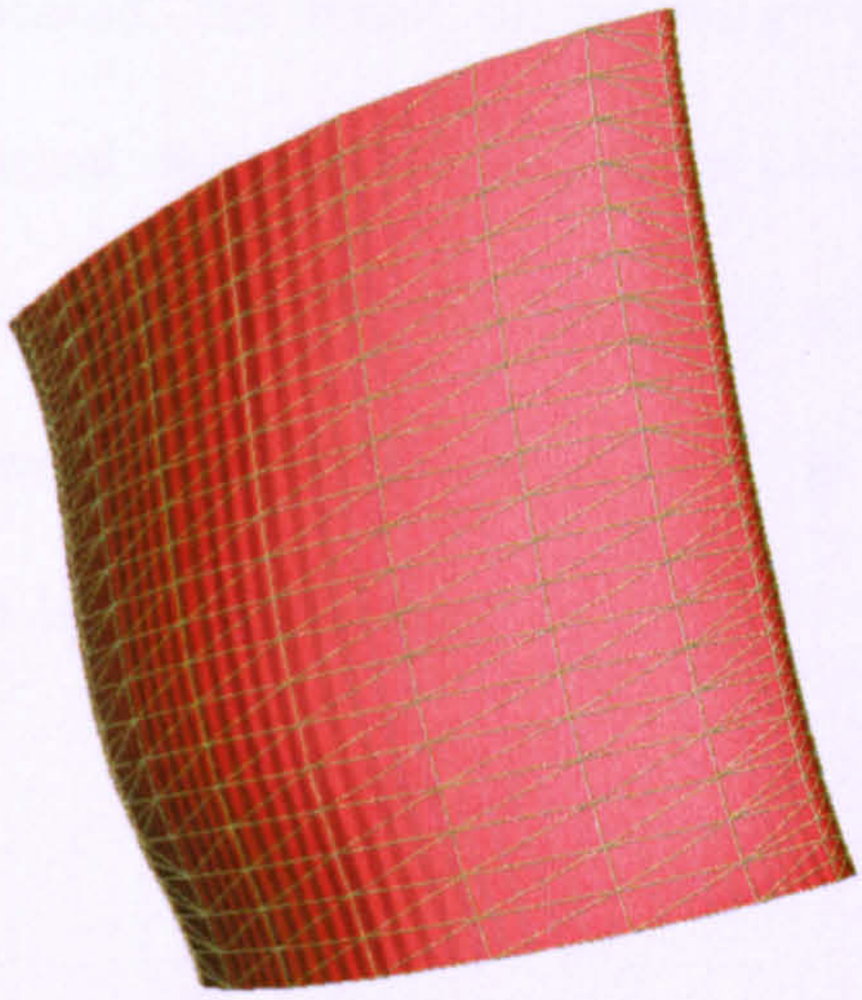


Fig. 4.12: A muscle with its triangular mesh visible

Figure 4.12 shows a muscle model composed of triangles. By taking

the three vertices of each face in turn and a point p, a tetrahedron may is formed.

The volume of the polyhedron can then be expressed as the sum of the volumes of the tetrahedrons. This method is proven in [LK84]. Any moment can be broken down in this way since all are integrals over the body volume of the polyhedron. The point p does not actually have to be within the body. If the point p is outside of a face of the polyhedron then the tetrahedron will have negative volume and thus subtract from the overall volume. This cancels out the excess volume contributed by the tetrahedrons on the opposite side of the body. It does not matter where point p is located, the result of summing the tetrahedrons will always equal an integral over the volume of the polyhedron.

To find the volume of a body, the volume of a tetrahedron Q must be calculated for every face of the object. Q is defined by vertices (v_0, v_1, v_2, v_3), which have coordinates:

$$\begin{cases} v_0 = (0,0,0) \\ v_1 = (x_1, y_1, z_1) \\ v_2 = (x_2, y_2, z_2) \\ v_3 = (x_3, y_3, z_3) \end{cases} \tag{4.14}$$

Calculating the volume of an orthogonal unit tetrahedron is simple and so a linear transform T is defined to transform Q into a unit tetrahedron.

$$\mathbf{T} = \begin{pmatrix} x_1 & x_2 & x3 \\ y_1 & y_2 & y3 \\ z_1 & z_2 & z3 \end{pmatrix} \tag{4.15}$$

which relates the old coordinate system (x,y,z) with the new (X,Y,Z) by:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1, x_2, x_3 \\ y_1, y_2, y_3 \\ z_1, z_2, z_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{4.16}$$

Using this transformation, Q is transformed into the orthogonal unit tetrahedron $W = (v_0, v'_1, v'_2, v'_3)$ with coordinates:

$$\begin{cases} v'_0 = (0,0,0) \\ v'_1 = (1,0,0) \\ v'_2 = (0,1,0) \\ v'_3 = (0,0,1) \end{cases} \tag{4.17}$$

The integral properties of the polyhedron Q can be described by :

$$
\begin{aligned}
m_{pqr} &= \iiint_Q x^p y^q z^r \, dxdydz \\
&= \|T\| \iiint_W (x_1X + x_2Y + x_3Z)^p \\
&\quad (y_1X + y_2Y + y_3Z)^q \\
&\quad (z_1X + z_2Y + z_3Z)^r \, dXdYdZ \qquad (4.18)
\end{aligned}
$$

The term $\|T\|$ is the absolute value of the determinant of the matrix T. To simplify the notation, define $c(i,j,k)$ to be the coefficient of the term $X^iY^jZ^k$ in the expansion of the integrand. Hence:

$$
m_{pqr} = \|T\| \int_W \left(\sum_{i,j,k} c(i,j,k) X^i, Y^j, Z^k \right) dXdYdZ \qquad (4.19)
$$

So, to evaluate the integral of polynomial $x^py^qz^r$ over the orthogonal unit tetrahedron W, the following formula is used:

$$
\begin{aligned}
\int_W x^p y^q z^r \, dv &= \int_0^1 \int_0^{1-z} \int_0^{1-z-y} x^p y^q z^r \, dxdydz \\
&= \frac{p!q!r!}{(p+q+r+3)!} \qquad (4.20)
\end{aligned}
$$

The derivation of this is straightforward and is desribed in [LK84].

This provides the ability to calculate the three quantities desired, volume, centre of mass and inertia tensor. Volume is the simplest and

will be covered here, the other two quantities will be described in section 4.5 where they are used.

To calculate the volume contributed by the tetrahedron of a single face then becomes very simple:

$$\begin{aligned} m_{000} &= \int_Q x^0 y^0 z^0 dx dy dz \\ &= \|T\| \int_W dX dY dZ \\ &= \|T\| \frac{0!}{3!} \\ &= \frac{\|T\|}{6} \end{aligned} \tag{4.21}$$

To calculate the volume in an entire body simply requires summing the contributions from each face of the polyhedron where the components of v_1, v_2 and v_3 are the coordinates of the face in clockwise order.

As the muscle shortens during contraction, to preserve its volume the muscle must expand in cross-section as it shrinks in length. The process used to achieve this is detailed next.

4.4.3 Keeping Muscle Volume Constant

It is common in muscle animation to try and keep muscle volume constant during contraction. Most researchers simply scale the muscle or its cross section in several places to try and keep the volume constant. This work

takes an approach which succeeds in keeping the muscle volume constant during contraction to within a user specified tolerance limit, typically in the region of 99%.

To keep the volume constant, first the volume at zero contraction is found and then the volume at sixty percent contraction is found. Sixty percent is chosen as the maximum contraction that is commonly useful in a muscle system. Real muscles do not actually contract as much as this, but in the realm of animation, over-emphasising an action is sometimes desirable and thus being able to produce realistic results in unrealistic situations is useful.

The width and height expansion of muscles during contraction is controlled by the expansion control points. To keep the muscle's volume constant these must expand enough to counteract the loss of volume from change in length. Likewise if the muscle is stretched then the expansion CPs must shrink to counteract the increase in volume from the elongated muscle. The amount to change the length of the expansion CPs by is first estimated by assuming the muscle is roughly cylindrical and producing a scale from this. The change in cross sectional area of a cylinder due to a change in length is defined as

$$\sqrt{\frac{contractedlength}{restlength}} \tag{4.22}$$

If the muscles were perfect cylinders then scaling the cross section by this amount would keep the volume constant. However the muscles used here can be any shape and as such the above estimate is never likely to be correct. For this reason it is used as a starting point for the iterative discovery of the true value by which to alter the expansion CPs. This value is first used to scale the expansion CPs and then the volume of the muscle is recalculated. Using the calculated volume, a binary search is used to get the volume to within the specified tolerance of the starting volume.

For each step of the search the current volume is compared to the desired volume. If the volume is too great then an expansion factor half way between the current and the current highest volume is selected. If it is too small then a volume half way between the current volume and the current lowest volume is selected. The current highest or lowest is then set to be the value of the previous current volume as is standard in a binary search.

The number of steps taken to reach a tolerance of 1% is usually only around five and is thus very quick to evaluate.

During muscle contraction, the expansion factor is interpolated from zero to the maximum contraction found by this search. Depending on the muscle shape, this can cause the volume to increase too rapidly or too slowly meaning that the muscle volume does not remain constant during the full contraction. To counteract this, in the muscle setup stage, keyframes are generated along the length of the contraction to keep the value constant. To keep the muscle volume within a tolerance of the original for the entire contraction, keyframes are generated at the positions during contraction which have maximum difference from the true volume. These positions are found by stepping along the contraction and finding the point of maximum difference in volume and then repeating the process of finding the correct expansion CP value for this contraction value. This process is repeated as many times as required to ensure the volume remains within a specified tolerance throughout the entire contraction. Once these keyframes are generated the expansion CP values during contraction are interpolated using a Catmull-Rom Spline between control points to keep the expansion smooth.

The expansion points all expand uniformly, there is not more scaling towards the centre than at muscle ends. This is by design, as the muscle model is designed to handle muscles of any shape, so there is no way to know that the centre section should expand more as if the muscle were

a simple fusiform shape. The expansion CPs are positioned near the surface of the mesh, so if the muscle naturally bulges in the centre, these CPs will already be further from the muscle fibre than others and so the expansion will bulge mode in the centre.

The muscles of facial expression are the muscles commonly simulated in facial animation models. These muscles do not however manipulate the jaw. The jaw or mandible is moved by the muscles of mastication, these muscles are commonly overlooked in facial animation simulations but they are essential in representing realistic motion of the face.

4.5 Muscles of Mastication

The muscles of mastication manipulate the mandible. Unlike the muscles of facial expression these muscles are attached to bone at both ends and are usually much larger due to the strength required for chewing and biting. Opening of the mandible can occur in more than one way. it can be a purely downward rotation at the temporomandibular joint. It can also include movement of the mandible anteriorly at the temporomandibular joint. It can sometimes even include lateral deviation of the mandible (side to side movement). This range of movements is caused by the contraction of four different muscles which connect the mandible

to the cranium.

The muscles of mastication all connect onto the mandible with the *temporalis* and *masseter* being superficial and the two *pterygoid* muscles lying deeper. Other muscles of the face can move the mandible slightly but these four muscles are the primary muscles of mastication.

Most previous works have modelled the jaw's movements as a simple rotation of a mesh, sometimes incorporating side to side movements. This work recreates the range of movements described earlier by simulating the forces applied to the mandible by the muscles of mastication to produce movements that are physically based. This allows motions such as chewing to be easily and realistically modelled by contracting the muscles as a real person would do.

The four muscles of mastication each perform a different task. Figure 4.13 shows the models used to simulate these muscles and the action performed by each muscle is defined next.

Temporalis

Elevation - During contraction, the temporalis pulls the mandible superiorly toward the cranium, elevating the mandible.

Retraction - When the temporalis contracts, it pulls the mandible posteriorly toward the cranium, retracting the mandible.

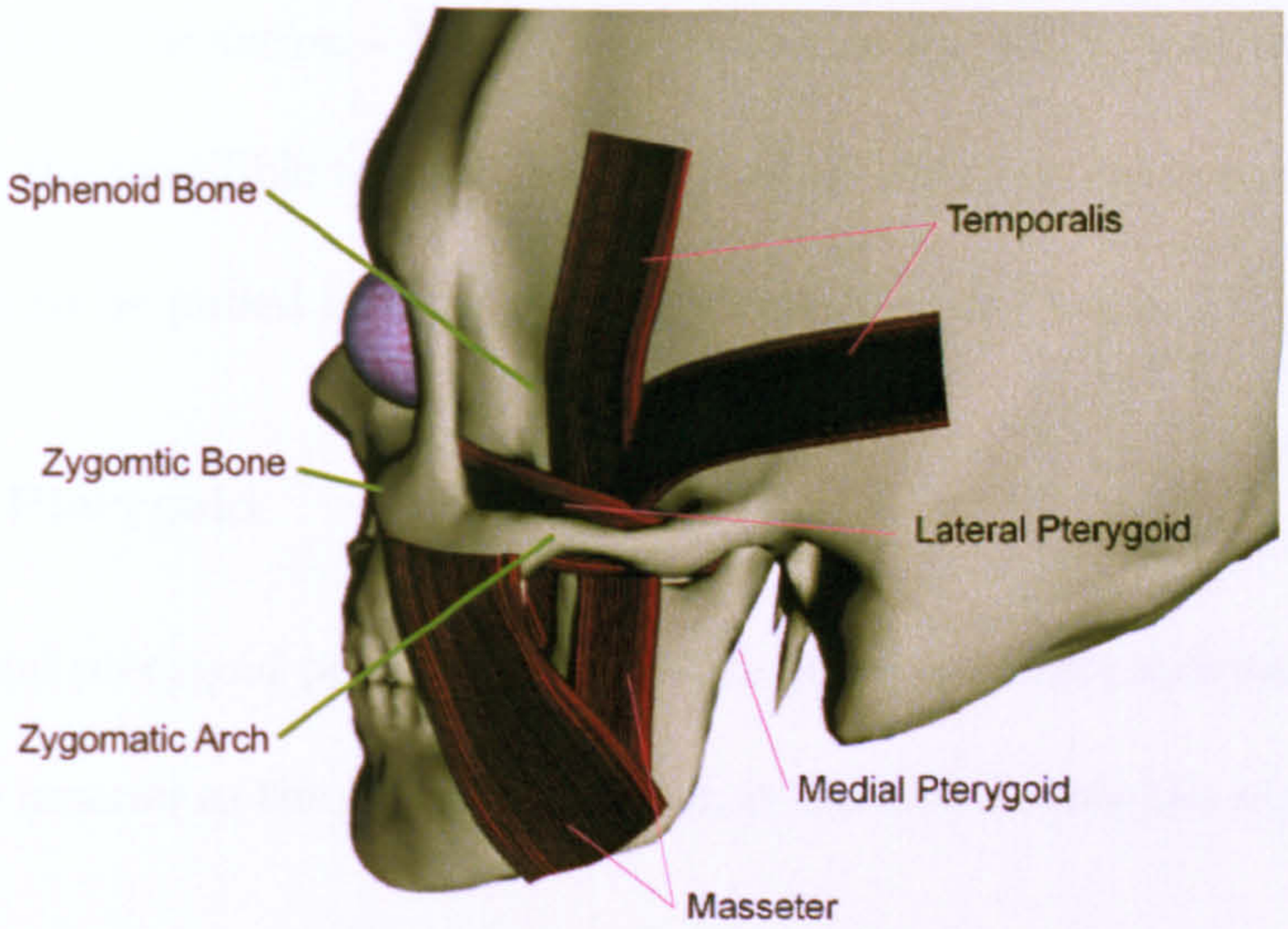


Fig. 4.13: Close up view of the muscles of mastication

Masseter

Elevation - When the masseter contracts it pulls the mandible superiorly toward the zygomatic bone and arch, elevating the mandible.

Protraction -When the superficial layer of the mandible contracts, it pulls the mandible anteriorly toward the zygomatic bone, protracting the mandible.

Lateral Pterygoid

Protraction - When the lateral pterygoid contracts, it pulls the mandible towards the sphenoid protracting it.

Contralateral Deviation - If only one lateral pterygoid is contracted, it pulls the the mandible toward the sphenoid on only one side causing the mandible to be pulled toward the opposite side of the body.

Medial Pterygoid

The medial pterygoid performs protraction and contralateral deviation in the same manner as the lateral oterygoid, it can also elevate the mandible however.

Elevation - When the medial pterygoid contracts it pulls the mandible superiorly toward the sphenoid, causing the mandible to be elevated.

These four muscles manipulate the mandible into producing a wide range of movements. When they are contracted, they pull on a rigid mandible which is modelled as a rigid body structure.

4.5.1 Rigid Body Dynamics

The mandible is made from bone and as such is a rigid structure which doesn't deform like skin and muscle under the small forces of muscle contraction or collision with other head parts. To model this non-deforming object a rigid body approximation of the mandible was created.

A rigid body is a mathematical abstraction of a solid physical object.

Rigid bodies are non deformable and their movements and interactions with the world are simulated using the laws of Newtonian mechanics. Rigid body dynamics will be briefly covered here to explain the methods used to simulate the mandible-muscle-cranium interactions. For a more complete coverage of rigid body dynamics the reader should consult [WB01].

Before explaining the motion and collision response of the mandible as a rigid body, some definitions will be presented to aid the reader in understanding the terminology used.

Definitions

Centre of mass - The centre of mass is calculated using the techniques detailed in section 4.4.2. The moment of the centre of mass is:

$$\begin{aligned} m_{100} &= \int_Q x^1 y^0 z^0 dx dy dz \\ &= \|T\| \int_W x_1 X + x_2 Y + x_3 Z) dX dY dZ \\ &= \|T\| \left(x_1 \frac{1!}{4!} + x_2 \frac{1!}{4!} + x_3 \frac{1!}{4!} \right) \\ &= \|T\| \frac{x_1 + x_2 + x_3}{24} \end{aligned} \tag{4.23}$$

This simplifies down to:

$$\frac{1}{4}((x_1 + x_2 + x_3)(y_1 + y_2 + y_3)(z_1 + z_2 + z_3)) \tag{4.24}$$

Linear Velocity - Linear velocity is the instantaneous rate of translation at time t of the centre of mass of the rigid body. Thus it is simply the first derivative of the position with respect to time,

$$v(t) = \dot{x}(t) \tag{4.25}$$

Angular Velocity - If the rigid body's centre of mass is frozen in position, any movement of the body is a rotation around an axis that passes through the centre of mass. This spin is described by the vector $\omega(t)$ The direction of $\omega(t)$ is the axis around which the body is spinning and the magnitude of $\omega(t)$ is the speed of rotation. The first derivative of position with respect to time is velocity, but as the orientation $R(t)$ of the body is a matrix and the axis the body is spinning round, ω is a vector then the derivative of $R(t)$ does not lead directly to ω. To find the derivative of $R(t)$, each column of the matrix must be treated separately as the cross product of each with ω.

$$\dot{R} = \omega(t) * R(t) \; defined \; as \; \left(\omega \times \begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} \omega \times \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix} \omega \times \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \right) \tag{4.26}$$

Torque - The torque τ on a body with centre of mass x at a point p, due to the total force F from external forces is $\tau = r \times F$ where $r = p - x$.

Linear and Angular Momentum - The linear momentum p of the rigid body with linear velocity v is $P = mv$. The rate of change of linear momentum is the total force acting on a body $\dot{P}(t) = F(t)$.

The angular momentum of a body $L(t)$ is defined in a similar manner to the linear momentum and is $L(t) = I(t)\omega(t)$ where $I(t)$ is a 3 x 3 matrix called the *inertia tensor* which is described below. The relationship between angular momentum and torque is analogous to the relation between linear momentum and force and is defined as

$$\dot{L}(t) = F(t) \tag{4.27}$$

The Inertia Tensor - The *inertia tensor* of a rigid body defines the relationship between angular momentum $L(t)$ and angular velocity $\tau(t)$. At time t, the inertia tensor $I(t)$ for a point p' (defined as a displacement from the centre of mass to the point p, $(p' = p(t) - x(t)))$ and with a mass m, is:

$$I(t) = \int_{p' \in V} \begin{pmatrix} m(p_y'^2 + p_z'^2) & -mp_x'p_y' & -mp_x'p_z' \\ -mp_y'p_x' & m(p_x'^2 + p_z'^2) & -mp_y'p_z' \\ -mp_z'p_x' & -mp_z'p_y' & m(p_x'^2 + p_y'^2) \end{pmatrix} \quad (4.28)$$

The inertia tensor varies with the body's rotation however the inertia tensor can be computed in body space and transformed into world space along with the body using the orientation matrix $R(t)$.

$$I(t) = R(t) I_{body} R(t)^T \quad (4.29)$$

This allows the inertia tensor to be pre-computed for the body and simply rotated at run time to align it with the body. To calculate the inertia tensor of a body, the same techniques as were used to find the volume and centre of mass are used. The inertia tensor can be expressed as a matrix constructed from the second order moments of the body:

$$I(t) = \begin{pmatrix} (m_{020} + m_{002}) & -m_{110} & -m_{101} \\ -m_{110} & (m_{200} + m_{002}) & -m011 \\ -m_{101} & -m_{011} & (m_{200} + m_{020}) \end{pmatrix} \quad (4.30)$$

The calculation of the moments used to fill this matrix are performed in the same manner as calculating the centre of mass and volume of the object.

4.5.2 Moving the Mandible Using Muscle Contractions

The muscles of mastication are modelled and contracted in the same way as the muscles of facial expression. The muscles of mastication must however apply forces to the rigid body mandible rather than moving the skin.

The muscles of mastication do not actually connect directly to the mandible. Instead, they are connected by springs which have a rest length of zero. When a mastication muscle is contracted, it will pull on the spring connecting it to the mandible and thus apply a force to the rigid body mandible. As the jaw is rigid and cannot deform, it will be moved by the force of the contracting muscle.

As the muscle contracts, it causes the spring to extend. The spring forces are modelled using:

$$f_1 = -k_s(x_1 - x_2) - k_d(v_1 - v_2), \quad f_2 = k_s(x_1 - x_2) + k_d(v_1 - v_2) \quad (4.31)$$

where k_s is a spring stiffness constant, k_d is a general damping constant and x_1 and x_2 are the positions of the ends of the spring. This equation causes the two particles to be forced toward each other whenever they are not coincident. When this force is applied to the jaw, it

causes it to move. How this movement is calculated is explained next.

Movement of a Rigid Body

To locate a rigid body in space at time t a vector $x(t)$ will be used which describes the translation of the body. Rotation of the body can be described in terms of a 3×3 rotation matrix $R(t)$. The components of this matrix are orthonormal vectors detailing the x, y and z axes describing the coordinate frame of the rigid body. The rigid body is defined in terms of a local space called *body space*. From a geometric description of the body in body space, $x(t)$ and $R(t)$ are used to transform this body into world space. The body is defined with its centre of mass at the origin of body space.

Moving the mandible is accomplished by first summing the forces applied to the body. The only two forces acting on the body are due to gravity and the pull of muscles. As gravity applies equally to all points on the mandible, it can be thought of as acting through the centre of mass and does not induce any rotation in the body.

The total force acting on the mandible at time t is calculated by summing the force from each muscle and that produced by gravity. This is shown in figure 4.32.

$$F(t) = \sum F_i(t) + g(t) \tag{4.32}$$

Any force not acting through the centre of mass will induce a turning force on the mandible, to calculate this total torque equation 4.33 is used.

$$\tau(t) = \sum \tau_i(t) = \sum (r_i(t) - x(t)) \times F_i(t) \tag{4.33}$$

r_i is the position where a the force acts, in this situation, it is the point where the spring connects from the mandible to the muscle and $x(t)$ is the position of the centre of mass of the mandible.

The linear momentum of a particle was defined earlier as the sum of the mass and velocity of that particle. The linear momentum of a rigid body can actually be defined as if it is simply a particle with mass M and velocity $v(t)$:

$$P(t) = \sum m_i v(t) = \left(\sum m_i\right) v(t) = Mv(t) \tag{4.34}$$

The concept of linear momentum allows the expression of the total force $F(t)$ on a rigid body as:

$$\dot{P}(t) = F(t) \tag{4.35}$$

This relation is straightforward to derive and can be found in [WB01],

it is analogous to the relationship between angular momentum and torque described in equation 4.27.

To calculate the movements of the mandible requires calculating the velocity, angular velocity and current inertia-tensor. All the values defining the state of a rigid body can be represented in a state vector $\mathbf{X}(t)$:

$$\mathbf{X}(t) = \begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix} \tag{4.36}$$

The mass M of the body and body space inertia tensor I_{body} are constants, at time t the unknown quantities $I(t)$, $\omega(t)$ and $v(t)$ can be computed by:

$$v(t) = \frac{P(t)}{M}, \quad I(t) = R(t)I_{body}R(t)^T \quad and \quad \omega(t) = I(t)^{-1}L(t) \tag{4.37}$$

The derivative of $X(t)$ is:

$$\frac{d}{dt}\mathbf{X}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t) * R(t) \\ F(t) \\ \tau(t) \end{pmatrix} \tag{4.38}$$

Movement

The position of the mandible at a certain time point in the future $t + \Delta t$ is calculated using Euler's method of numerical integration. As the derivative of $\mathbf{X}(t)$ can be calculated from the above equation and the current position of the mandible at time t is known. Then the position at time $t + \Delta t$ is:

$$\mathbf{X}(t + \delta t) = \mathbf{X}(t) + \Delta t \frac{d}{dt}\mathbf{X}(t) \tag{4.39}$$

This equation allows the mandible's position to be numerically integrated forward through time. At each step the forces are all taken into account and the state vector $\mathbf{X}$ is updated then the mandible is moved forward by time step t.

These equations have detailed how the mandible is moved by muscle forces, however this is only half of the mandible simulation method. The mathematics detailed so far are actually for the movement of a rigid body where the centre of mass is the point around which the body rotates. In this jaw model this is not the case, the rotation of the jaw is around the temperomandibular joint, as it is in a real skull. The reason that the mathematical jaw model works with a rotation point not where it should be, is down to the novel method of simulating the temporomandibular

joint. It is obviously possible to simulate a rigid body model where the centre of mass is not the centre of rotation, however the mathematics involved are far more complex than those outlined here. This mathematical complexity translates directly into a more complicated numerical simulation and thus a slower system. Simulating rigid bodies in anything close to real time requires a lot of processing power and to use an even more complicated rigid body model would have slowed the whole system down too much to be useable. The model of the temporomandibular joint developed here, simply and easily copes with correcting the movement of the jaw as it rotates around an offset point. The details of this joint will be covered next, along with how collisions between the jaw and the cranium are handled.

Collisions with the Cranium

To find collisions between the mandible and cranium, simplified models of each are used. Figure 4.14 shows the models used for this stage.

When the mandible is moved, it will come into regular contact with the cranium. There are three main points of contact, the two joints connecting the mandible with the cranium and the teeth. Collision with the cranium is nearly constant in that the mandible is connected to the cranium via the temporomandibular joint. This joint allows the mandible

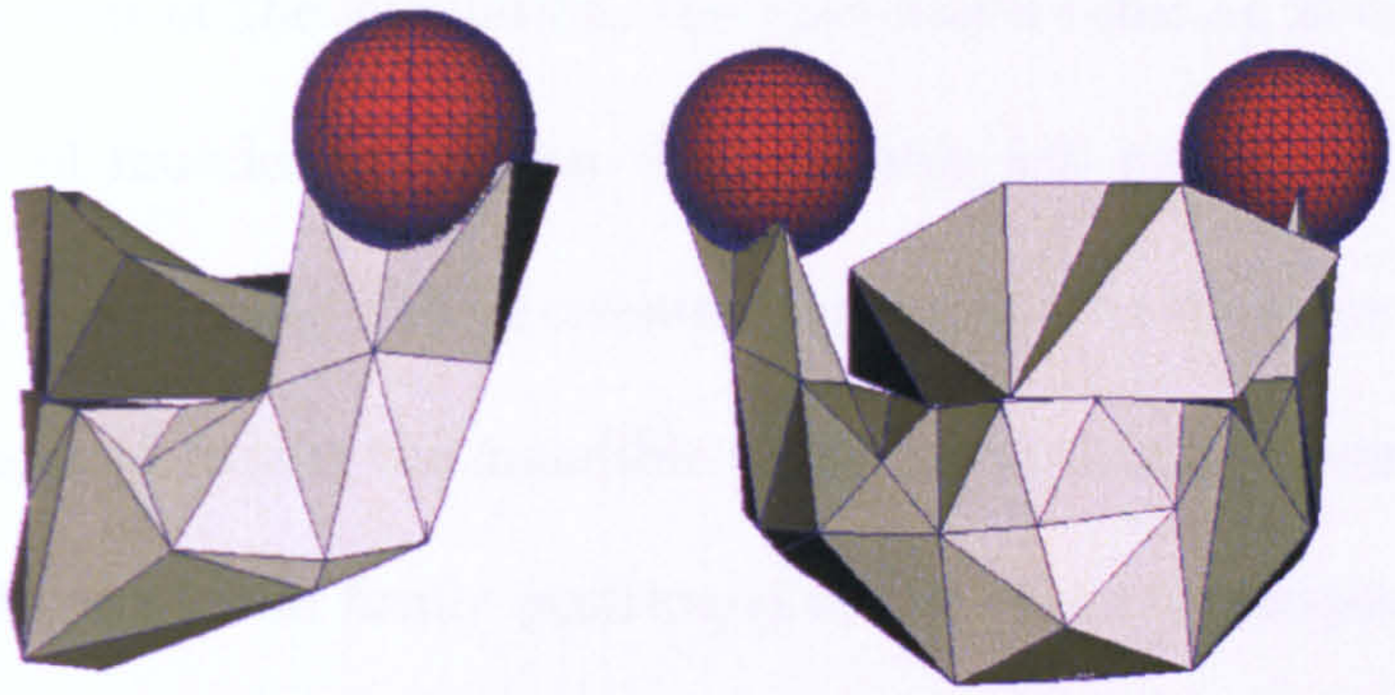


Fig. 4.14: The simplified models used for collision. The red spheres show the spring length for the temperomandibular joint

to move in a large range of directions but constrains its movements in certain ways. To simulate this connection, collision spheres are used. These spheres constrain the mandible to stay roughly the same distance from the cranium at the joints but allow it free range of motion in all other directions. The actual connections between the cranium and mandible are modelled with four springs, two on either side of the head at the points where the mandible usually slots around the temporomandibular joint. These springs hold the mandible at a certain distance from the cranium. Their stiffness is higher than the springs which connect muscles to the mandible and this helps stop muscles pulling the mandible rapidly out of position. These springs are the key to correcting the mandible's rotation around its centre of mass which is not the tempormandibular joint. For

each time step of the simulation, the mandible's rotation is calculated. The force of muscles pulling on the mandible will rotate it around its centre of mass causing it to be oriented incorrectly. At the same time, the stiff springs connecting the mandible to the cranium apply a strong force to keep the mandible firmly positioned in the temperomandibular joint area. These springs act as a counter-balance to the incorrect rotation point and continually keep the mandible within the correct joint area and thus cause the mandible to rotate correctly.

Collisions with the spheres is fast to test and is marked as true if one of the springs connecting the mandible to cranium is shorter than the radius of the sphere. If it is, then a collision is signalled at the point of attachment of the spring and it is handled in the same manner as a collision with the teeth which is detailed next.

Collision with the teeth are handled using a simplified model of the collision area. The upper teeth are roughly flat across the surface and calculating collisions between every vertex in the mandible and cranium would be a time consuming process and as the tooth surface is so flat, collision would occur at many points simultaneously. Simultaneous collisions slow down rigid body simulation greatly as finding the exact moment of contact involves rewinding the simulation back to the point where movement was impeded. There are solutions which do not require revers-

ing the simulation, such as [GBF03] but these solutions are excessive in this situation where the collision object is simple and a straightforward method of collision response will provide faster simulation time. To speed up calculations for the purposes of collision the tooth area is replaced by a low resolution proxy.

The upper teeth proxy model is considered to be immovable. In a real skull a strong movement of the mandible could move the cranium but in the actions of chewing or speaking this is not a likely scenario and so is ignored. Setting the top teeth to be immovable is accomplished by setting $\frac{1}{mass}$ to be zero and the inverse inertia tensor to be a zero matrix. In the dynamics calculations only the inverse mass and inverse inertia tensor are ever used there is not a problem with divide by zero.

As the movement of the mandible is simulated, a check for collisions is done at each time step. Collision detection is performed using the fast triangle-triangle intersection test of Tomas Möller [M̈97].

When a collision is detected, the numerical integration code backs up the simulator half way between the last time step and the current to check for collision. If there is still a collision, it backs up half way again. This process is repeated until the two bodies are within a certain tolerance of each other but not actually colliding.

When a collision has been found at a point p, the velocity of that

point on the mandible is calculated using the methods from earlier:

$$\dot{p} = v(t) + \omega(t) \times (p(t) - x(t)) \tag{4.40}$$

This collision will change the velocity of the mandible, however the velocity cannot be changed directly, and any force that is applied no matter how large, will take some time to cause a change in direction of the mandible. Instantaneous change in velocity is accomplished using impulse. Impulse can be thought of as a large force that acts over a very short period of time. Applying an impulse J to the mandible with mass M produces a change in linear velocity:

$$\Delta v = \frac{J}{M} \tag{4.41}$$

The change in linear momentum ΔP is simply: $\Delta P = J$

The torque produced by the impulse is:

$$\tau_{impulse} = (p - x(t)) \times J \tag{4.42}$$

The impulsive torque produces a change in angular momentum ΔL and the change in angular velocity is:

$$I^{-1}(t)\tau_{impulse} \tag{4.43}$$

Friction between the two bodies is not taken into account and so the direction of the impulse is the normal direction $n(t)$ of the surface at the collision:

$$J = jn(t) \tag{4.44}$$

After the application of impulse, the movement of the mandible will have a relative velocity of:

$$v_{rel} = -\epsilon(n(t) \cdot (\dot{p}(t)) \tag{4.45}$$

The velocity of the teeth object is not needed in the equations as it is always zero. The value ϵ is the *coefficient of restitution* and is in the region of zero to one. One producing a perfect bounce collision and zero producing no bounce at all.

The actual value of the impulse from the collision is straightforward to calculate but deriving the equation is extended. For this reason only the final equation is presented here, the derivation can be found in [WB01]

$$j = \frac{-(1+\epsilon)v_{rel}^{-}}{\frac{1}{M} + n(t) \cdot (I^{-1}(t)(r \times n(t))) \times r} \tag{4.46}$$

Resolving Contact Forces

The problem of contact is an extensive subject in rigid body literature. There are many methods in use, the one used here has the advantage of being simple and fast [Bou02]. This is essential in the developed system because there are a large number of simulation calculations being processed each frame due to the muscle, bone and skin.

A contact situation between the mandible and cranium is found in the same manner as the collisions above, however the velocity of the mandible must be zero and the acceleration of the mandible must be towards the cranium for the two to be in contact.

As the mandible is the only moving object in the simulation and it can only contact with the cranium then contact is dealt with in a straight-forward manner. A contact force is first found for the point of contact. This force is equal to the mass of the contacting particle multiplied by the acceleration of that particle. This can be thought of as assuming the mandible is made up of a collection of particles with equal mass. This force is then applied to the mandible at this point to cancel out the force which would cause the mandible to move into the cranium. This neatly allows the mandible to sit in contact with the cranium.

4.6 Fitting To Any Model

The muscle system described so far is based on creating muscles to fit a specific skull. This is not very useful for general purpose facial animation as the skull shape and thus muscle positions and size vary from person to person.

Fig. 4.15: The muscles and skull of one model positioned inside another. The green depth pegs are visible in some places where the mesh is too small for the skull and hidden in others where it is too large.

Figure 4.15 shows an example of the muscles not fitting into another head. To overcome this problem, a method of easily warping the skull and muscles of one person to fit another skin surface mesh was developed. To manually manipulate the vertices of the muscles and skull to fit a new

head mesh would be nearly impossible even with advanced modeling software. Collectively the muscles and skull have around 11,000 vertices. This number could be reduced significantly by using simpler skull and muscle meshes, but to keep enough detail in the models the number of vertices has to be in the thousands. Manually manipulating these vertices and faces to make them fit another skin mesh is a highly skilled artistic process. A user would most likely be quicker rebuilding all of the muscles from scratch than trying to alter the pre-built meshes to fit a new skin surface.

This problem is solved by finding a mapping between a simple version of the skull and muscles consisting of tens of points rather than thousands and the new surface skin mesh to be used. This mapping between a simple version of the skull and muscles will then be used to automatically move the majority of the skull and muscle vertices and control points. Moving only a small set of points manually and having the majority line up automatically requires a deformation method which can move large areas of a model using only a few parameters.

In 1998, Singh and Kokkevis introduced Surface-Oriented Free-Form Deformations (SOFFDs) [SK00]. SOFFDs are similar to FFDs in that the object to be deformed is manipulated by another proxy object, but they allow a more precise control over the deformations. In a SOFFD,

each vertex in the object to be deformed is transformed by the warping of the faces of a simplified proxy object. This close control of a detailed object by a simple proxy is exactly what is needed to manipulate a high resolution skull and muscles into fitting a new model.

4.6.1 Setting up a SOFFD

In a SOFFD, the object to be deformed S (in this case the skull and muscles) is bound to a deformer object O. When O is manipulated, it causes similar changes to occur in S. As O is usually a simplified version of S, it means that specific changes in O cause broader changes to the surface of S.

O can actually be considered as two surfaces, R and D, the *reference surface* and the *driver surface*. Initially, D is an exact copy of R. Surface D is the object that will be manipulated and as it is deformed its surface deviates from R and this difference between the two is what is used to calculate the deformation applied to S.

Before deformation can take place, a correspondence between S and the deformer object O must be produced. This correspondence defines how the movement of each triangular face in D will affect the movement of S. The vertices of the surface being deformed S make up the set P^S,

for every point in this set, an *influence weight* is produced for every face of the reference surface R. This influence weight determines how much a deformation of each element in D will affect each vertex in S. The influence weights are calculated based on a distance metric. Faces of R closer to a point P of S will have a higher influence than faces of R further away. To compute the influence weights the distance metric represented as a scalar function $f(d, local)$ is employed. The first parameter, d is the distance of P from the face of R. The second parameter, *local*, controls the rate at which the function f decays in value with an increase in distance d. The function $f(d, local)$ for any d, $local \geq 0$ is:

$$f(d, local) = \frac{1}{1 + d^{local}} \tag{4.47}$$

The distance d is defined from point P to the face being examined to be the length of the vector $\overrightarrow{PP'}$ where P' is the point on the surface of the face that is closest to point P. For each point P and every face k of the deformer object's reference surface R the influence weight is defined as $w_k^P = f(d_k^P, local)$, where d_k^P is the distance from point P to face k. The influence weights for point P are normalised to preserve the convex hull property of the surface S. After normalisation most weights become zero making the deformation calculation efficient as only faces with a non

zero weight need be used in the deformation calculation.

The influence weights define the amount to deform the vertices of S based on the movements of the faces of D but it says nothing of how to deform them. To find how to deform each point, a correspondence between each point of S and each face of R must be produced. This is done by defining a coordinate system based on the edges of each triangular face of R. Defining the vertices of an arbitrary face as A, B, and C, the coordinate system for the face is defined with its origin at A and axes as:

$$\vec{e_1} = \vec{B} - \vec{A}, \quad \vec{e_2} = \vec{C} - \vec{A}, \quad \vec{e_3} = \frac{\vec{e_1} \times \vec{e_2}}{\| \vec{e_1} \times \vec{e_2} \|} \tag{4.48}$$

This coordinate system can be defined in a 4×4 transformation matrix Q. To define a point P in the local coordinate system of a face, the point is multiplied by the inverse of the transformation matrix for that face. The point P defined in the local coordinate system of face k of the reference surface is $P_k^R = P(Q_k^R)^{-1}$.

4.6.2 Deforming with a SOFFD

The setup stage allows the deformation of all points in surface S to be manipulated by deforming the faces of surface D and moving each point to preserve its local position in each face. As the deformer surface D

is manipulated, the faces of D change shape, position and orientation. Comparing the coordinate system defined by the face k on the reference surface and the same face k on the driver surface makes it possible to calculate the deformation effect of each face k on a point P. To calculate the world space position of point P after deformation by face k, the point P_k^R is transformed back into world space using the transformation matrix for face k of the driver surface Q_k^D. So the world space position of point P after deformation by element k is $P_k^{def} = P_k^R Q_k^D$. To calculate the final position of point P after deformation by all faces of the driver surface, the position of P after deformation by each face is weighted by the influence value for that face which is stored in vector U^P and then all the positions are summed. This gives the final position of point P after deformation to be:

$$P_{def} = \sum_{k=1}^{n} u_k^P P_k^{def} \tag{4.49}$$

4.6.3 SOFFDs Applied to The Muscles and Skull

Surface Oriented FFDs provide a good basis for manipulating objects using a reduced set of controls. To get good results in terms of the muscle, skull and skin there are two questions that must be answered:

- What should be used for the deformer object?

- How to align the muscles and skull in their correct positions in an arbitrary head?

It turns out that solving the second problem leads on to solving the first so that is the order they will be approached here. To know which bit of skull and muscle should go where and at what depth from a new head surface is a problem that has been under examination by forensic reconstruction scientists for many years. Forensic scientists are usually trying to ascertain what a person would have looked like given only their skull. The problem here is the opposite as the skin surface is known and how the skull and muscles should look underneath is the unknown. The general idea is however the same, that of determining the relationship between a skull and a skin surface.

Through numerous studies of specific ethnic groups, tables of data showing the average skin depth at specific skull points are available. The data used here, is based on white adult males [IH93]. Forensic reconstruction artists usually use such data as a guide for building up layers on top of a skull to see how the skin surface covering the skull would have looked. In this system, this data is used to define where specific points on the skull and muscle model should lie in space under

the new skin mesh. By warping the skull and muscles so that these landmarks lie at the correct depths at all the markers then it will produce a good approximation of the actual skull and muscle of the mesh being conformed to.

Fig. 4.16: The skull and muscles with skin depth pegs attached

This answers the question of how the muscles and skull should be aligned on an arbitrary head mesh. It is done by selecting the points on the new mesh where these markers lie and then the skull and muscles need to be warped to fit these points at the correct depth and position. To actually warp the skull and muscles requires a deformer object. The object must be able to deform to fit exactly the small number of physical landmarks and also be close enough to the skull and muscles to be able to accurately move them.

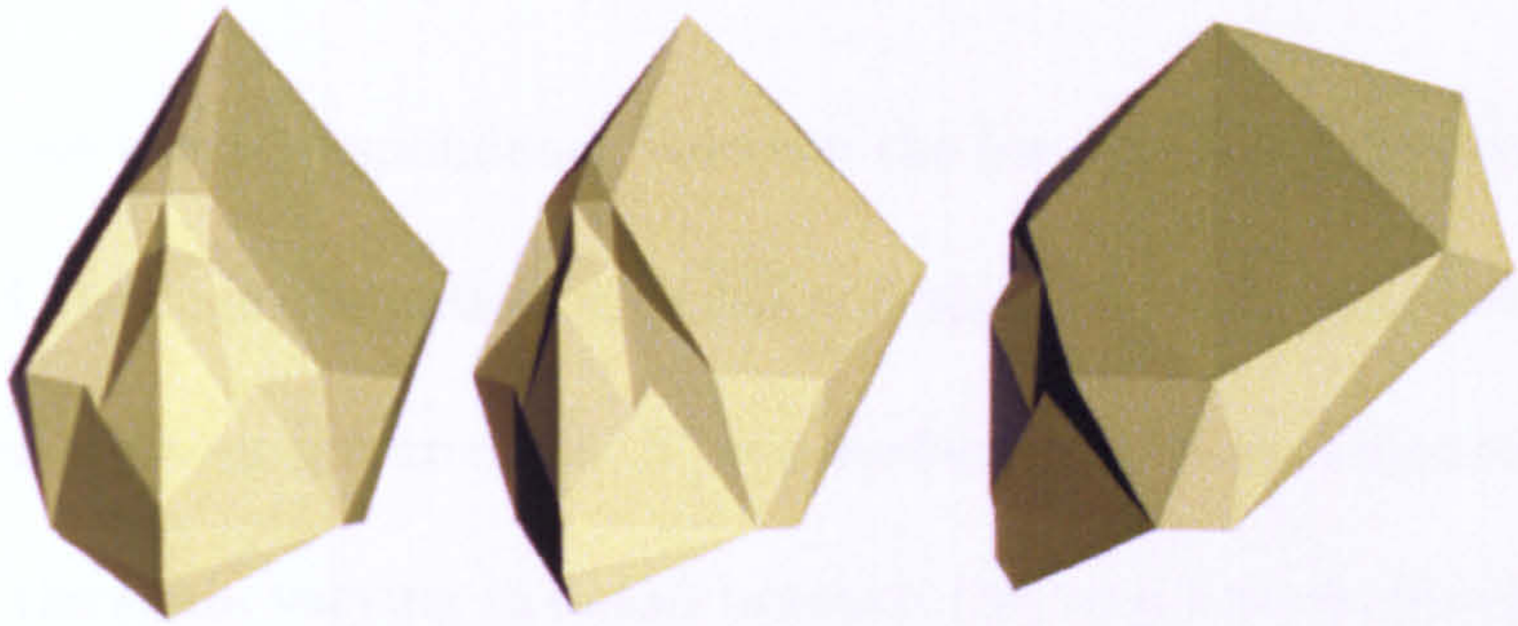


Fig. 4.17: Proxy mesh produced from triangulating the tips of the depth pegs

To produce such a proxy deformer object, a very low resolution mesh was constructed by triangulating the surface created by the tips of the depth pegs when placed on the created skull and muscles. This mesh is shown in figure 4.17.

This mesh is far removed from the high resolution muscle and skin models and so using this low resolution mesh as a deformer object can produce results that are often unrealistic, where the skull and muscles end up warped with an irregular surface. This happens because large parts of the skull and muscle geometry are being deformed by the same triangle on the deformer surface. This causes areas of the skull and muscle to be deformed by faces in the deformer surface which are quite far away on the model and in reality should not be affecting those points.

Deformer Surface Chain

To improve the correspondence between the low resolution deformer surface and the high resolution muscles and skull, a series of intermediate deformer surfaces are created. These surfaces are lower resolution versions of the skull, varying in detail between the very low resolution proxy and the full resolution model. Figure 4.18 shows four models in the chain.

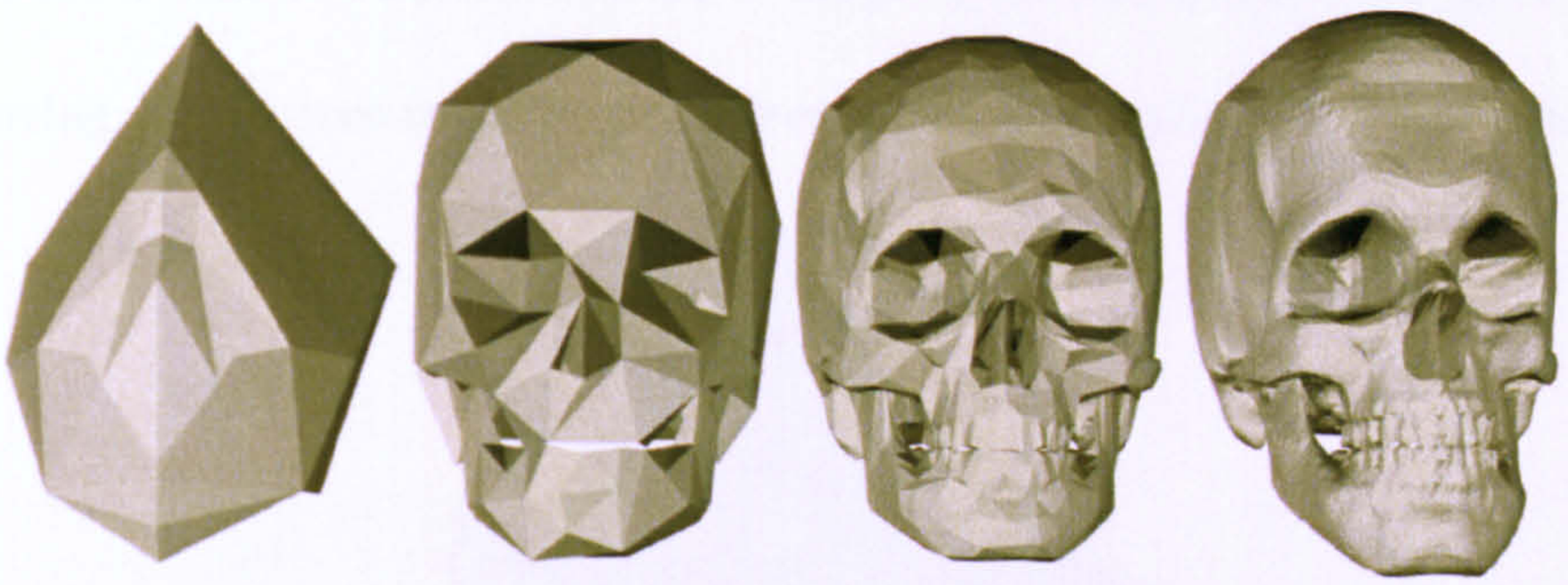


Fig. 4.18: Four skulls used in a wrap deforming chain to create a good correspondence between the landmark points of different head meshes. From left to right, the skulls have 38, 150, 2600 and 10189 vertices.

A chain of SOFFDs is then constructed with each model being the deformer surface for the next higher resolution model in the chain. The deformer object at the top of the chain acts as deformer to the actual skull and muscle models.

At the bottom of the chain is the model constructed from the triangulated surface S_0 of the ends of the depth pegs. To move this surface, the user selects the points on the skin mesh where the depth pegs should lie. These points will lie on triangles in the skin mesh and to find the position of object S_0, the inverse normal of this triangle is project down by the depth of the mesh at each point. Once all points have been selected, the low resolution deformer mesh is simply positioned under the mesh by moving its vertices to sit at the selected points. Figure 4.19 shows this.

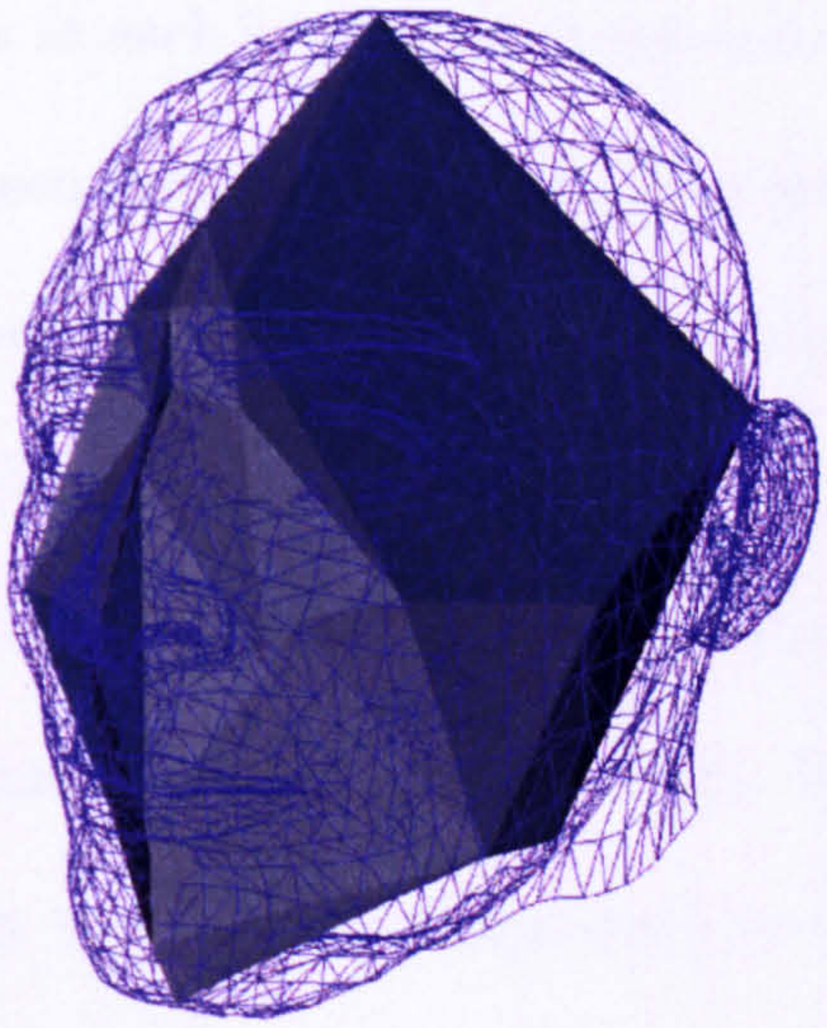


Fig. 4.19: The position of the lowest deformer object S_0 under the new skin mesh.

The position of the next object in the deformer chain S_1 is found automatically from the position of the lowest level object. A SOFFD

is created using S_0 as the deformer object and S_1 as the object to be deformed. This automatically creates a link between the movements of the depth pegs and the movements of the first level in the chain. The second level in the chain S_2 is connected to object S_1 in the same manner. S_1 is attached as the deformer object and S_2 is deformed. This process is repeated until the top of the chain is reached.

This chain of SOFFDs propagates the change in position of the lowest level up to the highest level containing the skull and muscles. The difference between models at each level is small enough that there is a good correspondence between the faces of the deformer and the vertices of the object being deformed. This causes the changes to gradually change the models shape to fit the new skin mesh.

The number of models required in the chain varies depending on how close the skull and muscles fit the new skin mesh. If the two are similar then in practice, only two or three stages need be used in the chain, if they are quite different then five or more may be required. To create the models for the SOFFD chain requires making several simplified versions of the skull model.

Creating the Chain of Models

The high resolution skull model has several thousand vertices and to get a good correspondence between the lowest and first level deformer objects, model S_1 must be really quite low in resolution. The model for level S_1 is created by reducing the number of vertices in the skull model to the region of one hundred and fifty. If the simplification algorithm is careful of preserving features then even such a low resolution mesh still resembles the original skull shape. The simplification algorithm used here [HG99], [GH01] is based on removing edges whilst trying to keep the error at each vertex to a minimum.

The simplification algorithm produces good results when run simply on the mesh, but if specific parts of the model are required to be in the final low resolution version, such as an obvious feature which will help the chain's correspondence between levels then these landmarks can be selected on the skull surface and tagged as non-moveable. These points will not change during the simplification process and are guaranteed to be in the lowest resolution model.

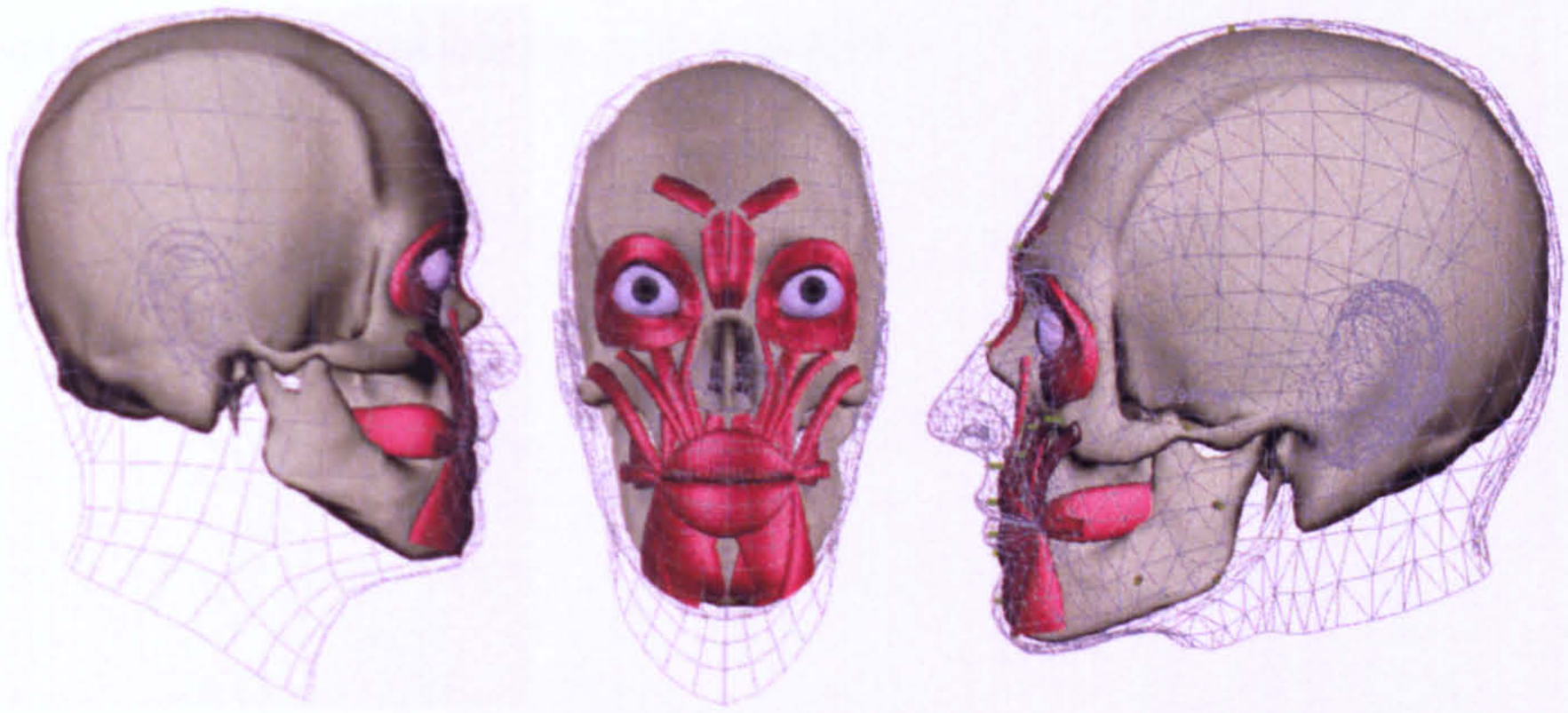


Fig. 4.20: The skull and muscles warped to fit several head meshes

4.7 Conclusions

The muscle model shown here allows muscles with realistic shapes to be deformed closely, mimicking the movement of real muscles, preserving volume and affecting each others movements in a manner similar to the human face. The muscle model can easily be used in any part of the body and as there is no restriction on muscle shape it could be used to model animal or even fantasy creatures for film or television. The muscle and skull can also be easily moved onto any head mesh allowing instant animation. The fitting process usually takes no more than five minutes from start to finish. This is a huge speed improvement over traditional techniques, where even if models could be moved between meshes, the skill involved was very high and the time was in the order of days before

a satisfactory animation rig was produced.

5. Skin

The muscle and bone models detailed in previous chapters have explained how the deep physiology of the face is simulated. When a person speaks however, these structures are not visible and it is the movement of the facial skin that portrays expression and forms the mouth shapes used in speech. As the skin is the main visible feature of any facial animation system, it is of the utmost importance that it behaves realistically or any illusion of realism created from the movements of muscles and bone will be lost.

Skin is the largest organ in the human body, it is waterproof, protects the body from harmful materials, UV light and microorganisms. Skin regulates body temperature, excretes water and the end products of metabolism and is the most extensive sense organ for tactile, thermal and painful stimuli. It is very important in protecting the body, but most importantly for facial animation it is the part of the body that is visible to the viewer.

Real skin slides over muscle and bone, it squashes and stretches as it is pulled into the shapes required for speech and emotive expression. To

produce realistic facial animation, the skin model developed here needed to be able to mimic these complex movements when moved by the muscle and bone subsystem. This chapter will explain in detail how the developed skin model accomplishes this. To help the reader understand the multiple layers of the model, the physical properties of real skin will be covered first which will help to illustrate the mechanical and physiological processes being simulated.

5.1 The Physiological Properties of Skin

Skin is composed of intertwined networks of collagen, nerve fibres, small blood vessels and lymphatics covered by a layer of epithelium [PW96]. At regular intervals on the surface are hairs and sweat glands. The thickness of facial tissue varies across the face, for example around the eyes it is relatively thin whereas around the lips it is thick. The appearance of skin is usually affected mainly by its age although thickness and disease can also affect appearance significantly. Wrinkles are caused by a loss of fatty tissue and skin elasticity as a person ages. Females have more of this fatty tissue than men, however women's skin is in general less thick than men's, giving women a lustre to their skin's appearance.

The structure of skin is arranged into three layers, the epidermis,

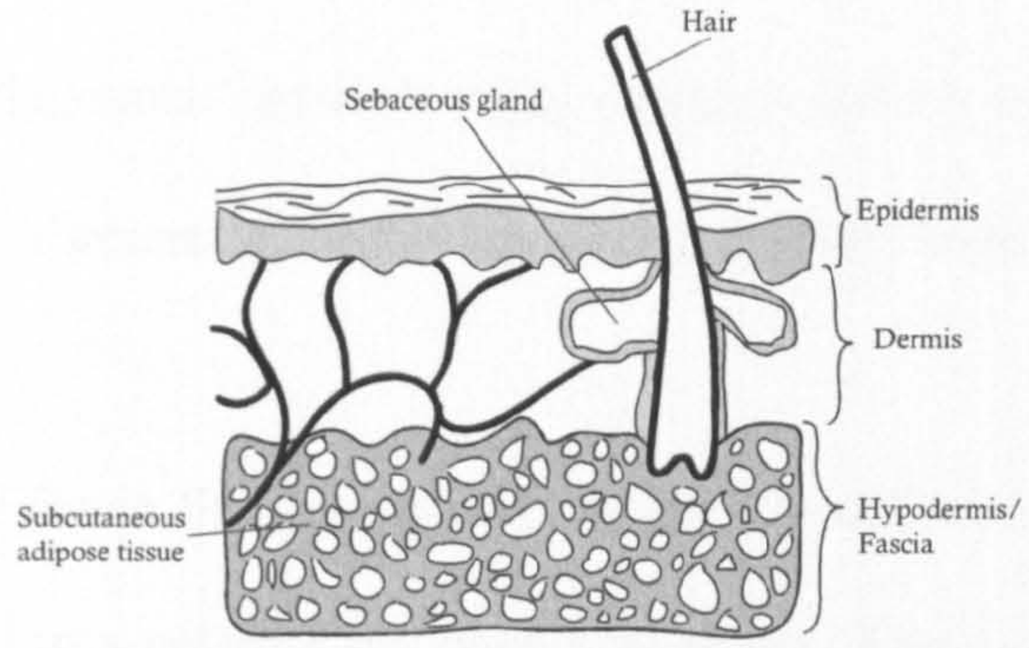


Fig. 5.1: The three layers making up skin [PW96]

dermis and fascia, this can be seen in figure 5.1

Epidermal Tissue

The epidermis is the surface of the skin, a superficial layer of dead cells composed mainly of keratin. The total thickness of the epidermis is about 0.5 - 1.0mm. The cells comprising the epidermis are continually changing, those on the surface are shed as younger cells gradually flatten out and move towards the surface to replace them. The epidermis is the skin's barrier to the outside world and this continual shedding of cells means that the cells in the layer change completely every three to five weeks.

Dermal Tissue

The dermal tissue lies between the superficial epidermis and the deeper

fascia. It is made up of irregular connective tissue and is mainly an interwoven collagenous network with varying content of elastin fibres, proteoglycans, fibronectin, blood vessels, lymphatic vessels and nerves.

Fascia

The superficial fascia lies underneath the dermis and consists of adipose tissue arranged in a network of connective fibres. This tissue is made up mostly of collagen arranged in a lattice of fat cells. Beneath this superficial fascia lies deep fascia which coats the bones and consists mainly of aponeurones which are flat, ribbon-like tendons.

As these descriptions show, skin is a complex structure made up of several substances situated in layers. The way skin reacts to forces, either from the movement of muscles and bone or external pressure is important to the development of a realistic model. The mechanical properties of skin will be covered next, which will explain how skin behaves under pressures and then the developed skin simulation model will be presented.

5.2 The Mechanical Properties of Skin

Skin has a viscoelastic response to stress and strain, meaning that it has properties of an elastic solid and viscous liquid. The elastic property of skin is shown in its storage of energy and how it returns to its rest

shape after removal of a load. The viscous behaviour is shown by the internal forces generated due to deformation, which are due to the rate of deformation *as well as* the amount of deformation.

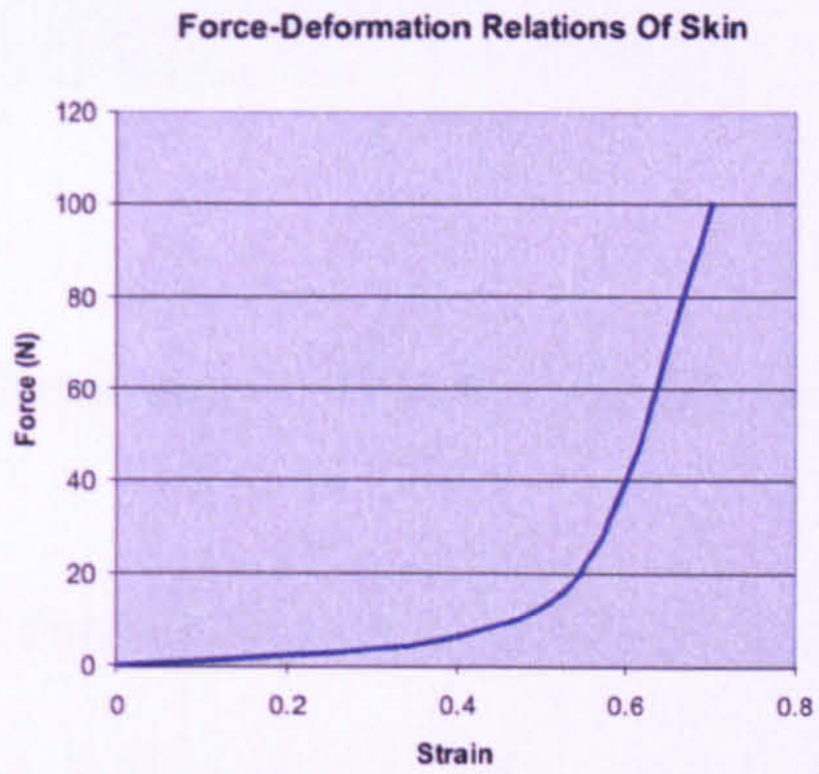


Fig. 5.2: Relationship between load and deformation for skin

The relationship between load and deformation of skin is non-linear, Figure 5.2 shows the relationship between stress and strain in soft tissue under uniaxial tension.

Skin also exhibits hysteresis in its response to loading and unloading, that is, its response to load is different to its response to the removal of that load. This can be seen in Figure 5.3. These properties of skin in reaction to load are due to the interaction of two cellular bases, collagen and elastin. Collagen has a strong response to load and has small range of deformation, it is the material that makes up tendons in the body. Elastin

Fig. 5.3: Response of skin to loading and unloading

is similar to an ideal rubber with an almost linear stress/strain response over a wide range of deformations. The low stress response to strain then rapidly increasing for a small strain increase shown in figure 5.2 is due to these two materials' different responses to strain. The low stress response is due to the stretching of the elastin and the rapid increase in stress may be due to the stretching of the collagen which is arranged in a lattice structure and during the initial strain gradually aligns its self with the direction of deformation causing it to be susceptible to stretching.

5.3 The Skin Model

The skin model developed here works by gradually turning the geometric movements of the muscles and skull into forces which propagate through the skin mesh, moving the surface smoothly over the underlying struc-

tures.

5.3.1 The Multi-Layer Model

As was detailed earlier, real skin has several discrete layers, each with differing physical properties. For realistic physical animation, these layers must be recreated and simulated. This multi-layer nature of skin cannot be captured by a simple geometric model. In the developed system, this multi-layer nature is recreated with each layer having different physical characteristics which are formed from a combination of physical simulation and geometric methods. The produces a novel hybrid approach to the simulation of skin.

Figure 5.4 shows a single skin element, the top epidermal surface layer is shown in brown and the skin element is built down and deeper towards the skull from this level. This single element would represent the skin below a single triangle in the skin polygonal surface mesh. A network of these elements connected below every surface triangle makes up the skin model.

These elements are automatically constructed below the polygonal face mesh by the software. The top layer is the polygonal skin mesh supplied to the system and the underlying layers are constructed by the

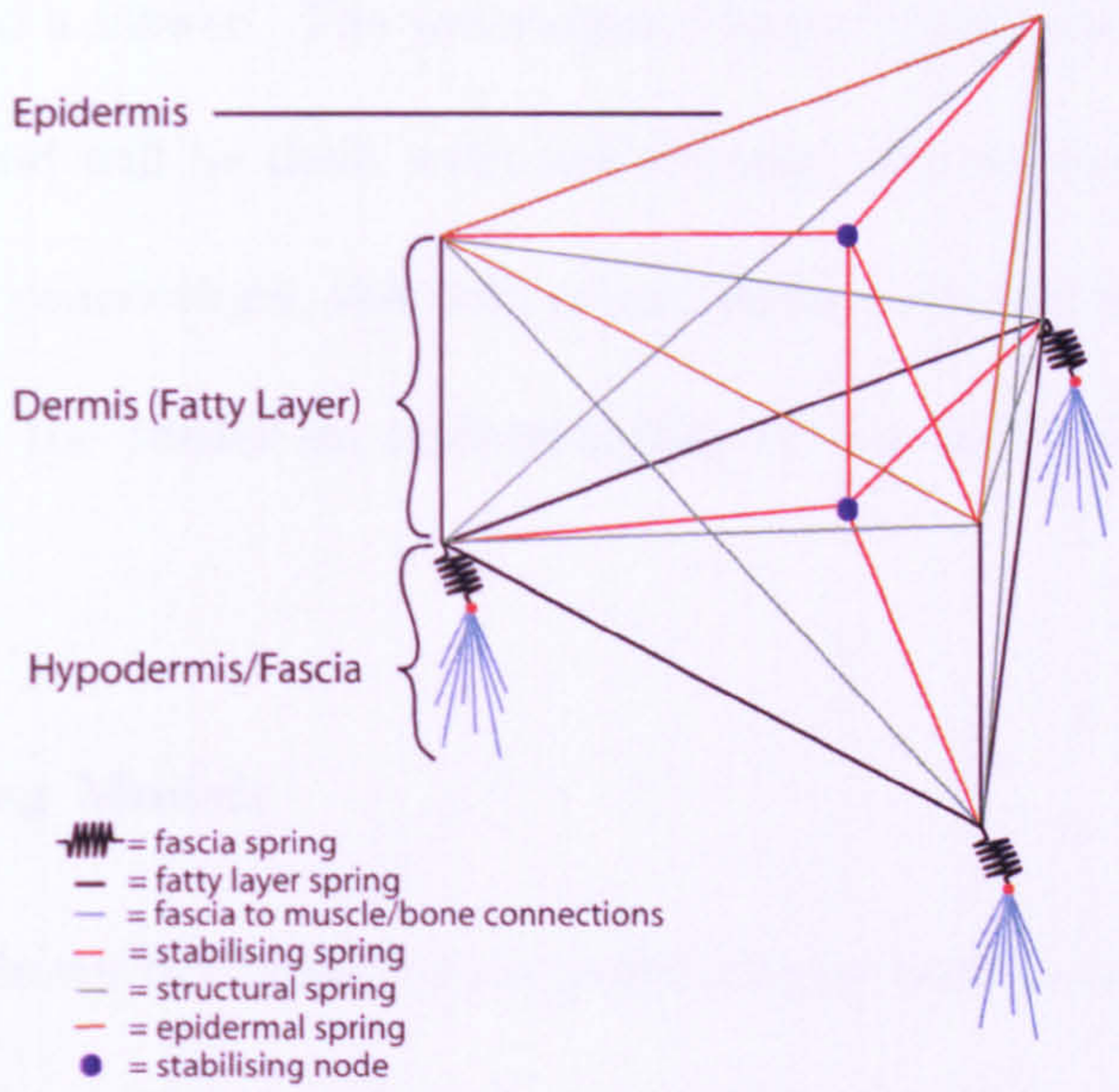


Fig. 5.4: A single skin element

software. The three layers in each element correspond to the three layers of real skin - the epidermis, dermis and fascia.

The bottom layer of the skin element connects into muscle and bone and as these structures move, they cause forces to move through the skin elements' layers up to the epidermal surface layer where the movements are visible to a viewer. The individual element layers each have unique properties and will be dealt with individually. As the layers all involve mass-spring connections, this sort of simulation method will be covered first to give the reader an understanding of the methods used in this model.

Mass Spring Models

The nodes shown in Figure 5.4 are point masses and the edges are modelled as varying types of spring. As was explained, these point masses are moved by connections to the underlying muscles and bone, the springs connecting the masses cause forces to propagate across the face in a realistic manner.

Simulation of mass-spring models is accomplished using the laws of Newtonian physics. Given a point mass m, the force f, acceleration a and velocity v are all related by the following equations.

$$f = ma \tag{5.1}$$

$$a = \frac{dv}{dt} = \dot{v} = \ddot{x} = \frac{f}{m} \tag{5.2}$$

$$v = \frac{dx}{dt} = \dot{x} \tag{5.3}$$

To simulate a point mass's movement through space the above equations would need to be numerically integrated over time. If two point masses are connected with springs, a spring force is introduced into the system.

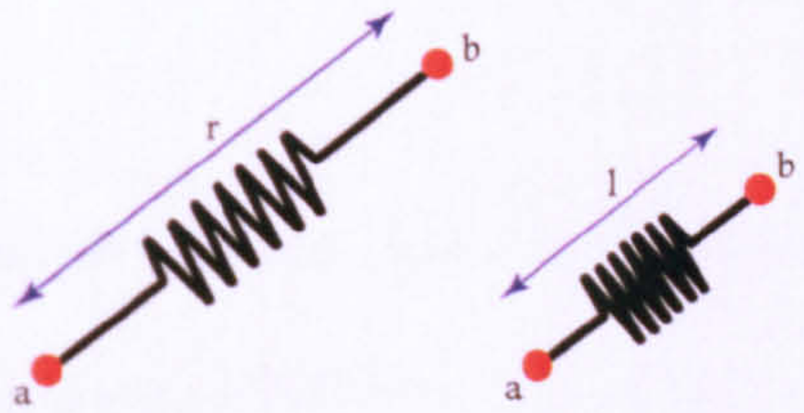


Fig. 5.5: Two point masses connected by a spring. The left image is the rest position of the spring, the right is the spring under compression

Figure 5.5 shows two point masses connected by a single spring. As the spring is compressed or extended, a force is generated which works to draw the masses back so that they lie at their rest distance from each

other. This force is modelled using Hooke's spring law [WB01], defined as:

$$f_a = -\left(k_s(|l| - r) + k_d \frac{\dot{l} \cdot l}{l}\right) \frac{l}{|l|} \tag{5.4}$$
$$f_b = -f_a$$

where:

f_a and f_b are the forces on a and b respectively, which are the particles at either end of the spring. $l = a - b$, r is the rest length of the spring, k_s is the spring stiffness constant, k_d is the damping constant and $\dot{l}$ is the time derivative of l.

The force applied on a point mass by a spring connected to it is proportional to the difference between the actual and rest length of the springs. The damping force is proportional to the speed which a and b are approaching or moving away each other. This damping term causes the springs to gradually come to rest rather than bouncing towards and away from each other.

In the real world, movement is subject to drag, such as from air resistance. In a numerical simulation, no such drag is present. This lack of drag and the small inaccuracies in calculations can combine to cause the system to become numerically unstable. To counteract this and to

increase the realism of the model, as well as spring damping, the entire system is subject to global viscous drag. This drag causes the particles to gradually come to rest in the absence of other forces. It is of the form:

$$f = -k_d v^2 \tag{5.5}$$

The drag is applied to every particle after all other forces have been taken into account. Without viscous drag, the particles' movements can quickly become erratic, excessive drag on the other hand makes the particles seem like they are moving through a viscous liquid.

Real skin has a more complex structure and response to stress than a simple mass spring network can produce. To replicate this, the skin model recreates the layers of skin and each is simulated in a way which reproduces the physical properties of real skin.

Creating each skin element is a three stage process, where the software loads the skin mesh, the muscles and the skull, and automatically builds up the three layers. This construction process is detailed next.

5.3.2 Creating the skin layers

The skin layers will be dealt with lowest level first. The fascia model will be explained followed by the fatty layer and finally the epidermis.

5.3.3 The Fascia

Fascia is the deepest skin layer, it covers and attaches into the bones and muscles of the face. These underlying muscles and bones have no restriction on their shape. For example, if the system was being configured for animation of a highly unrealistic fantasy character, there is no way of knowing how the muscles and bones will be designed and laid out. This means that the skin model must be able to attach onto any shape of underlying physiology. This adaptability is accomplished by projecting rays from the polygonal face mesh onto the underlying structures to find points where the skin will attach onto the muscle and bones.

Ray Casting

To attach the skin surface onto the underlying structures, a ray is projected from each vertex in the mesh in the opposite direction from the normal at that point. This ray should intersect with a triangle in muscle or bone. This point in the intersected triangle t becomes the *base point* for that skin node. If the ray does not intersect with any structure or it intersects considerably further away than its neighbours then it is flagged as dangerous. If the ray has gone through a hole in the object and attached to the other side of the skull, or it has missed the mus-

cles and skull entirely then it needs to be moved. This is accomplished automatically the majority of the time by finding the average position of the neighbouring rays which that point connects to and creating a point in space relative to them based on the original positions of each of these neighbouring points. If this automatic check does not produce satisfactory results, the user can manually position the ray hit-point in a more suitable place. This usually only happens on only a few vertices per model.

A particle defining the node for the fascia layer is then created at a distance of 60% from the hit point to the surface. This depth is based on the average fascia depth in humans and what was found to work well in simulations. It can be altered by the user to any value desired. This particle is then redefined in terms of the triangle t which the ray intersected. This redefinition is performed using the same method as the SOFFDs of chapter 4. A local coordinate system is created for the triangle t which the fascia node is then transformed into. When triangle t moves, due to muscle contraction or mandible movement for example, the fascia node will move relative to it.

The problem with defining the base points using a single ray, is that they are located in discrete steps across the skin surface and as such two fascia nodes which share a spring triangular element may have *base*

points on entirely different facial structures.

This split connectivity can lead to sharp divisions in the movement of fascia nodes as the underlying structures move. One node may be stationary whilst the two other fascia nodes in this skin element may be moving significantly.

To overcome this split movement, each fascia node is attached to multiple triangles in the underlying structure and the movement of the each node is an average of the movement of all base triangles.

Fig. 5.6: Rays and hit points on the face subsurface

To find the triangles to attach each node to, u random rays are fired alongside the inverse normal of the surface vertex being used. From experiment, it has been found that u works well if it is in the region of ten. The random rays are created within a twenty degree cone of the

reverse normal. The number of rays and the cone in which to project them is user configurable in the developed software.

5.3.4 The Fatty Layer

The fatty layer comprises the largest section of each skin element. In real skin, the fatty layer is able to freely move a small distance over the fascia layer. The layer is created by duplicating the points of the fascia layer to create the bottom layer nodes.

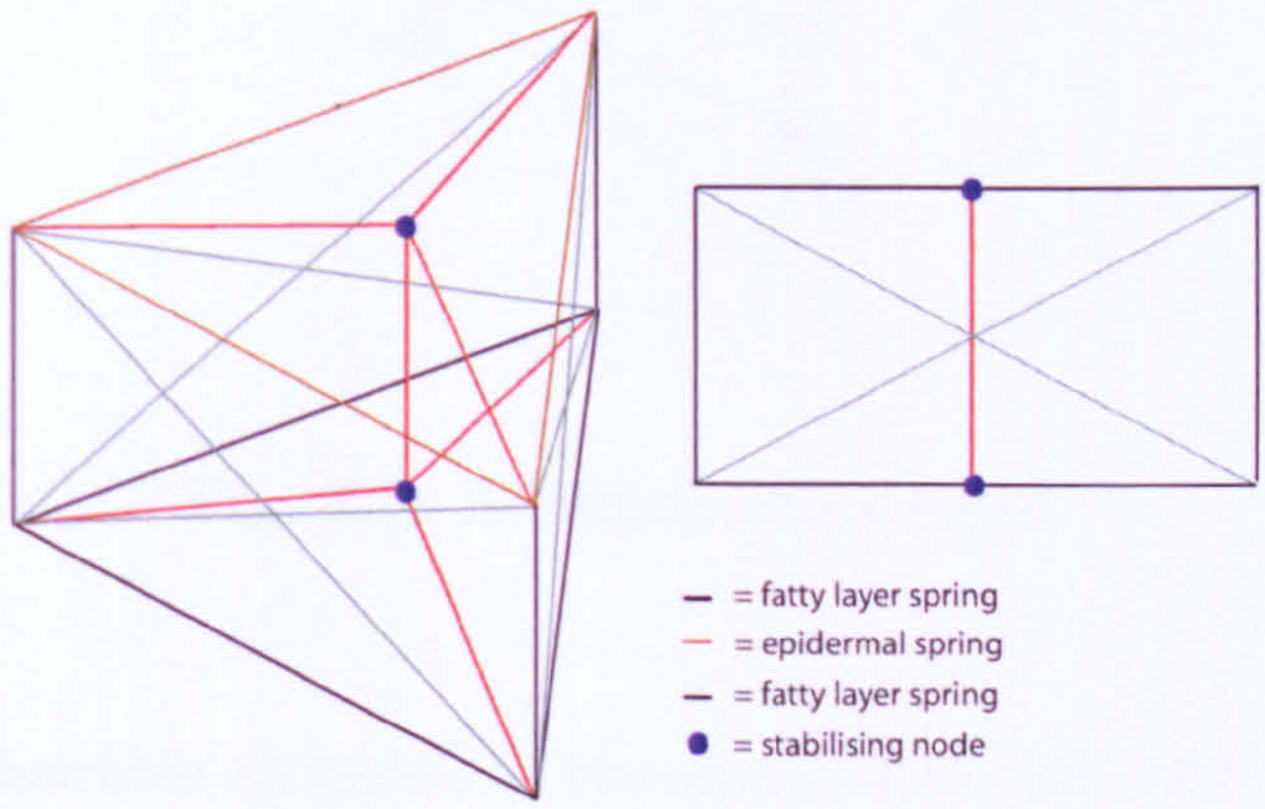


Fig. 5.7: The fatty layer from diagonally overhead and side on. The fascia layer has been removed for clarity but lies directly below the fatty layer.

The points in this layer are then connected up with springs to form a structure like that shown in figure 5.7. Two extra points are created in the centre of the points making up the epidermis and those making up the fatty layer. These nodes, shown by blue circles, are added to aid in the stability of the fatty layer and to aid in volume preservation. If a sufficient force is applied to an element then its structure could twist so that it enters a stable state whereby one of the nodes is on the opposite side of the element from its original position.

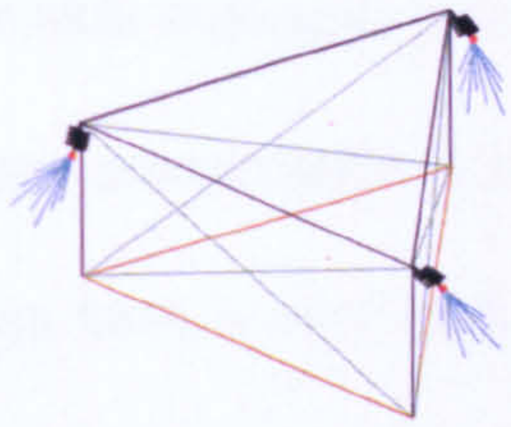


Fig. 5.8: A flipped skin element

This undesirable situation is shown in figure 5.8, which shows an element similar to the one in figure 5.4 except the epidermal layer has moved through the lower layers and would be penetrating muscle or bone. This is a known issue that arises with spring based models such as this [Coo98], and here it is controlled through the use of *stabilising* springs through the stabilising nodes. These springs make it much harder for elements to twist causing artifacts in the skin. The worst situation is

when the element twists so that it is in the exact opposite configuration from that which it started in and becomes stable. This is possible because the springs have no concept of direction, the force on their end nodes is based purely on the compression or extension of the spring. Due to the internal forces on each element being produced by the stabilising springs, it is much harder for elements to twist. Skin is quite incompressible due to its high water content. Some simulations preserve skin volume by measuring the volume of each skin structural part and scaling, or by adding a spring pulling the skin outwards to mimic the incompressibility of skin. In this work, volume preservation is accomplished by the stabilising springs. These springs have a very high resistance to compression and they force each triangular skin element to keep its structure and not squash or stretch too far out of shape.

To ensure elements do not twist over themselves or compress too much, the stabilising springs need to have a high resistance to compression. Unfortunately spring stiffness can only be increased so much before instability enters the simulation. For a given time step Δt and a given mass m there is a limit stiffness Ks above which the numerical simulation becomes divergent. This can be observed in the case of linear differential equations which this system can be reduced to if the natural lengths of the springs are assumed to be zero. The results from solving stiff equa-

tions are unstable if Δt is greater than the natural period of the system given by:

$$T_0 \approx \pi \sqrt{\frac{m}{K_s}} \tag{5.6}$$

If the stiffness of the springs is increased then the time step must be reduced below T_0. When increasing spring stiffness, T_0 rapidly shrinks causing the simulation to become increasingly slow. To make the stabalising springs stiff enough to keep the element from flipping and to preserve volume, the time step must be shrunk so low that the simulation could take hours to move forward.

As these very small time steps are undesirable the stablising springs are actually a hybrid spring-rod model. These springs are modelled as normal Hookean springs until their length decreases or increases too far from their rest length, at which point they become quasi-rigid rods [Pro95]. When a stabilising spring is forced to less than 60% or above 130% of its original length, the two end points of the spring are repositioned to make them 60% or 130% of their original length. This is accomplished by adding half the difference between the current length and the boundary length to each particle. This causes them both to move out along the direction of the spring and the spring length to be at

the boundary of 60% or 130%.

Moving particles explicitly like this will cause neighbouring springs to change length and possibly cause these springs to move outside the tolerance zone length. This is however, usually a local problem in the mass-spring network and so does not affect a large part of the mesh. To deal with any springs that have become too short or long by the movement of neighbouring springs, the entire mesh is scanned again and any springs affected are scaled appropriately. This process can be repeated as many times as is necessary to ensure all springs are within tolerances. Experience has proved this is rarely necessary however, and scanning and adjusting tolerances two or three times per simulation step has proved stable and produces good results. This adjustment process does take time and slows the simulation, however in practice the time it takes is very small, even with a network of thousands of springs, and is more than cancelled out by the time that would be lost by having to shrink the time step considerably to allow for very stiff springs.

Apart from the stabilising springs, there are the black and blue springs of figure 5.7. The black springs make up the main structure of the fatty layer and the blue diagonal springs are structural springs and are there to help resist shearing and twisting movements in the skin.

The fatty layer connects to the fascia layer via the fascia layer nodes

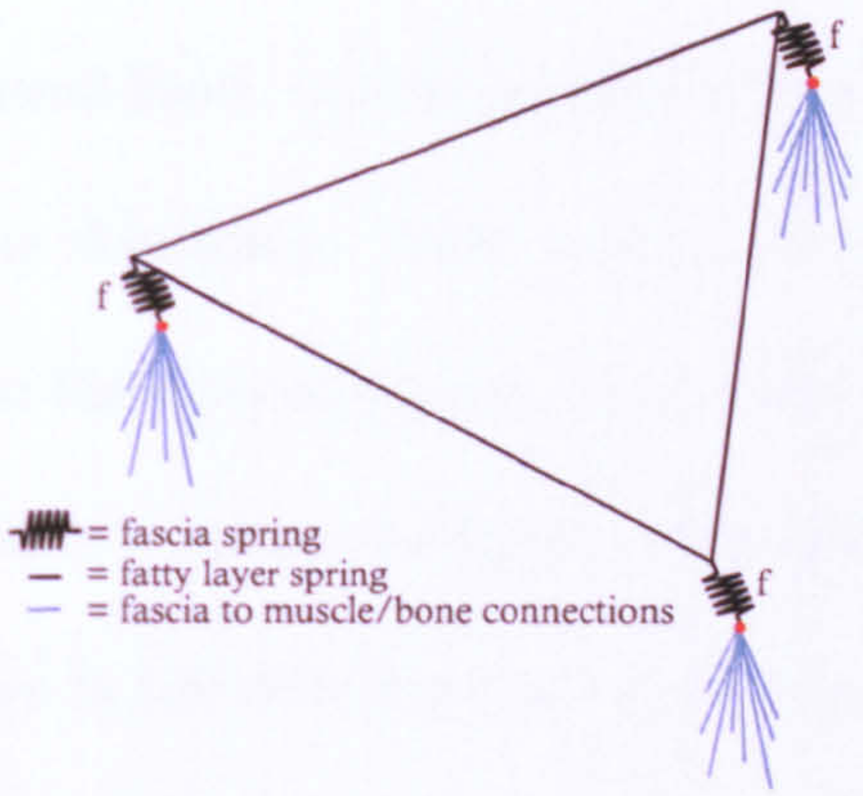


Fig. 5.9: The connections between the fatty and fascia layer

detailed earlier. These nodes are the ideal position for each fatty node assuming the underlying structures are moving evenly. In reality the underlying bone and muscle structures move independently and muscles near each other may be contracting in different directions and the skin needs to smoothly handle this. As the fatty layer should slide over the fascia layer, each fatty layer node is connected by zero rest length springs to the fascia nodes below it. This connectivity can be seen in figure 5.9. This connectivity keeps the fatty layer free to move when pulled by neighbouring skin elements but forces it to try and stay near its fascia base point.

The Epidermal Layer

To create the epidermal layer, a point mass is first created at the position of each vertex in the skin mesh. Each vertex's position in space is then geometrically *tied* to these point masses, so that the two are collocated in space and, as the mass is moved by forces, the mesh moves with it. The edges of the triangles in the skin mesh are used to lay out the springs on the epidermal surface. For every edge between two vertices in the mesh, a spring is inserted between the two point masses corresponding to those vertices. If the mesh is set up so that multiple edges connect between the same vertex pairs due to each triangle having its own edge set, only one particle or spring is used for each collocated vertex or edge respectively.

Skin in humans has a non-linear response to stress. This could be seen earlier in figure 5.2. This non-linear response is modelled in the epidermal layer by varying the spring stiffness constant as the springs change in length. The spring stiffness is controlled here by a non-linear function depending on the spring length:

$$k_f^i = k_i^s(1 + 0.5(|l_i| - r_i)^2) \tag{5.7}$$

where k_f^i is the stiffness for spring i based on its current length. k_i^s is the default stiffness constant for spring i, l_i is the current length and

r is the default length. This function was arrived at by extensive testing of the response of springs and the skin mesh to loads applied by muscle forces and mandible movement. By dynamically altering the spring stiffness like this, the springs respond more realistically to loads. The springs will easily move so far or be compressed so much before the stiffness rapidly increases. This mimics the way skin will can be stretched or compressed a small amount before strongly resisting forces.

If these springs are highly compressed or stretched then the stiffness of this function can rapidly reach large numbers, which leads to the the time step in the system having to be reduced accordingly. If the springs become stiff, the time step would have to become small to compensate for this and so the simulation becomes slow. To stop this happening, the spring stiffness is clamped to a reasonable maximum based on the minimum time step required.

This multi-layer model allows the movement of muscles and bone to propagate smoothly through the facial mesh and manipulate the skin in a realistic fashion. The movements of the skin are simulated by numerically integrating the forces in the mesh to iteratively step through time and push the simulation forward.

5.4 Simulation

The actual movement of the skin is governed by the forces that propagate through the triangular skin elements due to muscle and bone movements. These movements move the fascia nodes, which pull the fascia springs connecting the fascia to the fatty layer. The force generated by the movement of these springs pulls the fatty layer nodes, which propagates the force through the fatty layer up to the epidermis where skin movement is visible to an observer.

Due to the hybrid nature of this system, the only real force to consider in the simulation stage is that caused by spring compression or extension. The movements of the muscle and bones directly move the fascia nodes, which in turn pull on the springs connecting them to the fatty layer. This generates a force which causes the fatty layer to move, which will in turn move the springs of the fatty and epidermal layers. The non-linear spring stiffness and quasi-rigid spring-rods of the stabilising layer mean that volume is semi-maintained and the skin keeps its shape under pressure with no need for extra simulation parameters.

Due to this compactness in the model, the force on any point mass particle in the system can be modelled as a second order set of differential equations:

$$f_i = m_i \frac{d^2 x_i}{dt^2} + \phi \frac{dx_i}{dt} + S_i \tag{5.8}$$

where m_i is the mass of particle i, ϕ is the coefficient of velocity damping and S_i is the total spring force on node i.

To solve this equation for each time step, an explicit Euler method is stable enough. The explicit Euler method can have problems with numerical instability, but as the stiffness in the springs is handled through varying constants and quasi-rigid rods, it works well in this situation. For each time step Δt, the simulation is integrated over time as follows:

$$\begin{aligned} a_i^t &= \frac{1}{m_i}(f_i^t - \phi v_i^t) \\ v_i^{t+\Delta t} &= v_i^t + \Delta t a_i^t \\ x_i^{t+\Delta t} &= x_i^t + \Delta t v_i^{t+\Delta t} \end{aligned} \tag{5.9}$$

where a_i^t, v_i^t and x_i^t are the acceleration, velocity and position of particle i at time t.

5.5 Conclusions

This chapter has presented a new skin model that is automatically generated from a triangular mesh. The multi-layer model connects automat-

ically onto the underlying muscles and bone and moves smoothly along with these structures during muscle contraction. The skin behaves in a manner similar to real skin due to varying spring stiffness constants, and it maintains its volume well enough for animation using rigid rods to stop the skin elements compressing or extending too far.

The muscle and skin models that have been detailed are full models for facial animation. The next chapter will explain the testing processes that was undertaken to determine the quality of they system.

6. Testing and Results

The muscle and skin models detailed are designed to create realistic facial animation. Proving that animation is realistic is a difficult task as there are no real metrics to define realism in animation. This realism can only really be tested by comparison to real facial movement. There are some facial movements that are created by clearly defined muscle movements [Mus02], and to test the muscle and skin model it was decided to select a set of such movements and, by contracting the synthetic muscles, the skin should behave in a realistic manner. By comparing the motion of the synthetic skin and muscles to that of a real person, the realism of the model can then be shown.

As the muscle and skin simulation are three dimensional models, the realism of these models was compared to a real subject moving their face into the same positions. In depth examination of the simulated model and a real person is not a simple task, so a a real person was captured in 3D moving their face into the positions for the test.

The capture was performed using a dynamic 3D scanning device [Neb01]. The device can capture 3D models of subjects at 25 frames

per second which can then be played back and examined from any viewpoint, creating a sort of 3D film.

To create the simulated animation, two models were used. The first (head A) was a freely available model [Osi03]. This model had a good mesh structure for facial animation with the vertices and edges lined up along natural creases in the face. To create the second test head (head B), it was decided to use a model similar to that which would be captured in 3D.

The meshes from the dynamic scanner tend to be smoothed and can lack in detail, so a static high resolution model was captured and used. The models captured from the high resolution capture device have a mesh totally unsuitable for animation, with a random triangulation across the surface. To produce a model suitable for animating, head A was conformed onto the surface of the scanned model. This created a head with the same connectivity as head A, but with the shape of the scanned model. This process took several steps and is outlined in appendix A.

After conformation, the skull and muscles were created on head A and then transformed onto head B using the methods detailed earlier.

6.1 The Tests

Ideally when testing the system, the realism of the model would be tested along with how it compares to similar models. The problem with comparing the system to a similar model is that very few such models are publicly available. As this is an active research field the current state of the art is always shifting and very few researchers release their system into the public domain. This is partly due to facial animation research often being done in conjunction with entertainment companies who guard their work against all competitors. Due to this, the only publicly available facial animation models that were available for testing were outdated and the results would have proved very little.

To test the realism of the model, it was decided to compare the system producing different facial expressions against real human subjects moving their faces into the same poses. Four poses were chosen and each is common in speech and emotive facial movement. These four poses represent a wide range of motions. No more than four poses could be tested due to time constraints on the scanning equipment needed for the tests.

The four expressions can be described by FACS Action Units [EF78] and the poses all test different groups of muscles. Table 6.4 shows images

of the muscles used in each pose and details which muscles are used in which pose and which Action Unit this corresponds to.

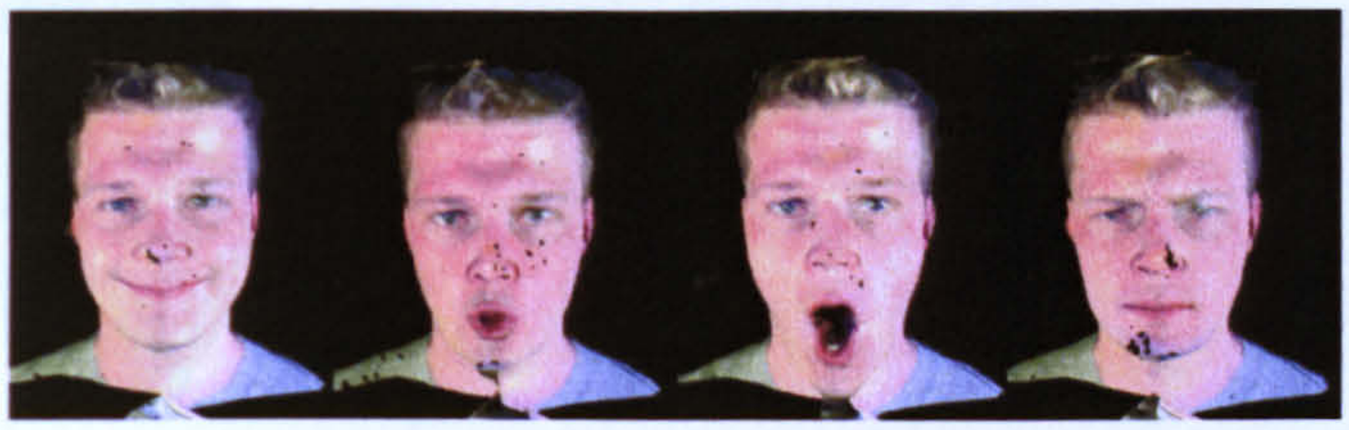


Fig. 6.1: The four poses used for testing.

	FACS Units	Muscles Used
	Lip Corner Puller Cheek Puffer	Levator Anguli Oris Zygomaticus Major
	Lip Funneller	Orbicularis Oris
	Jaw Drop	Masseter Relaxed Temporalis Internal Pterygoid
	Brow Lowerer	Corrugator Supercilii Depressor Supercilii

6.1.1 Making the Synthetic Data

The facial expressions selected for testing use quite clearly defined muscles and so creating a synthetic version is a case of contracting these muscles to match the real subject. The speed of muscle contraction varies between subjects and also depends on mood and environment. Therefore, the muscle contraction speeds were hand tuned to try and match the captured subjects.

6.2 User Testing

To test the realism of the simulation, volunteers were presented with the models and asked to rate them on two scales, realism and similarity to a scanned model. The number of volunteers was set at twenty so that a range of different types of people could be tested. A larger sample could possibly have provided more accurate results but it was desired to get a balanced range of subject types and it was not possible to extend the trial to a larger group within the time scale due to a lack of subjects with specific expertise. The group was made up of several different groups of people. *Animators*, who work professionally creating computer animation and are very quick to pick out any flaws in movement which is not correct. *Doctors*, who are used to working with the human body and are

susceptible to odd movements in the face due to examining for clinical conditions. *Computer Graphics Professionals*, who deal with computer simulations all the time but are not experts in the field of facial animation. Their work mostly involves simulating people and animals but more from a lighting and texturing perspective. The final group was made up of ordinary people who had no real computing, graphical or animation expertise. This selection of people were chosen to provide a critical analysis of the model. The sample is made up of more than half with people who are more likely to spot any errors in the model. This was purposefully done to try and get good feedback on what may be wrong with the system. From informal testing throughout the development of the system, it was noted that the average person is quite skilled at picking out a problem with movement but finds it hard to say what exactly is wrong with the model, by increasing the number of trained eyes in the test it was hoped to receive more direct, less vague feedback. To actually score the tests, the testers were asked to rate animations on two categories with a value of between one and ten.

For the realism test, the volunteers were told that one was defined as being completely unrealistic and ten is perfectly realistic, with five being reasonably realistic.

The similarity test had the scale of one being nothing alike, ten being

an exact copy and five being reasonably similar.

Realism Test To test the realism of the model, the 3D animations were presented to the volunteers in a random order and the user would gauge their realism on a scale of one to ten. Users were instructed to concentrate on the movement of the skin surface and ignore the non-moving eyes in their assessment of realism.

Similarity Test To find out how well the synthetic and real movements match, volunteers were presented with the animations and the real captured data simultaneously. The subjects then had to rate on a scale of one to ten how well the movements of the synthetic animation matched those of the captured animation. Some of the textures of the scanned heads had artifacts in them which can cause them to flash when played back. The subjects were asked to ignore the flashing quality textures and concentrate only on the movements of the skin surface. This could obviously have had some impact on how the users compared the simulated model to the scanned models but when asked most users felt that they had managed to concentrate successfully on the movements.

To actually rate the simulations, the volunteers used an interactive viewing program with two modes. The first mode presents the volunteer with the simulated head in 3D and the user can rotate, pan and zoom the model whilst playing back the animation. This mode was used to allow

the volunteer to rate the realism of the simulation. After the user has selected a value for the realism, the view splits in two and displays the simulated head alongside the scanned head. In this mode the scanned and simulated heads can be moved into and out of an expression by the user allowing the user to rate how close the two are. Once the user had answered for one simulated head, the same process was repeated for the other head.

The order in which the expressions and head were presented was randomly defined, so that the results for individual expressions would not be changed by always being first or last to be seen by the user. The testing process took no more than a few minutes and volunteers were asked to record any comments they had on any of the models whilst they undertook the experiement.

6.2.1 Test Images

The next ten pages show images from the testing. Each of the four expressions is shown frame by frame along with the muscle simulation and the two simulated heads. To save space, only the muscle simulation is shown from more than one angle, but the front on view gives a good image of the animations presented to the users.

The open mouth images clearly show the muscles around the lower lip when the mouth is opened. These were not visible in the user tests, but were left visible here to show the orbicularis deforming as the mouth opens.

Fig. 6.2: Output of the frown test.

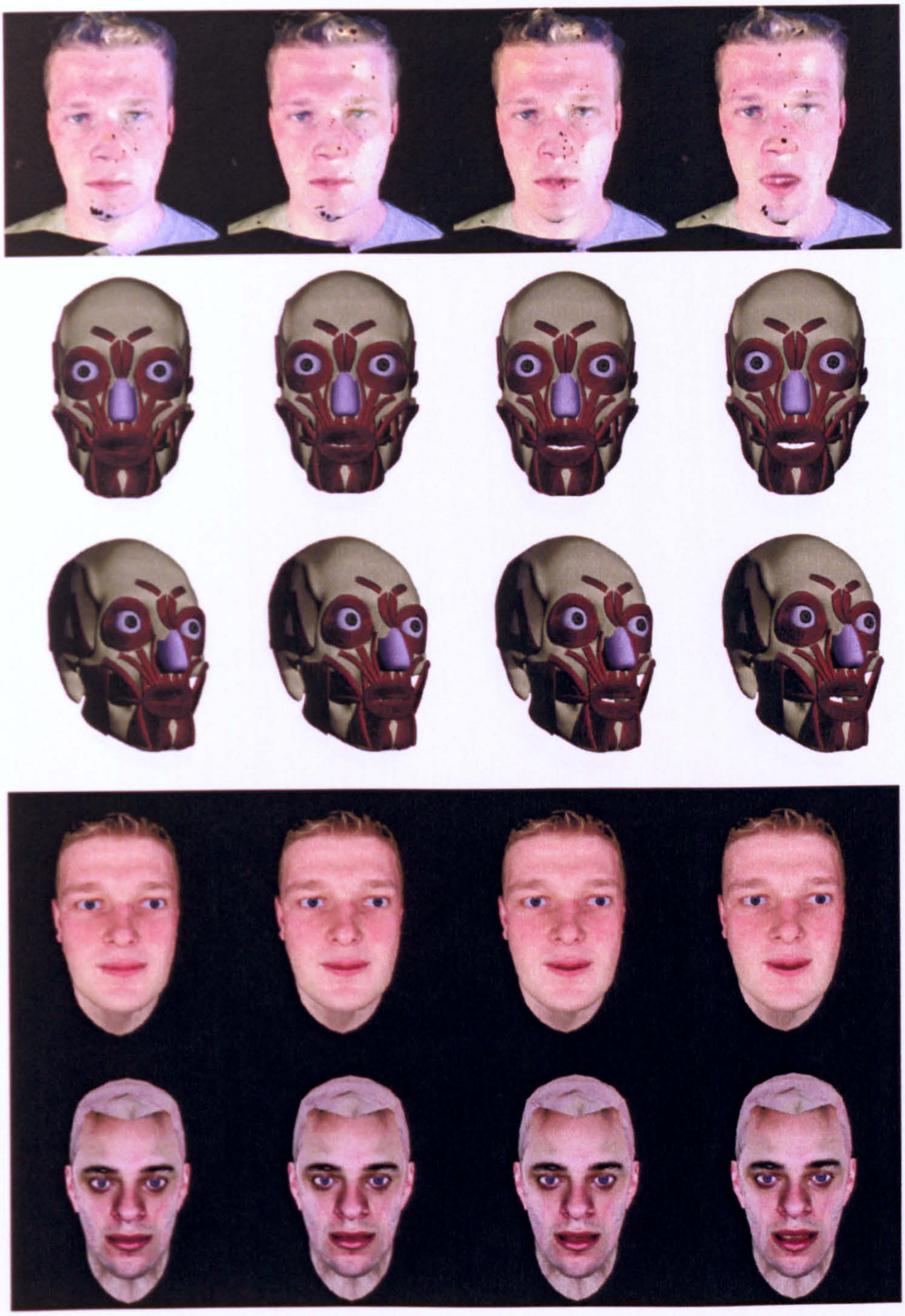


Fig. 6.3: Output of the Open Mouth test. (1)

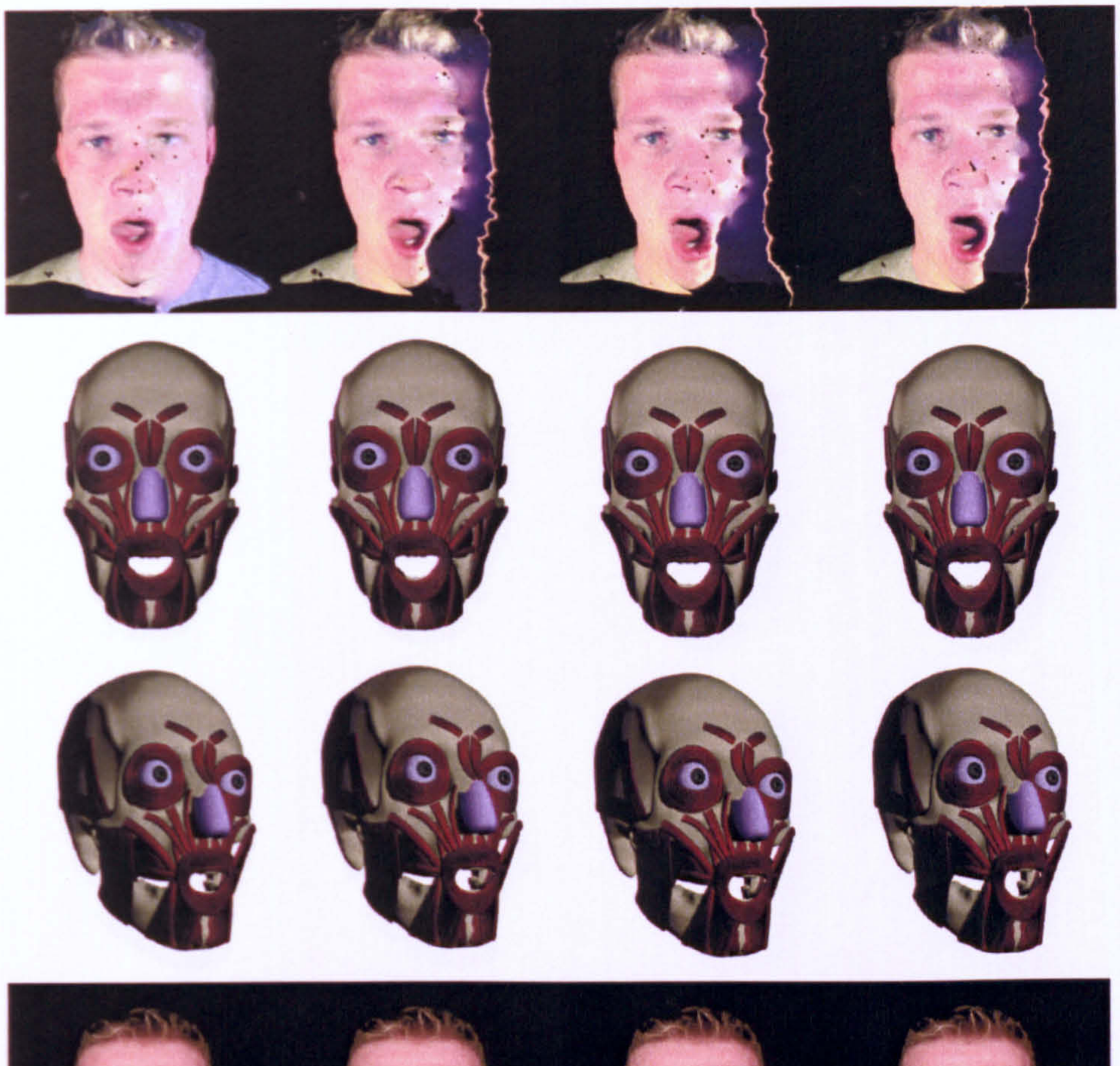


Fig. 6.4: Output of the Open Mouth test. (2)

Fig. 6.5: Output of the Open Mouth test. (3)

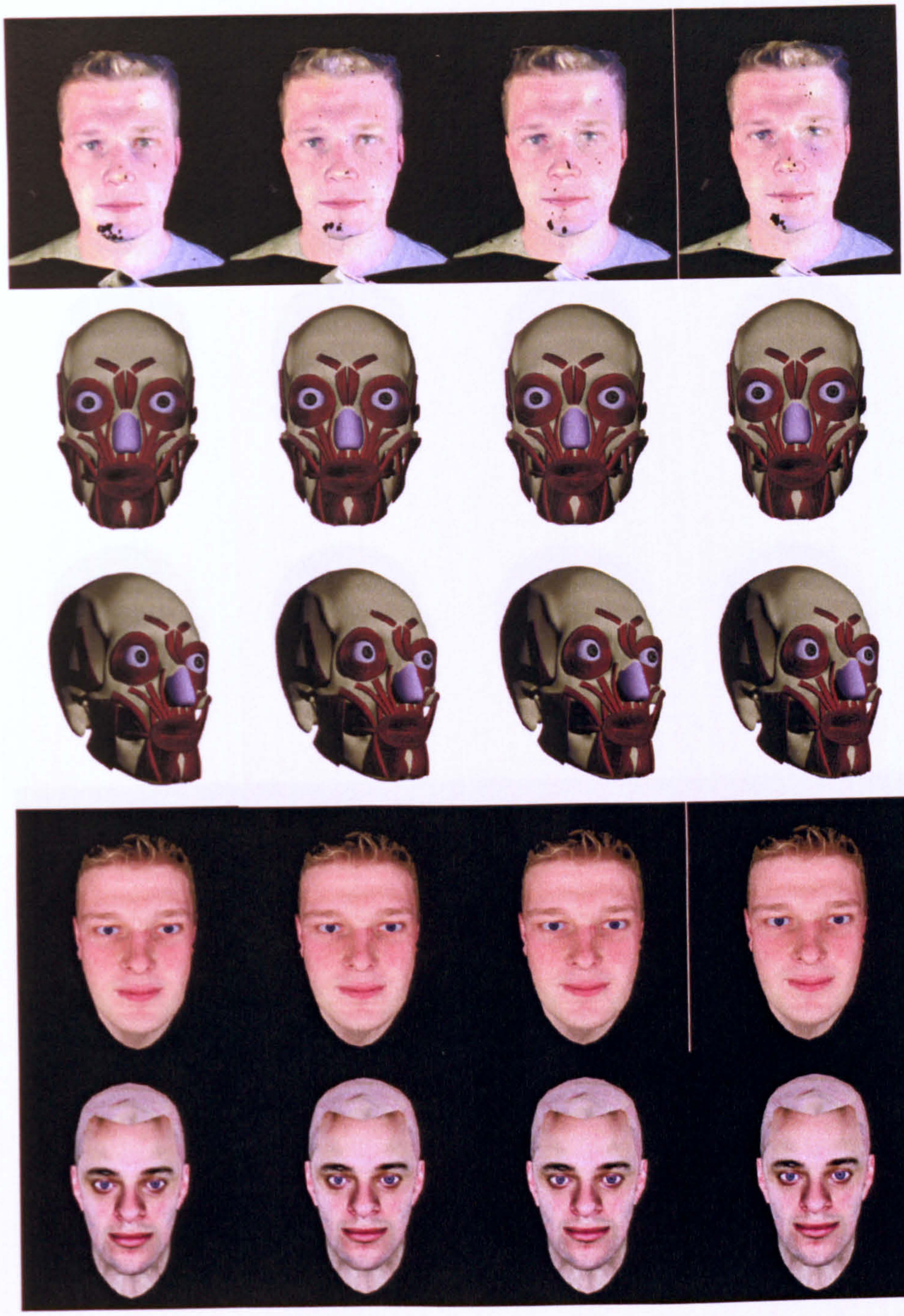


Fig. 6.6: Output of the smile test. (1)

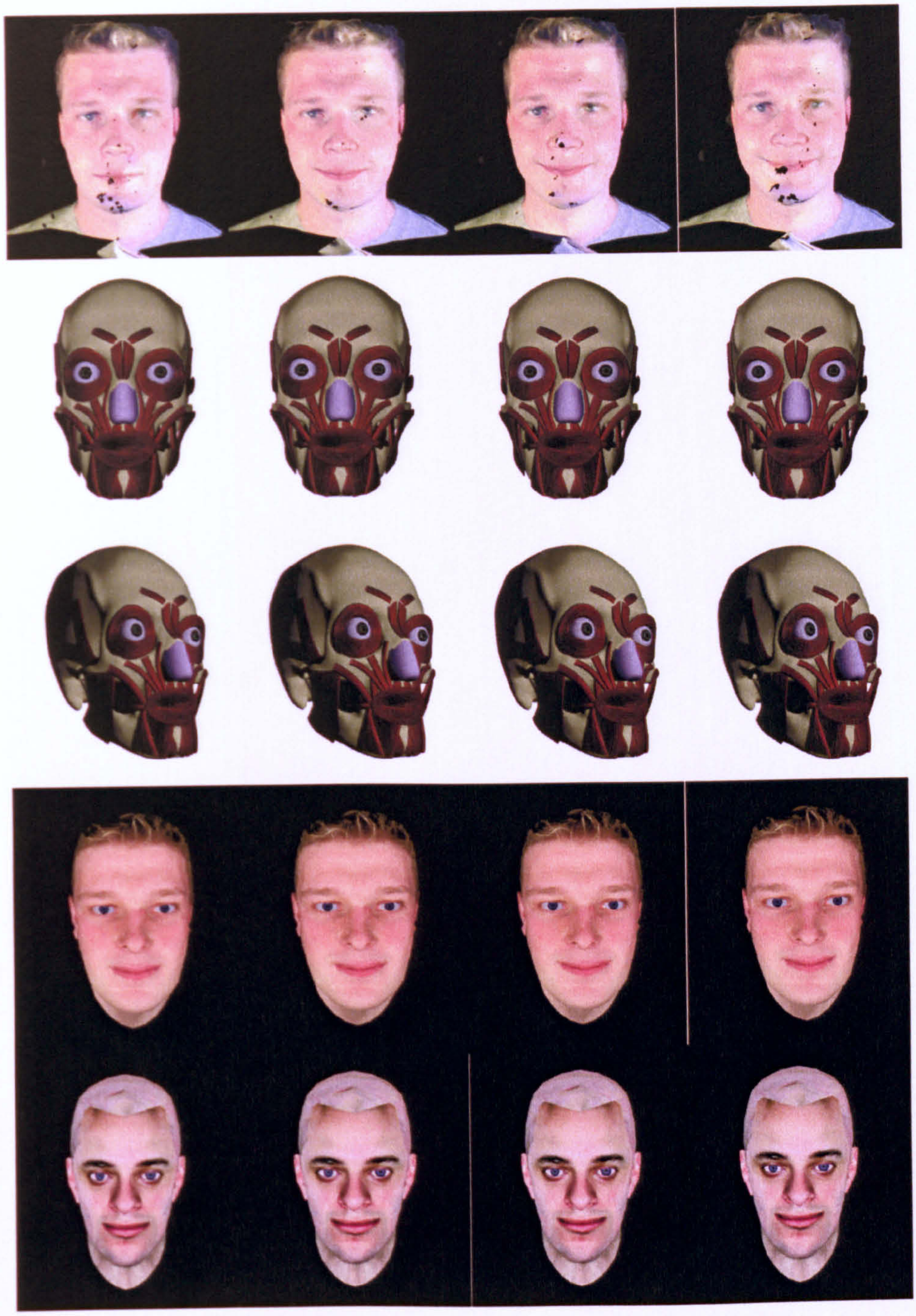


Fig. 6.7: Output of the smile test. (2)

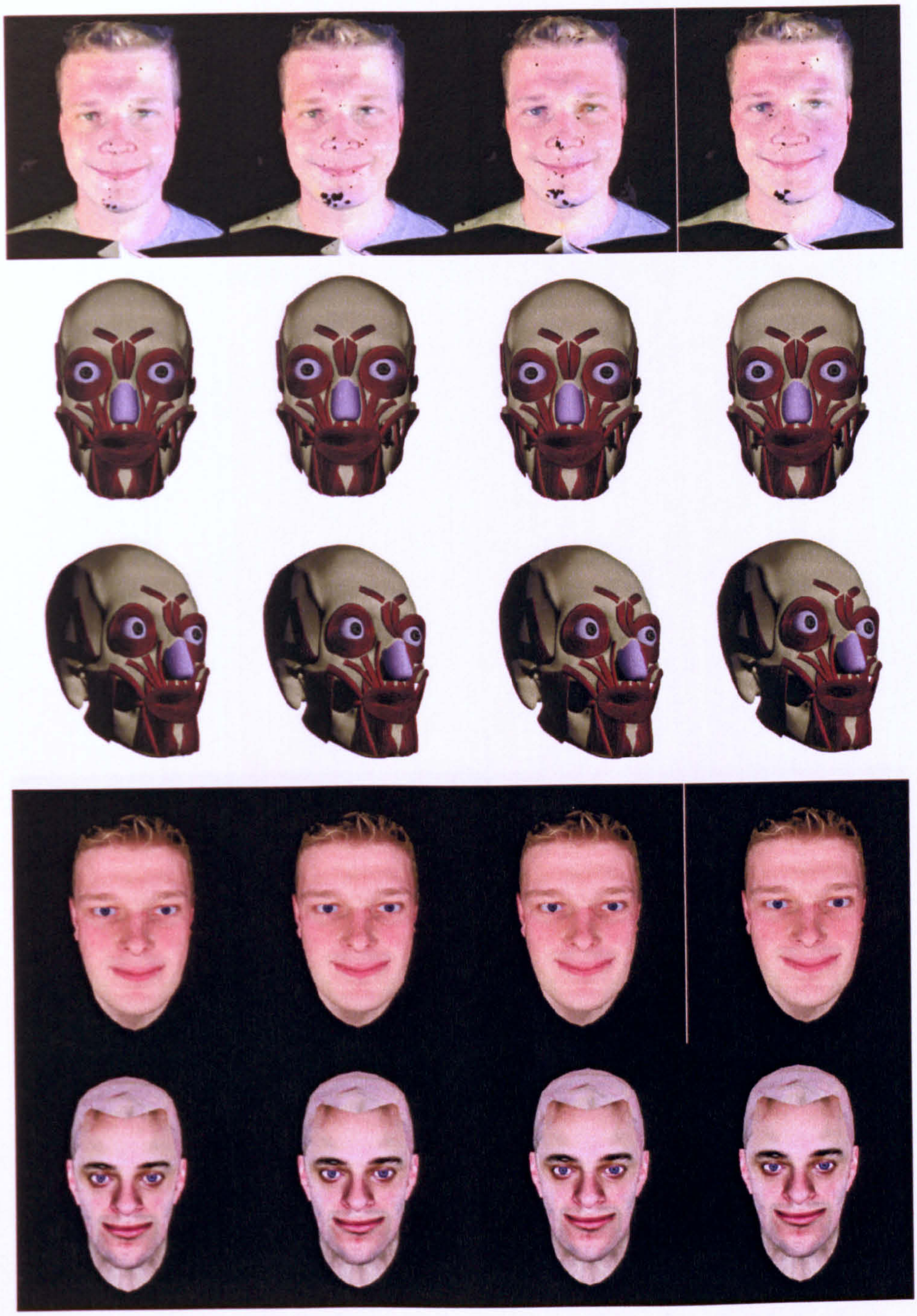


Fig. 6.8: Output of the smile test. (3)

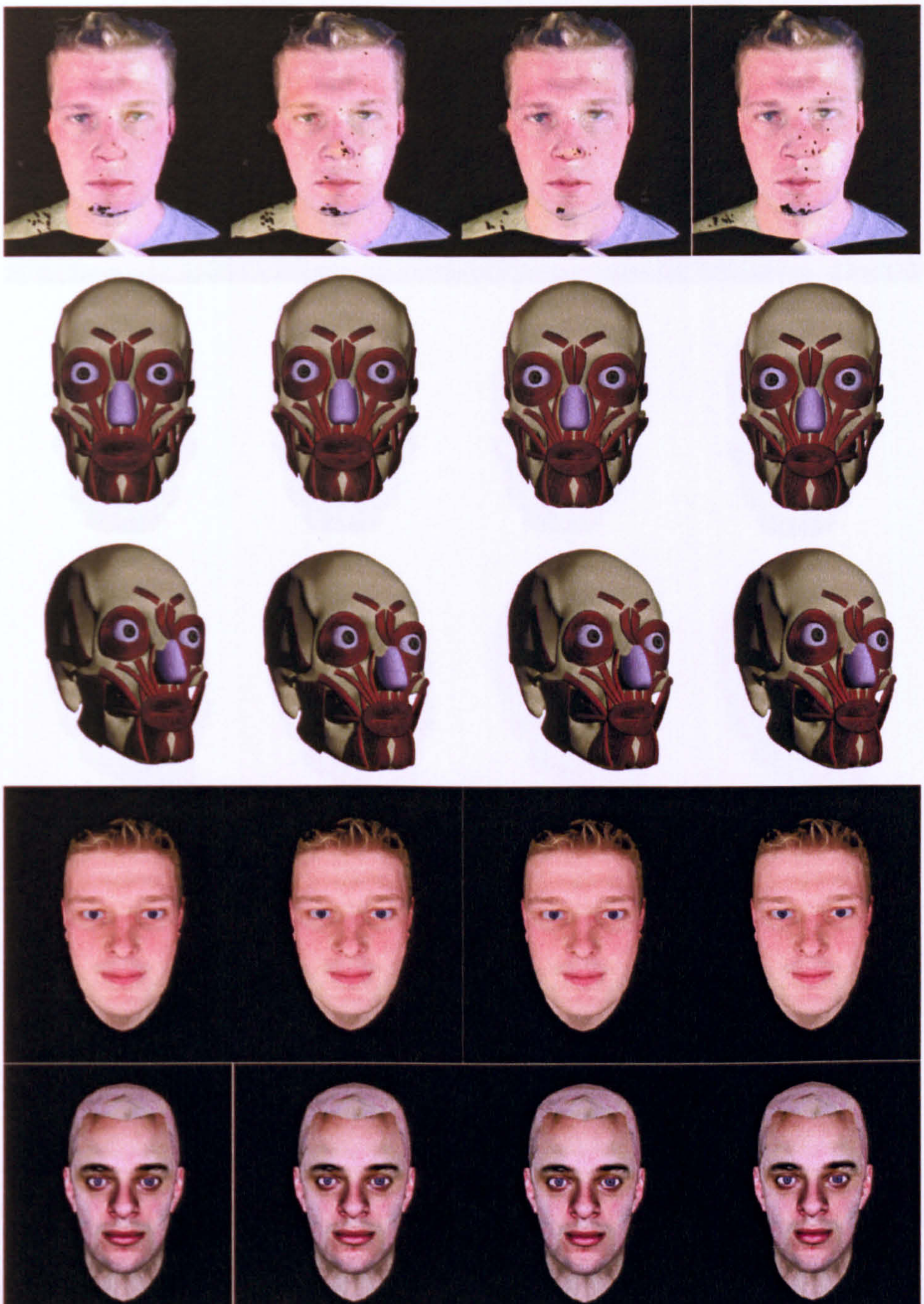


Fig. 6.9: Output from the pout test. (1)

Fig. 6.10: Output from the pout test. (2)

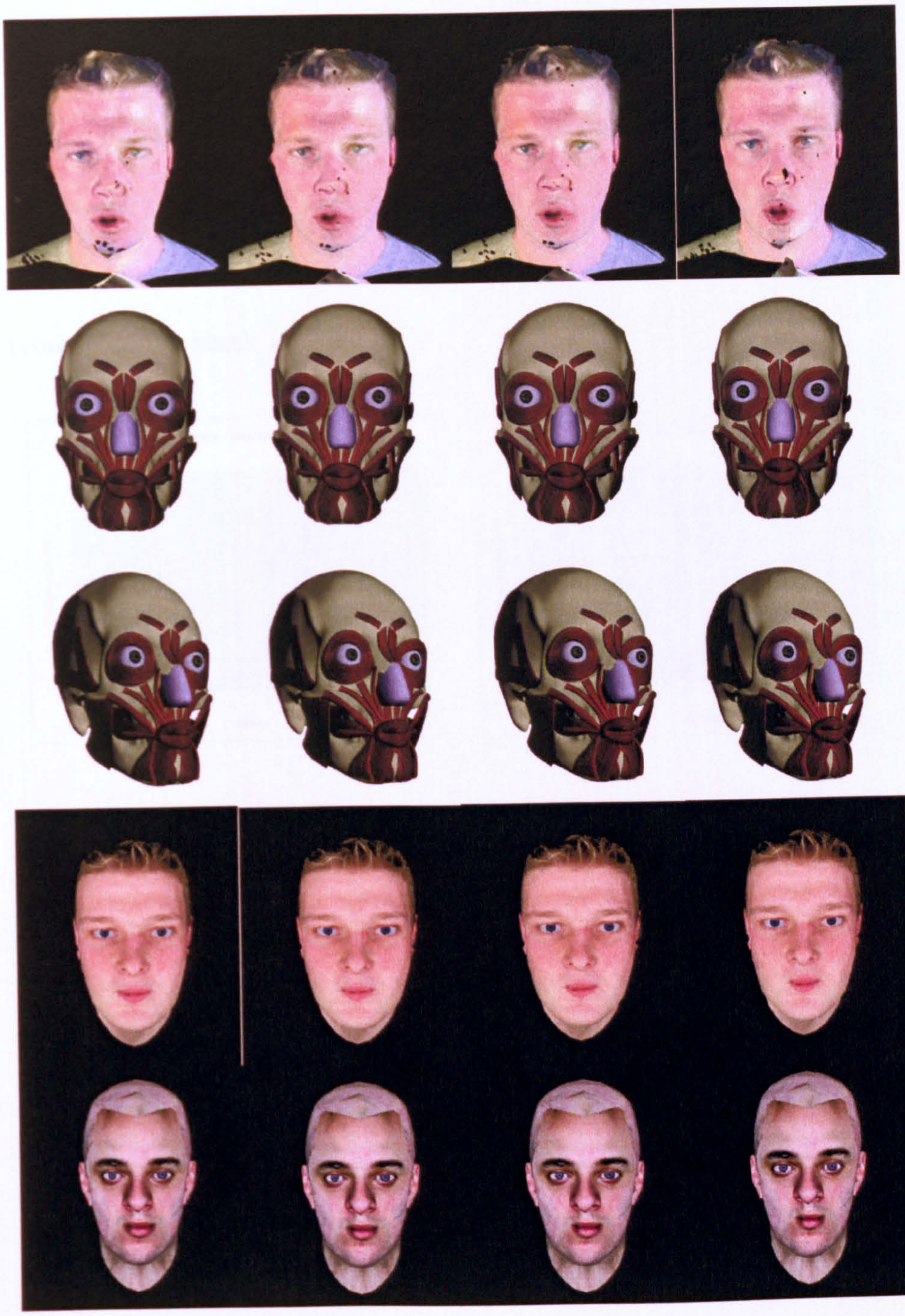
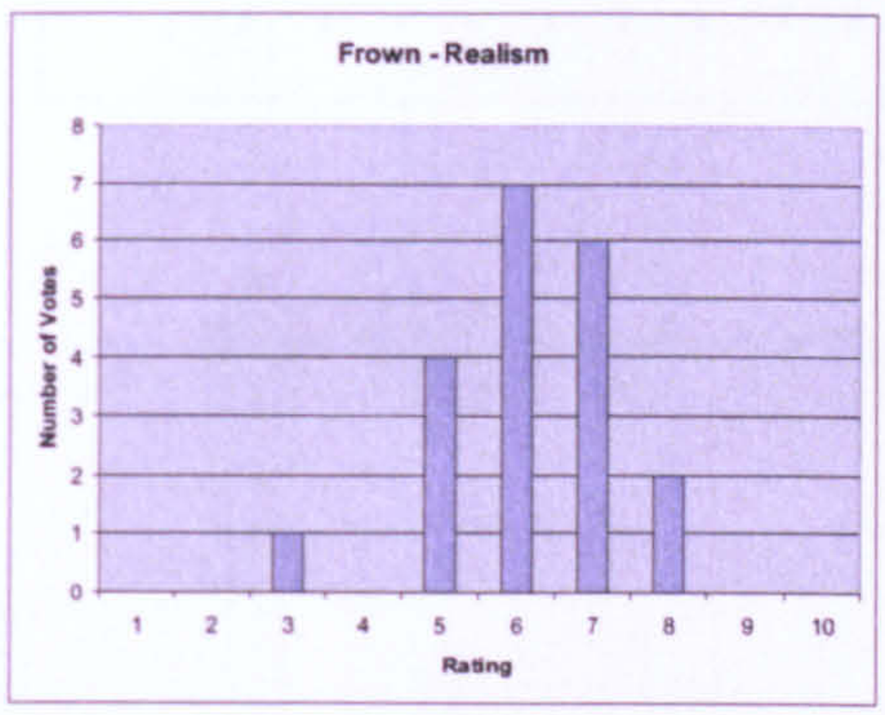


Fig. 6.11: Output from the pout test. (3)

6.3 Results

The graphs below show the ratings for each expression presented to the volunteers. After the experiment, it was noted that the results for the two simulated heads were very close and so these were averaged and presented as a single graph.



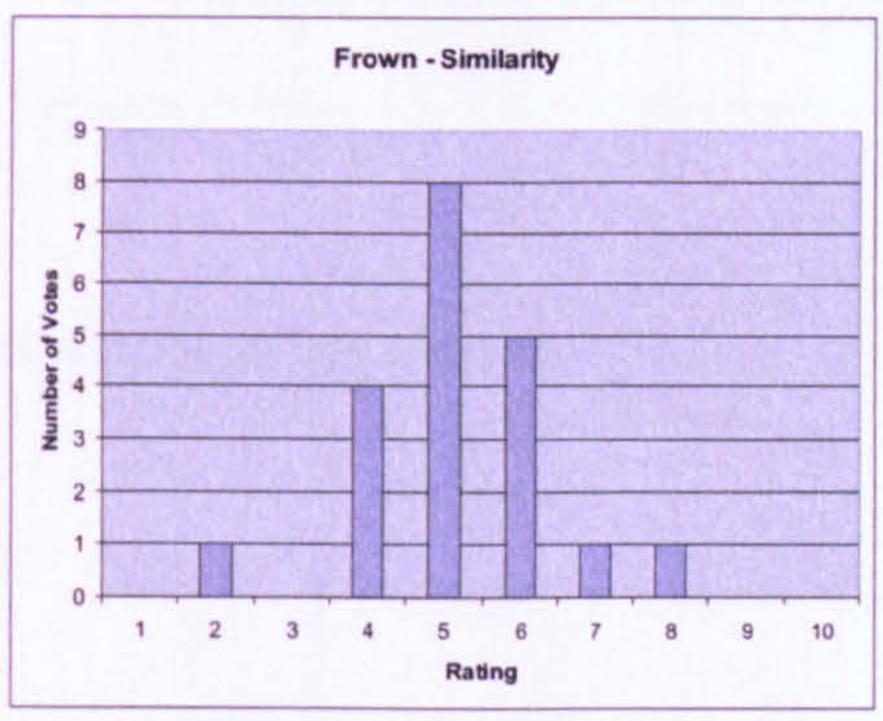


Fig. 6.12: Results for the frown simulation.

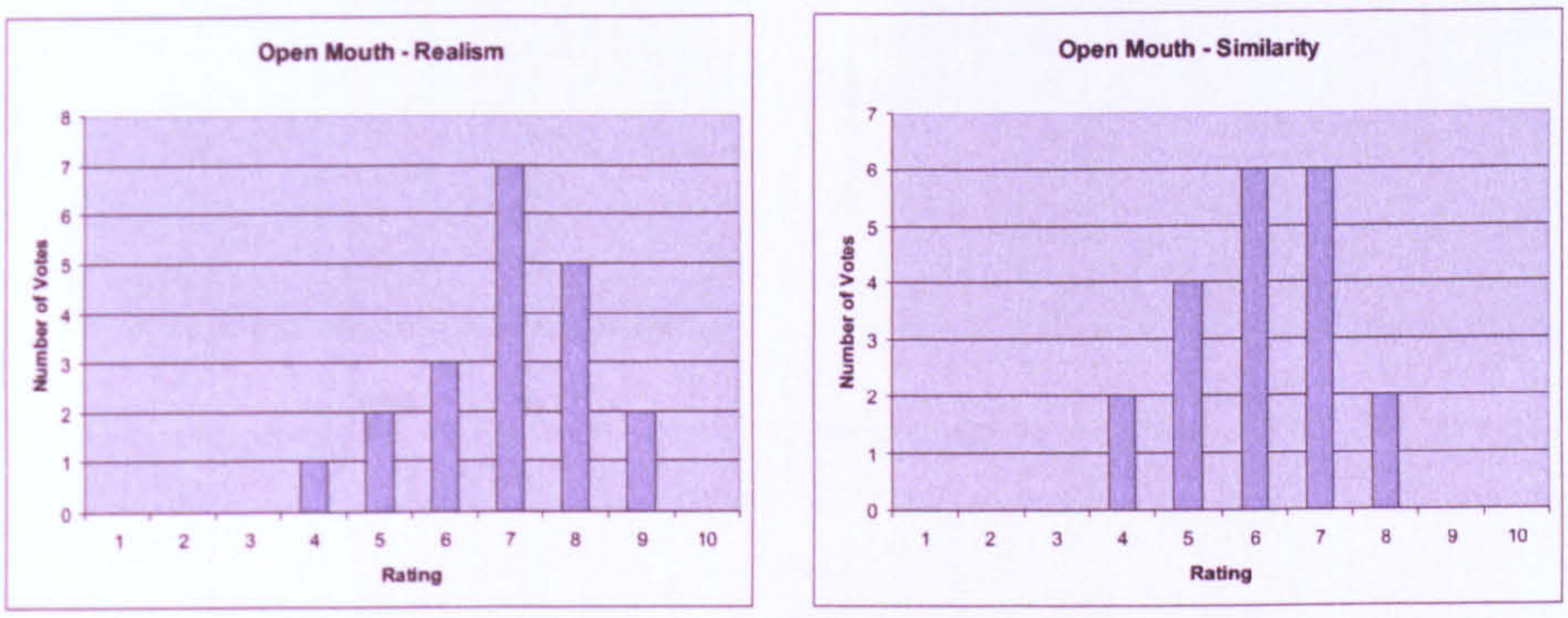


Fig. 6.13: Results for the mouth opening simulation.

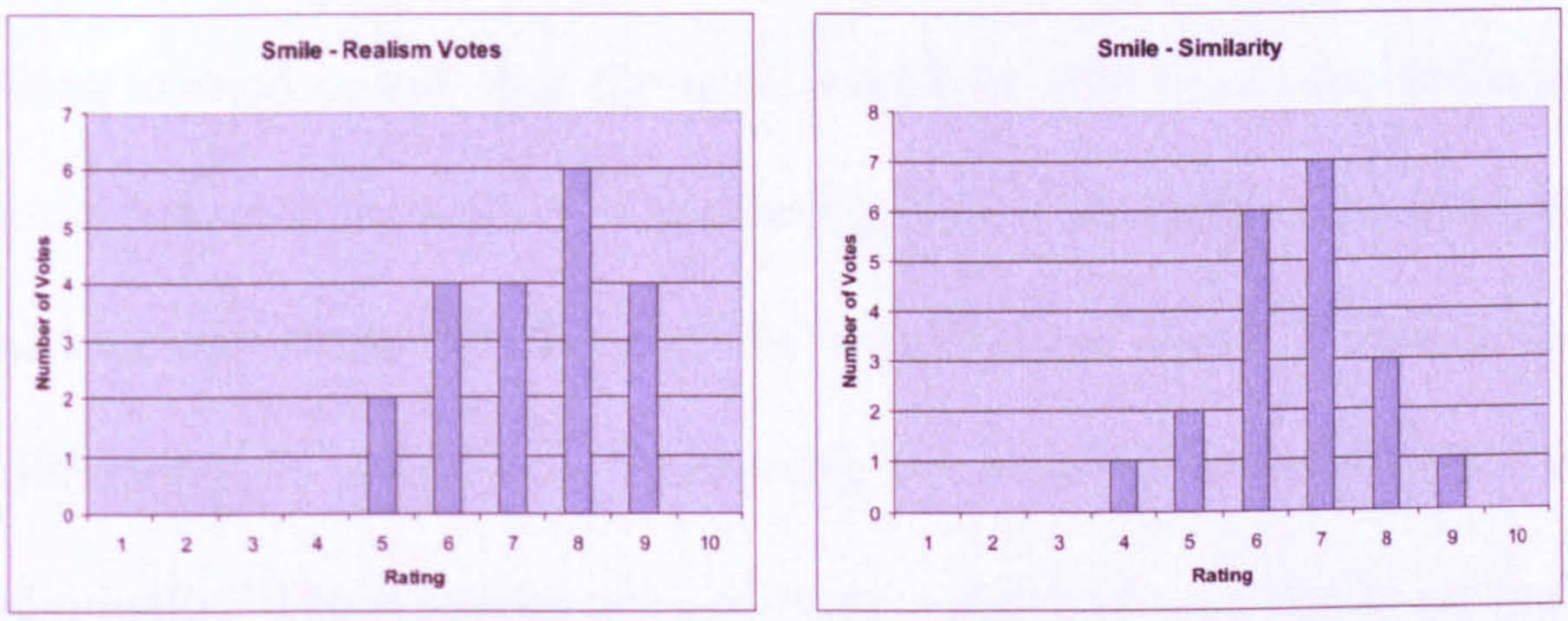


Fig. 6.14: Results for the smile simulation.

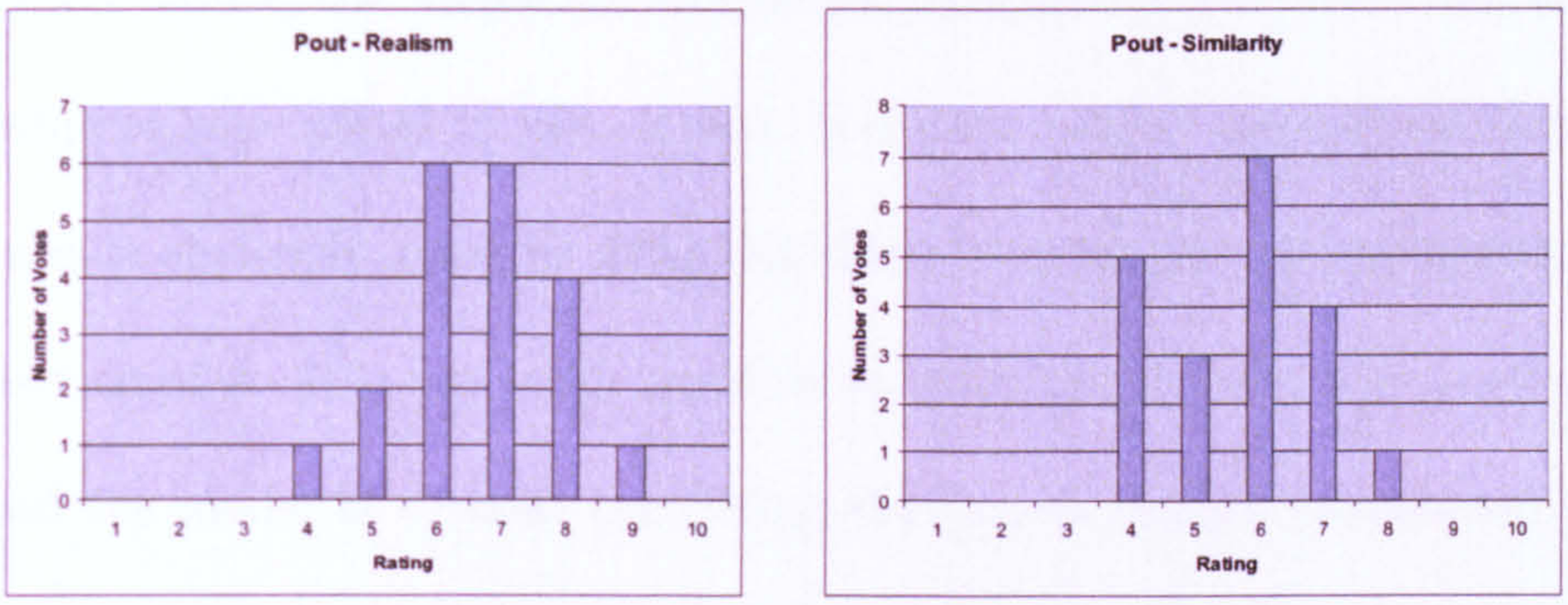


Fig. 6.15: Results for the pout simulation.

6.4 Observations and Conclusions

The user testing gives an overall positive picture of the simulations. The test volunteers were given a tool which allowed them to view the models from all angles and compare them in detail to the scanned heads. This testing method meant that the users would be able to see any inconsistencies between the scan and simulation and if the model did not look real from any angle then it would be very obvious. As each pose tested a different part of the face, the results for each test will be looked at individually. These results are obviously subjective as there is no hard data to compare it to. This is a problem in facial animation research and one model is almost never compared during testing directly to another for the intellectual property reasons explained earlier. Due to this sub-

jective nature, the results are interpreted compared to the scales that the subjects were asked to use. Again, how each subject interpreted those rules is obviously going to differ but these tests are just trying to gauge user opinion. It is not really possible to define empirically how good or bad the model is without comparing the precise surface movements of two models which was not possible due to the noise in the data of the captured models A.

The results for each test tended to congregate around a value and so the mean average and standard deviation were calculated to get an idea of the average realism or similarity, and its accuracy for each expression. The standard deviation for each test was in the range of 1.17 to 1.32 indicating that the responses were mostly clustered around the mean and we can take the mean as a good indicator of the responses overall. The mean and standard deviation for each test can be seen below.

Expression	**Realism (mean)**	**Realism (σ)**	**Similarity (mean)**	**Similarity (σ)**
Smile	7.3	1.3	6.6	1.19
Pout	6.65	1.17	5.65	1.23
Frown	6.15	1.18	5.15	1.27
Open Mouth	6.95	1.32	6.1	1.17

6.4.1 The Smile

The smile is deceptively simple in that it simply involves pulling the corners of the mouth up and back. In facial animation research it is often presented as an example of an animation method and usually shows the problems with the technique as humans are very sensitive to false smiles.

The mean value for realism is 7.3, which is very high considering the scale volunteers were asked to use. Smiles often seem fake as they do not move enough of the face, the skin right up to the eyes should move in a smile and it appears that the inter-connected muscles are moving the skin correctly.

The similarity rating of 6.6 again is reasonably high. From comments it seems that the main thing that differed between the two was the deep lines which form around the corners of the mouth during a smile. In the simulation, the skin does depress where the muscle contracts and bulge out further up, but the change between depression and bulge is smooth, whereas in the real subject there is a definite line. Allowing the skin to crease is an area for future work.

6.4.2 The Pout

The pout again had a good rating in realism, showing that the sphincter muscle model is working well. The comments showed that viwers felt that perhaps the corners of the mouth went inwards a little too far and not forwards enough. This could be resolved by increasing the stiffness in the skin so that it does not compress so much. The Orbicularis Oris is expanding in thickness as it contracts in width and height and this should keep the skin in the correct place but clearly there is room for some tuning over the way the skin connects to the Orbicularis Oris and in how it reacts to compression.

The similarity rating of 5.65 is lower than expected, but still a reasonable score indicating that users felt it was reasonably accurate. From the comments, it became clear that a lot of people though that the scanned head had it's lips parted in a funnel shape as if in an extended whistling position. This was not actually the case, but was a problem in the 3D capture process where the cameras could not get an exact match on the darker area of the lips so it looks like an opening. Due to this, it would appear that the similarity score would have been higher had the 3D scan been of better quality. Looking at the simulation and the scan, the simulated lips could be pushed further out and pulled a little further inwards.

This could be accomplished by tuning the sphincter muscle contractions so that they pull further in upon maximum contraction.

6.4.3 The Frown

The frown has the lowest score in the realism test at 6.15. The user comments indicate that this is particularly because of head B where the eyebrows do not move enough. Eyebrows vary in height from person to person and upon examination of head B, it was found that the eyebrows lie further away from the muscles used in the frown test than they do on head A. Due to this, when the Corrugator Supercilii muscles contract they do not pull the eyebrows on head B very much. Muscle position is not fixed between people, just as faces appear different, the underlying muscles are in slightly different positions. It could be that in the real head, the muscles of the person on which head B is based are positioned slightly lower than those of the person owning head A. It is possibly to easily adjust muscle positions by moving the whole muscle in the muscle builder, or in Maya, and possibly that was needed here.

The similarity rating of 5.15 is the lowest score in the tests. Again, the comments partly indicate the insufficient eyebrow movement in head B. The comments also mention that the skin in the scanned head furrows as

it is tightly compressed into a frown. The simulated skin tends to bulge and compress, but it does so smoothly and so wrinkles do not form with such definition as they do in real life. This lack of skin creasing is the same as was raised in the comments about the smile. This is possibly something that needs addressed in the skin model.

6.4.4 Open Mouth

The mouth opening test scored a high 6.95 in realism, indicating that it looked quite real to the testers. There were very few comments on this test, the only real point that was raised is that perhaps the top lip should curve and move down more as the bottom half of the face moves down. Fixing this is simply a case of tuning the tension between muscle and skin to allow the skin of the top lip to be moved more as the neighbouring skin moves and not be so rigidly held by the muscles of the top lip remaining stationary.

A similarity score of 6.1 for the mouth showed that the simulation is quite close to the movement of the scanned model. Tester comments indicate that the shape of the mouth is not quite the same as the scanned model. This comment was mostly aimed at head B, which is obviously going to be true as the two heads are quite different. There were no

significant comments about the reason head A and the simulated model were different, testers felt they could not quite say why they were not the same. This seems to indicate that the two animations were very similar.

6.4.5 General Comments

The results from the user testing show that the simulations look quite realistic and look similar to the scanned model. The only big comment that arose was that the skin does not wrinkle like real skin. This appears to detract from the realism of the animation. Overall, the testing has given very positive feedback to the system and rated it as quite realistic when compared to a real person.

Although the results of the user tests were positive, it is important to note that these tests were limited in what they can prove. The limitations were explained throughout the chapter and from the feedback and the similar results for most subjects it seems that the results can be believed. In the interests of completeness however it is worth noting what could be done given more time and resources.

Firstly, the number of testers could be increased. Twenty subjects were chosen to nicely represent different types of viewer with different backgrounds. By increasing the number of testers a larger amount of

useful feedback would definitely be produced and possible important differences between the test groups would emerge. For example, do the animators with their very precise and exacting standards really represent the vast majority of the population? Or would the layman's opinion be that very different aspects of the model are important to convince him of its realism?

It would have been enlightening to do some form of example with the state of the art in facial animation. Directly comparing this system to that of recent researchers would have provided valuable insight into how this model weighs up against the state of the art. This was not possible as was explained earlier but given enough time and some form of reciprocal sharing of knowledge it could be possible to arrange. If it was not possible to compare systems directly, some future researcher could compare their system to this one by doing the same tests as were performed here. This would obviously be limited and their results very subjective but it would be a good comparison to provide a rough benchmark. What is really needed is some form of well defined general test for facial animation, testing in facial animation research is almost universally subjective and not comparative between systems.

Whilst the four poses were chosen to represent a wide range of motions, increasing this number would have improved the validity of the

results and shown the model in a large range of positions. This was not possible because of limits on time with the dynamic face capture device but would be a good further test.

Doing a full comparison of the movement of the skin surface during the animations may provide useful data to see how close to the scanned model the movements are. This could be done on a small scale by simply tracking landmark points on the two models and comparing their movements over time. This is currently not possible with the dynamic capture device as it stands due to the captured surface having noise in its movements. With a more advanced capture device this could be possible however.

In spite of the limitations of the user testing, the results were promising and show the system does produce animation which appears realistic. The testing could be extended to find specific issues and this would be very useful before undertaking further work.

As it was not possible to compare the model developed here against another model, it is worth briefly comparing this system against the model which is closest to it. The research of Kähler et al. [KHS01], [KHYS02]. [KHS03], described in the earlier literature review led to a PhD for Kolja Kähler. His PhD has similarities to this work and these will be covered now, highlighting the similarities and differences.

The skin model of Kähler is really a single layer model. Springs connect the polygonal surface mesh directly onto either bone or muscle. The model developed here uses a multi-layer skin model where each polygonal triangular element sits atop two layers which simulate the layers of skin with different stiffness and responses to stress. Kähler uses a bi-phasic spring stiffness model similar to that of Lee et al. [LTW95], where the springs take one of two possible stiffness values depending on their current length. In comparison spring stiffness is modelled here using a non-linear function which varies the stiffness as the skin compresses and expands, giving a smoother and more realistically mimicking the changes in skin tension. Kähler's skin model is fast to simulate because of its simplicity, but the skin model shown here is not that much slower and with some development on parallelisation it would almost certainly be a real time model.

Kähler's muscle model consists of a piecewise linear polygon defining the control structure of the muscle. This is similar to the central muscle fibres developed here except interpolating splines are used to model the muscle fibre giving a smoother contraction path for the muscle. The splines also help keep things smooth when one muscle pulls a neighbouring muscle. In his model, the muscle surface is created using either elipsoids or boxes scaled to fit each section of the linear polygon. The muscles

described here allow any polygonal shape to be used as the muscle surface so that very realistic muscle shapes can be used. These muscles can also have multiple control lines intersecting to create multi-headed muscles. Upon muscle contraction, Kähler's muscles expand upon contraction like real muscles but the amount of contraction is defined by a formula and a parameter defining the amount of contraction. This formula is unrelated to real muscles. In this model, muscle volume is kept constant as muscles contract, this provides a good approximation to the amount of bulge required to expand a muscle as it contracts.

To transform muscle and skull to another mesh, Kähler uses thin plate splines. These are applied directly to the skin mesh. The user selects a set of landmarks and a low resolution mesh is constructed from these landmarks and the landmarks moved to lie in their respective positions on the mew mesh. The low-resolution mesh is then subdivided and each new vertex's position is found on the target mesh and the subdivision process is repeated. Around 60 landmarks need to be initially selected by the user and the whole process takes around 30 minutes. The method developed in this work uses a similar low-resolution mesh as a starting point, however only 30 landmarks are required to be positioned. A chain of meshes of decreasing complexity from the full resolution skull down to the low resolution mesh is then automatically created. The entire skin,

skull and muscle system is then transformed automatically using Surface Oriented Free-Form deformation. This process produces good results but like Kähler's model it can require manual intervention.

Kähler's model and the one described here are trying to solve the same problem and in doing so they have several similarities. The approaches taken are quite different in the implementation details however, and this leads to two entirely different systems. His is very fast and runs easily in real time, the one developed here is more accurate in its simulation of the face but is slower to run. The speed of execution is currently around 5 to 10 frames per second on a 2.6GHz Pentium Xeon processor. There has been no real optimisation of the code to date, and it is entirely possible that large increases in speed will be possible when the system is examined for bottlenecks. The largest of which is most likely the possible parallelisation of the numerical simulations.

7. Conclusions and Future Work

7.1 Conclusions

This thesis has detailed a new muscle and skin model which allows realistic facial animation based on manipulation of physical parameters and presented a method for easily moving the muscles and bone from one head mesh to another.

The muscle model easily handles complex, realistic muscle shapes which preserve their volume during contraction or expansion. The muscles of mastication are fully modelled and manipulate a rigid body mandible which behaves in a physically correct manner, reacting to forces applied from the connecting muscles.

The skin model developed here is automatically pulled and pushed by the muscles and mandible and again behaves in an anatomically correct manner. Its multi-layer construction mimics real skin and the different spring stiffness models allow the multiple layers to emulate the way real skin moves.

During the testing in chapter 6, these models were examined by twenty users with varying degrees of computer graphics knowledge and

skill. The results from these tests were very positive and show that the system works very well.

The two main points that came out of the testing were the need for the skin to wrinkle correctly and that muscles are not in exactly the same position on all people. Adding full skin wrinkling to the model would improve the realism even further. Some sort of test which moves all the muscles and checks that the skin moves as expected should perhaps be developed to check the position of muscle and bones after they have been warped onto a new skin mesh.

This system is a useful tool for facial animation. By first creating muscles and skull on a standard mesh, the expression set for this mesh can be quickly built up by manipulating the muscles of expression and mastication. These expressions can then be quickly transferred to any other model using the Surface Oriented FFD chain of models by selecting a small set of landmarks on the new mesh.

Using the dynamic capture device and conforming the generic mesh which has the muscle and skull fitted, a person's facial expressions can be very accurately mimicked. After the model has been tuned to a subject from a dynamic capture, the system can emulate that person in an accurate anatomically correct manner. An animator could then use the model to aid in lip-sync and easily adding expressions to enhance realism.

Overall, the system seems to work well. User testing showed that the results appear realistic and seem to accurately simulate a real person. Animation is easy to produce by manipulating muscles, and the ability to set up an animation on one model and easily transfer this to another model could be a great speed asset to animators.

7.2 Further Work

The techniques developed in this work produce high quality facial animation and this system could be the starting point for several points of research.

As was mentioned in the results chapter, the skin model does not wrinkle as much as real flesh, and this appeared to reduce the realism of the animations somewhat. By including wrinkles, the system could be increase its realism even more.

In the mastication model, the muscles of mastication could have increased realism by calculating the actual forces generated by the muscles as they contract and applying these directly to the mandible. The current method of using springs works well but by using direct forces, the movements of the jaw could be made faster and more responsive than by using springs.

A major area of future development is in the use of the dynamic capture device. By automatically analysing 3D captured data in terms of muscle movements, the synthetic muscle movements could be automatically generated which would give a virtual clone of the captured person. If the muscle movements could be discovered directly from the scans then the animation possibilities are very great. As the muscles involved in various expressions and sounds are known, it could be possible to train the system to match a captured person. For example, if a person is captured in 3D smiling, and the muscles for a smile are known, then possibly the system could train its muscle movements to match the captured subject. This would lead to extremely accurate simulations of people and give animators a huge helping hand in creating realistic animation.

The system works well for the expression tests that were performed, an interesting avenue of work would be to start looking at using the muscles to create the shapes required for speech. This could lead on to creating a very realistic talking head. Again this could involve the dynamic capture device, and mimicking of captured people speaking or it could involve work with a speech synthesiser.

7.3 Summary

This work has produced a muscle and skin model that were well received in testing and produce realistic results. The model is very easy to set up, allowing muscles of any shape or size to be created. These muscles preserve volume during contraction and the developed software automatically calculates the connections between muscles so that as one contracts, its neighbours are smoothly moved too. This muscle model is covered by a multi-layer skin model which mimics real skin well. The developed software system automatically creates the connections between the skin and the underlying muscles so that as the muscles are contracted, their movements translate into smooth skin surface changes. When muscles pull the skin, it stretches or compresses a small amount, but like real skin it rapidly resists compression and extension through varying spring stiffness.

When a muscle and skull setup has been carefully tuned to produce good quality animation it can be easily moved to another head mesh using a small selection of landmarks. The developed software allows a user to select a small number of landmarks on a new mesh and the program automatically warps the muscles and skull to fit the new head. Due to the automatic creation of the multi-layer skin and skin-muscle

connections, once a muscle and skull set have been warped to a new head, animation can begin almost straight away.

This work has shown a new, successful set of muscle, skin and transferrable facial animation concepts. User testing showed that the system works well and it is currently being evaluated for possible use at Europe's biggest visual effects company.

Appendix

A. Conformation of Scanned Models

To turn a scanned model into one useful for animation, a mesh which is suitable for animation must be conformed onto the surface of the scanned model. Although only a single model was required to be conformed for the testing work in this thesis, the method developed and used is capable of conforming a model to an entire scanned sequence and the whole technique is briefly described here.

A.1 Background

A.1.1 The Dynamic Capture Device

In the Computer Vision and Graphics group at the University of Glasgow, a dynamic 3D capture device is being developed. Using 24 cameras it can capture images at 25 frames per second and then from these construct 3D models of the subject. Using this device to capture a subject speaking, 3D models of the face surface can be captured giving the location of every point on the face (up to the resolution of the capture device) twenty-five times a second.

The system works using photogrammetry, finding the same point in multiple images and working out its position in 3D space. To help the software find the same point in each image, a random speckle pattern is projected on to the subject. This pattern should be washed out in the texture images by the very bright strobe lights used to illuminate the subject for texture capture. Figure A.1 shows a model captured and built from a longer sequence using the dynamic capture device.

Fig. A.1: A model from a dynamic capture sequence, showing the texture and the mesh.

As can be seen in the image the mesh is constructed in an arbitrary fashion with dense points in areas of high curvature and less in smoother areas. There is no correspondence between the meshes of individual models for each frame and thus no simple way to track a surface point from one frame to the next. The meshes produced are also completely unsuitable for facial animation. For facial animation, the edges should form curves flowing along areas where the face naturally creases which

will allow the mesh to curve naturally when it is used in animation.

The textures covering each captured model can vary dramatically between frames. The differences between the textures are due to a number of factors. During the capture process, the strobe light that illuminates the object must be perfectly timed to match the instant the camera captures the image. If the timing is slightly off then a different amount of light will reach the surface causing the texture to appear brighter or darker. The miss-timing of the flash also causes the projected speckle texture to be visible. The strobe should nearly wash it out in the texture images with its brightness, but again depending on the timing, the speckle pattern can be visible.

Having the entire surface available every frame allows every point on the model to be examined and tracked through time. However, as the meshes produced for every frame are completely separate, a correspondence must be found between the same points on every model, such as the corners of the mouth in every mesh. This will allow the motion of that point to be tracked throughout the entire sequence. Using these landmarks, an entire mesh can be fitted to every model in the sequence. This mesh will then deform as the original models did but retain the same connectivity and vertices throughout the sequence allowing it to be analysed and the animation to be extracted easily.

Marking a set of feature points on the model could be performed manually and for a single frame this wouldn't be a problem. However for a short sequence of ten seconds, 250 models are generated by the capture device and finding feature points on that many models would be a very time consuming and monotonous job. For this reason, an automatic method was developed.

A.2 Finding and tracking a set of feature points

A.2.1 Noise

The images produced by the dynamic scanner are 640x480 pixels in size. This is a relatively low resolution for a digital camera. For example, in a static capture setup in the same lab, the images used are 4096x4096 pixels in size. Thats 54 times as many pixels in the images.

The actual three-dimensional models themselves are smoothed significantly to remove noise which appears in the model building process. It would be very unreliable trying to detect surface features on these smoothed models as there are no distinguishing features for an algorithm to latch on to.

Although the texture images for the captured models are noisy, features can still be made out unlike on the smoothed 3D surface where it is very hard to pin point features with accuracy. For this reason, it was decided to use the texture data as the primary tool in finding model features and then use the positions found to calculate the 3d surface points on the model. The two texture images which make up the head surface are mapped onto the model so that each image is used where the camera which captured it had the best view of that surface point. This means that not all of each texture is used and in some cases very little of some textures is used. Figure A.2 shows how a set of textures map onto the model.

As only parts of each texture are used and there is no way to predict which parts, feature detection cannot be performed simply by using the texture images and then finding where each point maps onto the 3D surface, as that point may have no mapping onto the surface if it is not used in the final textured mesh. To overcome this and allow the features to be found in a 2D texture, a combined image which has all the facial features in it must be constructed.

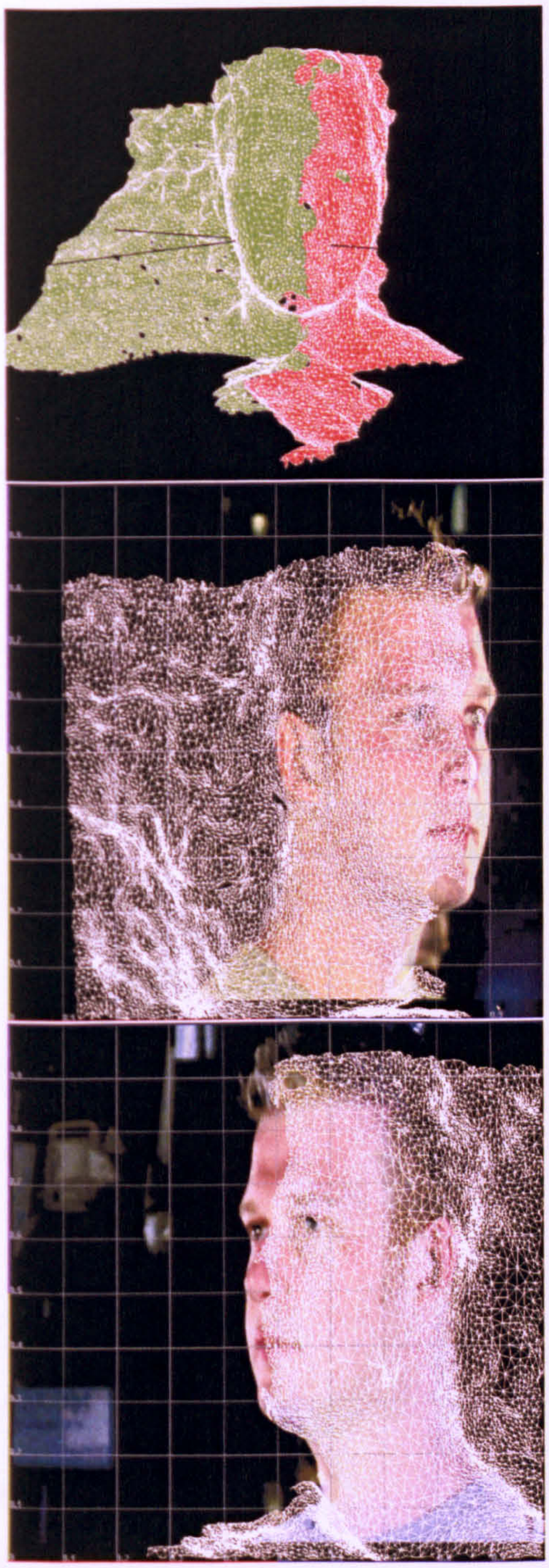


Fig. A.2: Some dynamic textures and their mappings on a constructed model

A.2.2 Creating a 2D Image from the 3D Model

To create a 2D image from the 3D textured model requires mapping the points of the model from 3D into 2D and then the texture can be easily extracted by rendering the model in its new 2D space. There are numerous projection methods to turn a 3D object into a 2D image. The full human head is most suited to cube or spherical projections of which there are several types. For this work however, only the face is of real interest as the top, back and rear sides of the head do not change during speech or expression and due to accuracies in capturing such things, these areas are covered in hair and as such change so rapidly from frame to frame that they are of little use in retrieving an accurate surface representation. To project the face into 2D, cube or spherical projection could still be used, but cylindrical projection gives little distortion if the radius is carefully selected as the face is closer to a cylinder than any other simple geometry.

The rough direction the subject is facing during the scan is known from the capture session or by examining any of the models. By aligning this direction roughly with the z-axis and interactively moving a bounding box to contain only the face and features required for projection the models can have a large portion of them culled immediately. After this

culling the object is ready for projection.

Projection onto the cylinder is a simple process, the cylinder radius is first calculated from the bounding box of the points to be projected and then the points are all mapped onto the cylinders surface. The process of turning a point into cylindrical coordinates and then onto the $x-y$ plane is simply performed using:

$$\begin{aligned} \theta &= \arctan\frac{x}{z} \\ S_x &= r\theta \\ s_y &= y \end{aligned} \tag{A.1}$$

where r is the radius of the cylinder and x, y and z are the coordinates of the point in three dimensional space.

Creating a 2D image of the model is then accomplished by displaying the triangles of the mesh using the new 2D coordinates and saving out the resulting image.

A.2.3 Finding the Feature Points

Level Sets

In facial feature tracking, active contours (snakes) [KWT88] are often used to find the lip outline. Snakes however suffer from the problem

that they can be very susceptible to noise which, as has been covered, these image maps have an abundance of. Level sets are less common than snakes but have enjoyed popularity in the medical research community for helping to extract data from noisy images [TYW+03], [AJL04], [ASML04].

A 2D image can be defined as a $S_x \times S_y$ array of integer values. The locations of pixels are (x, y) where $0 \leq x \leq S_x$ and $0 \leq y \leq S_y$. For the domain square with corners (x_0, y_0), $(x_0 + 1, y_0)$, $(x_0, y_0 + 1)$ and $(x_0 + 1, y_0 + 1)$, then let $F(x, y)$ be a continuous function representing the image on the square. For a value L in the range of F, the *level set* of F for this value L is the set of all points that satisfy $F(x, y) = L$.

The level set method used here is from by David Eberly [Ebe03]. The set of edges produced by the level set image search is tailored to find the mouth. The best value L to produce the lip contour is usually in the same region for most people but can be user defined by selecting multiple points on the lips and an average value can be found which is used for the level set algorithm. The level set code produces a set of edges which wrap round the lips and eyes in the texture. This provides a set of which can be used to align a generic mesh with the scanned models.

A.3 Conforming The Generic Mesh

After the landmarks are selected, the generic mesh must be warped to fit the scanned model of every frame. To do this, the generic mesh is first scaled to roughly match the scanned model and then projected onto a cylinder and flattened out in the same manner as the scanned models were. The generic mesh is scaled so that the landmarks at the edge of either eye and the height of the models are roughly the same. The generic mesh is then projected onto a cylinder of the same size as that calculated and used for the scanned models.

The generic mesh can then be overlayed on each scanned model and both will have the same proportions. The vertices of the generic mesh corresponding to the landmark positions in the scanned model are then moved to lie in the correct position on top of the scanned model. This movement of some of the vertices will cause the mesh to be twisted upon itself as other vertices are now sitting on the wrong side of the moved vertices. To reposition the vertices in a layout roughly as they were before, a relaxation scheme is used.

A.3.1 Relaxation

The relaxation method is similar to that of Lee et al. [LTW95]. The main difference is that they used springs to relax the mesh, it was found that springs can be too erratic in their movements and can tend to settle into a relaxed state in a configuration which is not evenly spread across the mesh. The mesh used for demonstration in their paper was also of low resolution compared to the generic mesh used here which adds greatly to the problem of twisted edges.

To improve the relaxation of the mesh, a particle-constraint system is used where the vertices are represented as particles and the edges are constraints which try to keep the particles a set distance apart. The problem of finding the position of all the points with all the constraints satisfied effectively produces a system of equations that need to be satisfied. Solving such a system directly may produce a result that satisfies every length constraint but is not a well spread out mesh. Due to this, a semi-interactive iterative solution was employed which gradually resolves the constraints until the user is satisfied with the results. During the relaxation process, the user can move particle positions to a more satisfactory position if it seems that the system is converging to a solution where the particles are in a suboptimal position. The interactive process

is usually not needed but is available incase the process fails to deliver a result. The only time this is really likely to happen is if the generic mesh and scanned head are of very different shapes and the system needs a small push to get in the right direction. For every frame apart from the first the conformed mesh of the previous frame is used as entry to this stage and so user interaction is only ever needed on the first frame.

To satisfy the constraint c connecting particles p_i and p_j with rest length r, the following is used:

$$\begin{aligned} \overrightarrow{\delta} &= p_i - p_j \\ \tau &= \frac{|\overrightarrow{\delta}| - r}{\delta} \\ p_i &= p_i + 0.5\overrightarrow{\delta}\tau \\ p_j &= p_j - 0.5\overrightarrow{\delta}\tau \end{aligned} \tag{A.2}$$

This will satisfy a single constraint, the same process is carried out for every constraint in the mesh. Obviously changing the length of one constraint affects all constraints sharing one of the end particles. So, to try and satisfy those constraints, the entire mesh is scanned several times and the above constraint satisfaction applied to all constraints which are not at their rest length each time. This process will converge to a solution or close to in a small number of steps and rapidly spreads

the mesh smoothly across the area.

A.3.2 From 2D to 3D

To turn the flat 2D conformed generic mesh into a 3D model, each vertex's position is first defined in terms of the scanned model. To do this, each vertex of the generic model is first defined in the barycentric coordinates of every triangle of the scanned mesh. By definition the three barycentric coordinates α, β and γ should add up to one and all be positive. If any are negative then the point lies outside the triangle. To find the triangle that the generic point lies in, a brute force approach would be to try defining the point in every triangle successively, as soon as a triangle is found where all three barycentric coordinates are positive the triangle has been found. To find the barycentric coordinates of a point in a triangle is a simple calculation, however finding the position of every generic vertex by scanning thousands of triangles per frame adds up to a significant amount of time. To speed this up, a quadtree is constructed with the faces of the scanned mesh and this is used to rapidly find the triangles that the generic mesh lies in.

Once the barycentric coordinates of each generic mesh vertex are found, the position of that point in 3D space is simply a matter of using

those barycentric values to work out the relative position to the same three scanned mesh points in 3D. This places the generic mesh vertex onto the surface of the scanned mesh triangle making up that point.

The texture coordinates are found by applying the barycentric coefficients to the texture coordinates of each of the vertices of the scanned mesh barycentric triangle. This produces the generic mesh conformed to fit on to the scanned model, so it will have the same connectivity and number of vertices but the shape of the scanned model.

A.4 Noise Removal

After the generic mesh has been conformed to each model in the sequence, interpolating the vertices between their positions for each frame produces animation where the generic mesh should move in the same manner as the scanned models. This works but the problem of noise still impairs the quality. The surface captured by the scanner varies between frames due to the error rate in the capture process. Even a still skin surface will appear to move slightly each frame in the captured data. When played back at 25fps this translates into high frequency noise in the surface motion. This problem is completely uncontrollable in areas of the face where the scanner couldn't get a precise match on the surface such as

hair. In such areas, the surface changes wildly from one model to the next and the data cannot be trusted to give accurate results. If the models are looked at separately, the hair looks plausible in every frame, but because it varies between frames it seems to jump around. To fix this, conformation masks were developed.

A.4.1 Conformation Masks

A conformation mask is a weighting for every vertex in the model that defines how much it is affected from frame to frame. Each vertex is assigned a value between zero and one which is its conformation mask value. This value is then used to define how much that point should move during animation. Assuming that the head is not moving, then the hair should not move at all each frame and would be assigned a value of zero meaning it should never move from its position in the first frame. The mouth on the other hand can move greatly from frame to frame as a person speaks and as such is assigned a value of one meaning take the full value of the scanned mesh in every frame. Areas between the mouth and the hair have varying degrees of movement and the conformation mask gradually blends from full movement to none around the face.

The mask is created once and can be used as is, or the user can

manipulate it if they find it does not fit the target head sequence. Manipulating it is like painting on the mesh, the user sets the value of mask they wish to use and drags the mouse over the vertices they want to have that value. This can produce abrupt changes in mask strength, so after the user is finished a smoothing process is applied to all vertices. This is a simply an averaging over the values of each vertex and its neighbours.

A.4.2 Noise Reduction

The motion of the vertices in the conformed mesh is produced by simply moving them from their current position to the conformed position of the next model 25 times a second. The noise in the conformation can cause this to appear as if the vertices jump from place to place. To overcome this a Gaussian filter is applied to the position data for each vertex which smooths out the path.

A.5 Results and Conclusions

This chapter has presented a method of conforming a generic mesh to a series of scanned models. This conformation produces a correspondence between the models so that the movement of any point on the surface can be tracked throughout the entire sequence.

The noise reduction techniques employed change a very noisy set of conformed models into a set of quite smoothly flowing models which closely follow the original subjects movements.

Bibliography

[AH02] I. Albrecht and J. Haber. Speech synchronization for physics-based facial animation. In *Proceedings WSCG 2002*, pages 9–16, 2002.

[AHK+02] I. Albrecht, J. Haber, K. Khler, M. Schrder, and H.-P. Seidel. "may i talk to you? :-)" – facial animation from text. In *Proceedings Pacific Graphics 2002*, pages 77–86, 2002.

[AJL04] E.D. Angelini, Y. Jin, and A.F. Laine. State-of-the-art of levelset methods in segmentation and registration of medical imaging modalities. *Handbook of Medical Image Analysis: Advanced Segmenation and Registration Models*, 2004.

[ASML04] E.D. Angelini, T. Song, B.D. Mensh, and A.F. Laine. Multi-phase three-dimensional level set segmentation of brain mri. In *Medical Image Computing and Computer-Assisted Intervention*, 2004.

[BCS97] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360. ACM Press/Addison-Wesley Publishing Co., 1997.

[BKMTK00] L. Boissieux, G. Kiss, N. Magnenat-Thalmann, and P. Kalra. *Simulation of Skin Aging and Wrinkles with Cosmetics Insigh.* 2000.

[BKN02] Y. Bando, T. Kuratate, and T. Nishita. *A Simple Method for Modeling Wrinkles on Human Skin.* 2002.

[BL85] P. Bergeron and P. Lachapelle. In *Controlling facial expressions and body movements*, volume 2, pages 61–79. ACM, 1985.

[BN92] Thaddeus Bier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92 proceedings*, pages 35–42. ACM Press, 1992.

[BOP98a] S. Basu, N. Oliver, and A. Pentland. 3d lip shapes from video: A combined physical-statistical model. *Speech Communication*, pages 131–148, 1998.

[BOP98b] S. Basu, N. Oliver, and A. Pentland. 3d modeling and tracking of human lips. In *IEEE International Conf. on Computer Vision*, 1998.

[BOP98c] S. Basu, N. Oliver, and A. Pentland. Coding human lip motions with a learned 3d model. In *International Workshop on Very Low Bitrate Video Coding*, 1998.

[Bou02] David M. Bourg. *Physics for Game Developers.* O'Reilly, 2002.

[CG98] Eric Cosatto and Hans Peter Graf. Sample-based synthesis of photo-realistic talking heads. In *CA*, pages 103–110, 1998.

[CHP89] J. Chadwick, D. Haumann, and R. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):234–243, 1989.

[CM93] M. Cohen and D. Massaro. Modeling coarticulation in synthetic visual speech. *In D. Thalmann N. Magnenat-Thalmann, editor, Computer Animation*, 1993.

[Coo98] L. Cooper. *Physically Based Modelling of Human Limbs, PhD Thesis.* University of Sheffield, 1998.

[Coq90] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196. ACM Press, 1990.

[CR74] E. Catmull and R. Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, 1974.

[Deb96] Paul Debevec. *Modeling and Rendering Architecture from Photographs.* PhD thesis, University of California at Berkely, 1996.

[Dip91] S. Dipaola. Extending the range of facial types. *Journal of Visualization and Computer Animation*, 2(4):129–131, 1991.

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 98*, pages 85–94, 1998.

[DLG90] Nira Dyn, David Levine, and John A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.*, 9(2):160–169, 1990.

[DMS98] Douglas DeCarlo, Dimitris Metaxas, and Matthew Stone.

An anthropometric face model using variational techniques. In *Proceedings of SIGGRAPH 98*, pages 67–74, 1998.

[DWD+83] Parvati Dev, Sally Wood, James P. Duncan, David N. White, Stuart, and W. Young. An interactive graphics system for planning reconstructive surgery. In *National Computer Graphics Association*, pages 130–135, 1983.

[Ebe03] David H. Eberly. *Game Physics.* Morgan Kaufmann, 2003.

[EF78] P. Ekman and W. V. Friesen. *Manual for the Facial Action Coding System.* Consulting Psychologists Press Inc., 1978.

[EF84] Paul Ekman and Wallace V. Friesen. *Unmasking the Face.* Consulting Psychologists Press, 1984.

[EFH02] P. Ekman, W. V. Friesen, and J. C. Hager. Manual for the facial action coding system, 2002.

[EGP02] Tony Ezzat, Gadi Geiger, and Tomaso Poggio. Trainable videorealistic speech animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 388–398. ACM Press, 2002.

[EM01] J. D. Edge and S. Maddock. Expressive visual speech using geometric muscle functions. In *Proceedings of the 19th Eurographics UK Chapter Annual Conference*, 2001.

[EP97] Tony Ezzat and Tomaso Poggio. Videorealistic talking faces: A morphing approach. In *Proceedings of the Audio-visual Speech Processing Workshop, Rhodes, Greece*, 1997.

[EP98] Tony Ezzat and Tomaso Poggio. Miketalk: A talking facial display based on morphing visemes. In *Computer Animation '98*, pages 96–102. IEEE Computer Society, 1998.

[EP00] Tony Ezzat and Tomaso Poggio. Visual speech synthesis by morphing visemes. *Laryngoscope*, 38:45–57, 2000.

[GBF03] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[GCE00] H. Graf, E. Casotto, and T. Ezzat. Face analysis for synthesis of photorealistic talking heads. In *Proceedings of the International Conferance on Autom. Face and Gesture Recog*, pages 189–194, 2000.

[GH01] Michael Garland and Paul Heckbert. Simplifying surfaces

with color and texture using quadric error metrics. In *IEEE Visualization 98*, 2001.

[GHH+96] J. Gosling, P. Harris, J. Humpherson, I. Whitmore, P. William, A. Bentley, and J. Hargreaves. *Human Anatomy.* Mosby-Wolfe, 1996.

[Gro00] Moving Picture Experts Group. *ISO 14496.* ISO, 2000.

[HFG94] Michael Hoch, Georg Fleischmann, and Bernd Girod. Modeling and animation of facial expressions based on b-splines. *The Visual Computer*, 11(2):87–95, 1994.

[HG99] Paul Heckbert and Michael Garland. Optimal triangulation and quadric-based surface simplification. In *Journal of Computational Geometry: Theory and Applications*, volume 14, pages 49–65, 1999.

[IH93] M. Yaser Iscan and Richard P. Helmer. *Forensic Analysis of the Skull: Craniofacial Analysis, Reconstruction, and Identification.* Wiley-Liss, 1993.

[Jac02] Peter Jackson. *The Lord of the Rings.* New Line Cinema, 2002.

[Jr.86a] Wayne F. Larrabee Jr. A finite element model of skin deformation: Part i - biomechanics of skin. *Laryngoscope*, pages 399–419, 1986.

[Jr.86b] Wayne F. Larrabee Jr. A finite element model of skin deformation: Part ii - experimental model of skin deformation. *Laryngoscope*, pages 399–419, 1986.

[Jr.86c] Wayne F. Larrabee Jr. A finite element model of skin deformation: Part iii - the finite element model. *Laryngoscope*, pages 399–419, 1986.

[KAA83] Arthur I. Karshmer, Daniel Allan, and Dolores Anderson. 'virtual' craniofacial surgery using interactive computer graphics: A pilot study. In *National Computer Graphics Association*, pages 125–129, 1983.

[KGC+96] Rolf M. Koch, Markus H. Gross, Friedrich R. Carls, Daniel F. von Büren, George Fankhauser, and Yoav I. H. Parish. Simulating facial surgery using finite element models. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 421–428. ACM Press, 1996.

[KGPG96] Erwin Keeve, Sabine Girod, Paula Pfeifle, and Bernd Girod. Anatomy-based facial tissue modeling using the finite element method. In *Proceedings of the 7th conference on Visualization '96*, pages 21–ff. IEEE Computer Society Press, 1996.

[KHS01] Kolja Kähler, Jrg Haber, and Hans-Peter Seidel. Geometry-based muscle modeling for facial animation. In *Proceedings Graphics Interface*, pages 37–46, 2001.

[KHS03] Kolja Kähler, Jrg Haber, and Hans-Peter Seidel. Reanimating the dead: reconstruction of expressive faces from skull data. *ACM Transactions on Graphics (TOG)*, 22(3):554–561, 2003.

[KHYS02] Kolja Kähler, Jrg Haber, Hitoshi Yamauchi, and Hans-Peter Seidel. Head shop: generating animated head models with anatomical structure. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 55–63. ACM Press, 2002.

[KMMTD92] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D.Thalmann. Simulation of facial muscle actions based

on rational free form deformations. In *Proceedings Eurographics*, pages 59–69, 1992.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *IJCV*, pages 321–331, 1988.

[Lew91] John Lewis. Automated lip-sync: Background and techniques. *The Journal of Visualization and Computer Animation*, 2(4):118–122, 1991.

[LK84] S. Lien and J. T. Kajiya. A symbolic method for calculating the integral properties of arbitrary non-convex polyhedra. *IEEE Computer Graphics and Applications*, 4(10):35–41, 1984.

[LP87] J.P. Lewis and Frederic I. Parke. Automated lip-synch and speech synthesis for character animation. In *Proceedings Human Factors in Computing Systems and Graphics Interface '87*, pages 143–147. ACM Press/Addison-Wesley Publishing Co., 1987.

[LS89] Tom Lyche and Larry L. Schumaker. *Mathematical Methods in Computer Aided Geometric Design.* Academic Press, 1989.

[LTW95] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic face modeling for animation. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 55–62. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[M̈97] Tomas Möller. A fast triangle-triangle intersection test. *Journal of Graphic Tools*, 2(2):25–30, 1997.

[MHL95] R. McMinn, R. Hutchings, and B. Logan. *Color Atlas of Head and Neck Anatomy.* Mosby-Wolfe, 1995.

[Mon80] A. Montgomery. Development of a model for generating synthetic animated lip shapes. *Journal of the Acoustical Society of America*, pages 68–78, 1980.

[Mus02] Dr. J. E. Muscolino. *The Muscular System Manual.* JEM Publications, 2002.

[Neb01] J.C. Nebel. Generation of true 3d films. In *Proc. 1st Int. Conf. on Virtual Storytelling (ICVS2001), Avignon, France*, 2001.

[NMTT88] E. Primeau N. Magnenat-Thalmann and D. Thalmann.

Abstract muscle action procedures for face animation. *The Visual Computer*, 3:290–297, 1988.

[NvdS00] Han-Wen Nienhuys and A. Frank van der Stappen. Combining finite element deformation with cutting for surgery simulations. In A. de Sousa and J.C. Torres, editors, *EuroGraphics Short Presentations*, pages 43–52, 2000.

[Osi03] Jason Osipa. *Stop Staring - Facial Modeling and Animation Done Right.* Sybex, 2003.

[PAL+97] F. Pighin, J. Auslander, D. Lischinski, David Salesin, and R. Szeliski. Realistic facial animation using image-based morphing. Technical report, University of Washington, 1997.

[Par72] Frederic I. Parke. *Computer generated animation of faces.* Master's Thesis, University of utah, 1972.

[Par74] Frederic I. Parke. *A Parametric Model for Human Faces.* PhD Thesis, University of utah, 1974.

[Par75] Frederic I. Parke. A model for human faces that allows speech synchronized animation. In *Journal of Computers and Graphics*, volume 1, pages 1–4, 1975.

[Par82] F. I. Parke. Parameterized models for facial animation. *I.E.E.E. Computer Graphics and Applications*, 2(9):61–68, 1982.

[PB81] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 245–252. ACM Press, 1981.

[PHL+98] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and David Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98, in Computer Graphics Proceedings, Annual Conference Series.* ACM Press, 1998.

[Pie89] Steve Pieper. Physically-based animation of facial tissue for surgical simulation. *SIGGRAPH 1998 Course Notes Notes 22: State of the Art in Facial Animation*, pages 79–124, 1989.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, pages 147–155, 1995.

[PW96] Frederic I. Parke and Keith Waters. *Computer Facial Animation.* A K Peters, 1996. ISBN 1-56881-014-8.

[PWWH86] Andrew Pearce, Brian M. Wyvill, Geoff Wyvill, and David Hill. Speech and expression: A computer solution to face animation. In *Proceedings of Graphics Interface '86*, pages 136–140. ACM Press/Addison-Wesley Publishing Co., 1986.

[RMKB98] M. H. Gross R. M. Koch and A. A. Bosshard. Emotion editing using finite elements. In *Computer Graphics Forum (Proceedings of Eurographics 1998)*, pages C295–C302, 1998.

[SK00] Karan Singh and Evangelos Kokkevis. Skinning characters using surface oriented free form deformations. In *Graphics Interface '00.* Canadian Information Processing Society, Canadian Human-Computer Communications Society, 2000.

[SKW+94] K.C. Scott, D. S. Kagels, S. H. Watson, H. Rem, J. R. Wright, M. Lee, and K.J. Hussey. Synthesis of speaker facial movement to match selected speech sequences. In

Proceedings of the Fifth Australian Conference on Speech Science and Technology, pages 620–625, 1994.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160. ACM Press, 1986.

[Squ01] Square. *Final Fantasy: The Spirits Within.* Comlumbia Tristar, 2001.

[Stu90] Pixar Animation Studios. *Tin Toy.* Walt Disney Pictures, 1990.

[TW90] Demetri Terzopoulos and Keith Waters. Physically-based facial modelling, analysis and animation. *Journal of Visualization and Computer Animation*, 1(2):73–80, 1990.

[TW93] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anaomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569–579, 1993.

[TYW+03] A Tsai, A Yezzi, W Wells, C Tempany, D Tucker, A Fan, and A Willsky E Grimson. A shape-based approach to

curve evolution for segmentation of medical imagery. *IEEE Transactions on Medical Imaging*, 22(2):137–154, 2003.

[UGO98] Baris Uz, Ugur Gudukbay, and Bulent Ozguc. Realistic speech animation of synthetic faces. *Computer Animation '98*, pages 111–118, 1998.

[VY92] M.-L. Viaud and H. Yahia. *Facial animation with wrinkles.* 1992.

[Wat87] Keith Waters. A muscle model for animating three-dimensional facial expression. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 17–24. ACM Press, 1987.

[Wat89] Keith Waters. Modeling 3d facial expressions. *SIGGRAPH 1998 Course Notes Notes 22: State of the Art in Facial Animation*, pages 127–152, 1989.

[Wat92] Keith Waters. A physical model of facial tissue and muscle articulation derived from computer tomography data. In *SPIE Visualization in Biomedical Computing*, volume 1808, pages 574–583, 1992.

[WB01] A. Witkin and D. Baraff. Physically based modelling. *SIGGRAPH 2001 Course notes.*, 2001.

[WKMT96] Yin Wu, Prem Kalra, and Nadia Magnenat-Thalmann. Simulation of static and dynamic wrinkles of skin. In *Computer Animation*, pages 90–97, 1996.

[WL93] Keith Waters and Thomas M. Levergood. Decface: An automatic lip-synchronization algorithm for synthetic faces. Technical report, Digital Equipment Corporation Cambridge Research Lab, 1993.

[WT90] K. Waters and D. Terzopoulos. A physical model of facial tissue and muscle articulation. In *Proceedings of the 1st conference on Visualization in Biomedical Computing*, pages 77–82, 1990.

[WT91] Keith Waters and Demetri Terzopoulos. Modelling and animating faces using scanned data. *Journal of Visualization and Computer Animation*, 2(4):123–128, 1991.

[YLW93] D. Terzopoulos Y.C. Lee and K. Waters. Constructing physics-based facial models of individuals. In *Proceedings of Graphics Interface '93*, pages 1–8, 1993.

[Zem04] Robert Zemeckis. *The Polar Express.* Warner Brothers, 2004.

[ZSS96] Denis Zorin, Peter Schröder, and Wim Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 1996: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 189–192. ACM Press, 1996.