Facial Animation and Speech Synthesis

BY
Tunde Adegbola
B.S. Computer Science, Saint Louis University, 2003

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the graduate studies program
of DigiPen Institute Of Technology
Redmond, Washington
United States of America

Summer
2008

Thesis Advisor: Xin Li
DIGIPEN INSTITUTE OF TECHNOLOGY

GRADUATE STUDY PROGRAM

DEFENSE OF THESIS

THE UNDERSIGNED VERIFY THAT THE FINAL ORAL DEFENSE OF THE MASTER OF SCIENCE THESIS OF TUNDE ADEGBOLA

HAS BEEN SUCCESSFULLY COMPLETED ON JULY 16, 2008

TITLE OF THESIS: FACIAL ANIMATION AND SPEECH SYNTHESIS

MAJOR FIELD OF STUDY: COMPUTER SCIENCE.

COMMITTEE:

Xin Li, Chair	Gary Herron
Suzanne Kauffman	Dmitri Volper

APPROVED :

Graduate Program Director date	Dean of Science Division date
Chair of Computer Science date	President date

The material presented within this document does not necessarily reflect the opinion of the Committee, the Graduate Study Program, or DigiPen Institute Of Technology.

INSTITUTE OF DIGIPEN INSTITUTE OF TECHNOLOGY

PROGRAM OF MASTER'S DEGREE

THESIS APPROVAL

DATE: JULY 16, 2008

BASED ON THE CANDIDATE'S SUCCESSFUL ORAL DEFENSE, IT IS RECOMMENDED THAT THE THESIS PREPARED BY

TUNDE ADEGBOLA

ENTITLED

FACIAL ANIMATION AND SPEECH SYNTHESIS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE FROM THE PROGRAM OF MASTER'S DEGREE AT DIGIPEN INSTITUTE OF TECHNOLOGY.

Dr. Xin Li
Thesis Advisory Committee Chair

Dr. Xin Li
Director of Graduate Study Program

Dr. Xin Li
Dean of Science Division

The material presented within this document does not necessarily reflect the opinion of the Committee, the Graduate Study Program, or DigiPen Institute Of Technology.

Table of Contents

Abstract

The purpose of this research is to investigate and determine effective ways to simulate facial animation for a specific person. It should introduce the reader to the world of facial animation, highlighting its history and origins. Existing techniques that have been devised for computer based facial animation will be explored, identifying the positive and negative aspects of each in the process. New physically based approaches will also be examined and compared to previous techniques to highlight improvements that have been made. Applicability of these relatively new algorithms to video games and similar entertainment outlets will also be examined.

1 Introduction

Facial animation is a heavily researched area of computer science and has been so for decades. Even before then, it was a major part of feature film production for companies such as Disney. This dates as far back as 1928.

Animation of human faces is a difficult task, above all other forms of animation. When one factors in the range of expressions and emotions of individuals, it becomes evident how vast an area of research facial animation is.

Facial animation consists of two parts: animation of emotions, and speech animation, both which complement each other. How so? Well, speech animation alone would seem robotic without some form of accompanying fine tuned by adding emotional expressions.

2 A Little History

This section not only offers information on the history of facial animation, but it also highlights a non muscle-based implementation approach with a focus on speech animation

2.1 Lip Sync: The Early Days

Lip syncing has been explored since the release of Disney's Steamboat Willie in 1928. This was the first time sound and animation had been joined. Shortly thereafter, the first techniques were developed in the early 30's for timing facial expressions with accompanying dialogue.

2.1.1 Development of Animation Shortcuts

Disney played a major part. Their artists' concern was on how the mouth looks while making the sounds. They later on discovered that several sounds can be made with the same mouth position.

Figure 2.1 is an example of a lip sync template. Animators relied on a chart consisting of a number of archetypal mouth positions to represent speech.

Fig. 2.1: Lip Sync template

2.2 Lip Sync: The Last Few Decades

In the early 70s, Frederik Parke presented the polygonal representation of a head with animation of eyes and mouth opening and closing [Parke 1972].

Various approaches and techniques were introduced:

1. concatenative [Lee et al. 1995]
2. parameterized [Parke 1982]
3. muscle based approach [Parke and Waters1996].

One major problem with most if not all techniques and was accomplishing accurate, realistic speech animation with minimal effort. In order to get a better grasp of the sub task at hand, one must be familiar with the concept of definition of a phoneme set and viseme creation.

2.3 What is a Phoneme?

A phoneme can be described as the smallest contrastive unit in the sound system of a language. Just as one or more syllables make up a word, one or more phonemes make up a syllable. They are the individual sounds that make up speech, the number of which vary depending on the language.

2.4 What is a Viseme?

A viseme is described as a generic facial image that can be used to describe a particular phoneme. They are sometimes referred to as visual phonemes, though one should note that they do not have a 1-to-1 viseme/phoneme relationship. This will become more evident later on in the phoneme reduction description.

2.5 Putting things together for Speech animation

In order to simulate speech animation, visemes will need to be synced with each phoneme uttered by a given character, the naive approach to such animation would be to attempt to create a separate facial position for each phoneme, but this is a wasteful approach. In order to achieve realistic lip synching, the following general steps are taken by the Authors in [1], using 3ds Max and Maxscript:

1. Viseme creation
2. Assigning visemes to morpher modifier channels
3. Animating the percentage of viseme appearance in key frames of animation.

Step 1: Viseme Creation

This consists of creating a particular number of copies of the basic head model. Every copy represents one of the visemes with facial vertices moved to achieve the look of the face during pronunciation of the corresponding phoneme. The number of these copies(visemes) created and stored depends on depends on reduction level.

Step 2: Assigning visemes to Morpher modifier channels

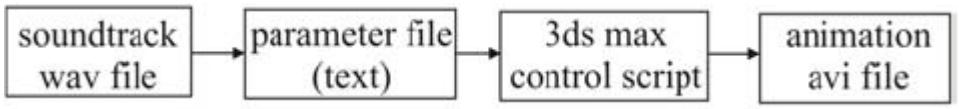
A Morpher modifier is a data structure designed to handle particular number of channels and their parameters. Basically, the idea is to gradually transform from one copy of the head model into another. Recall from step one that each copy stores a unique viseme. After visemes are created, they are then assigned to morpher modifier channels as morph targets. These channels allow for animators to interpolate between

the different visemes, via repositioning of vertices, taking note of their position in both copies of the original object. For this to work, these copies must possess the same number of vertices.

Step 3: Animation, Key frames etc

At this stage, the animator determines key frames for every phoneme. The Animation is created by assigning corresponding percentages to adequate visemes in Morpher modifier channels in key frames of time. Software calculates interpolation between key frames and generates the final animation that could be rendered and recorded to the tape and later edited/formatted into a final movie sequence.

Since step 3 can be a very tedious step, Rizvic and Avdagic, created a script to automatically generate these key frames. The algorithm to achieve this will not be discussed in great detail, but it is still important to highlight the algorithm used:



Phases of the algorithm:

- initialization, file opening
- viseme check
- key frame creation in corresponding Morpher modifier channel
- end of file check
- final animation rendering

MaxScript, being native to the 3ds environment was used. For information on how the scripting was performed, refer to [1] for more information on how this was performed.

2.6 Phoneme reduction

In the early days of animation the basic principle was simplicity, as attempts at accuracy seemed too unnatural [Madsen 1969]. Because of this, animators decided to create visual shorthand that passes unchallenged by the viewer, similar to dropping frames per second in a game, yet maintaining fluidity at the same time.

Reduction to a small set of visemes was feasible in the past was more feasible since characters were not as realistic, but due to increased realism of characters in games and feature films, When level of detail becomes a factor. Rizvic and Avdagic propose the use of phoneme reduction to facilitate all levels of detail.

2.6.1 Full Animation

According to Flemming and Dobbs [3] there are 40 phonemes in the American English language. They also describe a unit of speech being considered a phoneme if replacing it in a word results in a change of meaning. Based on this information, they developed the following list of phonemes in the table below:

Articulation	Phoneme	Example	Visual Phoneme	Articulation	Phoneme	Example	Visual Phoneme
Vowels(unitary)	IY	B**ea**t	1	Nasals	M	**M**aim	9
	IH	B**i**t	2		N	**N**one	10
	EY	B**ay**	2		AN	B**ang**	2
	EH	B**e**t	2	Fricatives	F	**F**luff	11
	AE	B**a**t	2		V	**V**alve	11
	AA	H**o**t	3		TH	**Th**in	12
	AO	B**ough**t	3		DH	**Th**en	12
	OW	B**oa**t	4		S	**S**ass	13
	UH	F**oo**t	5		Z	**Z**oo	13
	UW	B**oo**t	4		SH	**Sh**oe	14
	AH	B**u**t	2		ZH	Mea**s**ure	14
	ER	B**i**rd	5		H	**H**ow	2
Diphtongs	AX	Ab**ou**t	4	Plosive (Stops)	P	**P**op	9
	AY	B**uy**	2		B	**B**ib	9
	OY	B**oy**	4		T	**T**op	7
	AW	H**ow**	2		D	**D**id	7
Glides	YU	B**eau**ty	4		K	**K**ick	15
	Y	**Y**ou	6		G	**G**ig	15
Liquids	W	**W**ow	4	Affricatives	CH	**Ch**urch	14
	L	**L**ull	7		J	**J**udge	16
	R	**R**oat	8				

Fig 2.2: Phoneme Table

2.6.2 Visemes according to Flemming and Dobbs

The following visemes are according to the rules of reduction. Recall that the naïve approach would result in a 1-to-1 phoneme/viseme relationship (i.e. 40 visemes would be required)

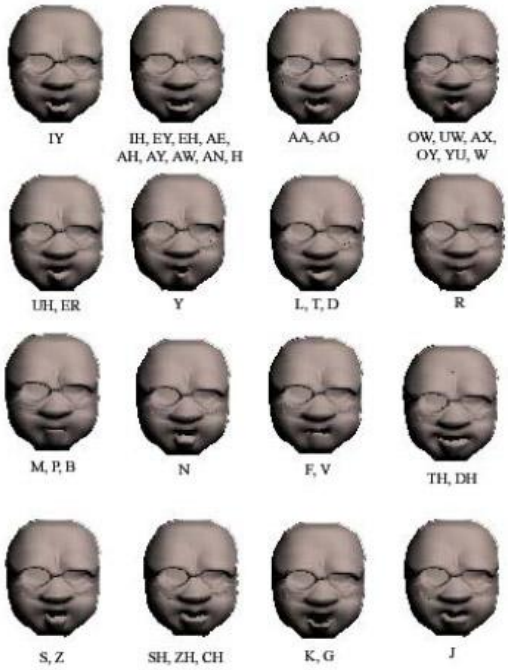


Fig 2.3. Visemes for First Reduction

One thing to be considered is when creating visemes is whether the model has or will have a visible tongue or not. Animating the tongue further complicates things, like the requirement of probably twice the number of visemes displayed above. For simplicity, the tongue will not be considered in the viseme creation.

2.7 First Reduction

The classification of phonemes plays a very important role in lip synch animation. Flemming and Dobbs have phonemes of the English language divided in the following groups:

Consonants:

Fricative (F, V, TH, DH, S, Z, SH ZH and H) – formed by forcing air through a narrow gap, creating a hissing sound.

Plosives (P, B, T, D, K, G) – involve the same restriction of the speech canal as fricatives, but the speech organs are substantially less tense during the articulation of a spirant

Affricative (CH, J) – these are a plosives immediately followed by a fricative in the same place of articulation

Nasal (M, N, AN) – consonants in which air escapes only through the nose

Vowels:

Unitary (IY, EY, EH, AE, AA, AO, OW, UH, UW, AH, ER, AX) – a single syllable sound with no change in articular position

Diphtong (AY, OY, AW, YU) – a gliding, single syllable vowel sound that starts at or near one articulator position and moves toward the position of another

Glide (Y, W) – subclass of diphtongs, but even slower

Liquids (L, R) – another subclass of diphtongs tending to be more like a rolling or thrill sound

Guidelines for animation [Flemming and Dobbs 1999]

1. Unitary phonemes are the strong vowel sounds and should be emphasized with visemes. They should never be dropped.

2. Diphtongs tend to take longer to express. That's why they should be given a higher frame count. They should never be dropped.

3. Glides are slow diphtongs and they have to be given even more frames.

4. Liquids are strong phonemes and they are needed to be accentuated.

5. Fricatives can be accentuated or passed over completely depending on the word. For example, in the word "vice", the V is uttered quickly, so it probably can be dropped completely since it's followed by a strong vowel. On the other hand, in the word "voluptuous" the V is spoken slowly and we need to accentuate it.

6. Plosives are the stop consonants, so they are never emphasized in lip synch animation. They can be completely dropped.

7. Affricatives may never be dropped since they are the strong accent of the word and feature two consonants combined.

Guidelines for Dropping Phonemes (Based on their position)

1. Never drop a phoneme in the beginning of a word
2. Drop nasal visemes between two vowels to smooth transitions

2.8 Second Reduction

From the viseme table in figure 2.3, there are negligible differences between some of the visemes (phoneme groups). This being the case, further reduction can be carried out by merging similar phoneme groups. As a result, the 16 visemes from the first reduction drop to 10:

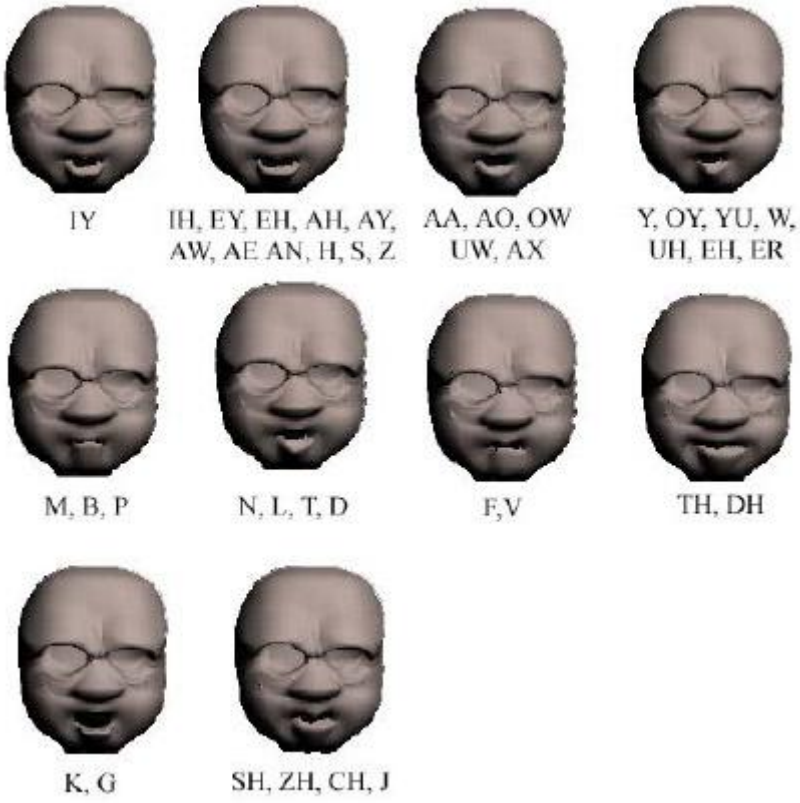


Fig 2.4: Visemes for second Reduction

2.9 Third Reduction

This follows the guidelines, for dropping phonemes, and is merely for reducing the number of key frames to transition among. The number of visemes remains at 10.

It is important to note that further reduction beyond the third level would probably only work well with characters without tongue animation. They would otherwise begin to look non-realistic, with the tongue most likely out of place. It is still important to point out further reduction as they could be applied in cases where level of detail is low.

2.10 Fourth Reduction

Madsen rules for cartoon animation, should one feel the need to further simplify animation. These are based on visual pattern of vowel lip motions accented by the consonants, according to Madsen.

The resulting 4 visemes are as follows:

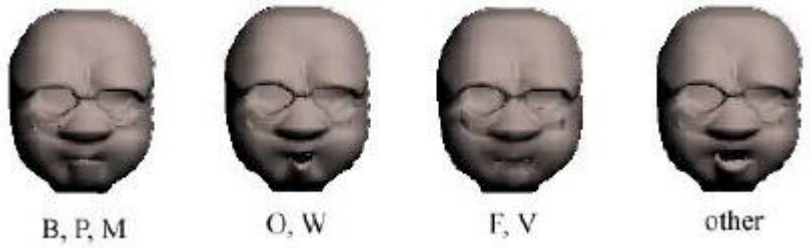


Fig 2.4: Visemes for fourth reduction

B, M, P – lips are closed
O, W – oval mouth shape
F, V – lower lip is tucked under the upper front teeth (in our case upper lip)
OTHER – some intermediate shape of the mouth for all the other phonemes

2.11 Results and Analysis

It was pointed out earlier that full animation led to forced, and unnatural, hence the need for reduction. Rizvic and Avdagic point that at first reduction looks slightly smoother but still appeared to be forced. Their observations showed that at the second reduction, there was not much of a difference visually between 10 & 16 visemes and thus still wasn't

fluid enough. Things began to take shape and looked smooth on the third reduction. One is led to assume that the third reduction would be ideal for realistic facial animation.

The fourth reduction proved to be too simplified and far from accurate, and would be more ideal for less detailed shots. They concluded that Madsen's rules better for wide shots

3 Animation of Facial features

It has been established that both the third and fourth levels of reduction in section 2.11 have their place in computer facial animation. Choice would depend mainly on the level of detail required for a given scene.

For animation with high level of detail, speech animation alone as described by Rizvic and Avdagic will not be sufficient, as there would be no emotion attached to each viseme. Furthermore, the models would remain generic and lacking character. Ideally, one would require a system by which a finite set of key points could aid in controlling a number of vertices to create a semi-automated system of morphing the skin and other facial features. This lends to the idea of physically based approaches to facial animation.

At this stage, we shall take a look at a couple of variations of techniques in [5],[6] & [7], but first, in order to appreciate where we are today, let us take a look at some of the previous approaches adopted over the years for personalized facial animation.

3.1 Previous Approaches for Personalized Facial Animation

Personalized face modeling has been an active area of research since the early 80s. Due to the uniqueness of individuals, animation of specific human faces is a difficult task. Over the last 3 decades several approaches have been brought up in order to achieve visually graphically appealing models that simulate facial animations efficiently. A couple of examples are parametric conformation models, plaster model and deformation

method, image based method, anthropometry based method, scattered data interpolation method and the Direct3D digitization method.

3.1.1 Parametric Conformation models

This was invented in the early 80s. The intended model was to be designed such that it would be capable of creating a wide variety of deformable faces using a limited number of input conformation parameters. Flexibility was an issue due to the inability to develop a parameterization universal enough for such a task. That being the case, heavy manual tuning would be required for these models to be of any use. Unfortunately, this proved to be too difficult a task.

3.1.2 Plaster Model and interactive deformation method

This was a heavily researched approach between the mid 80s to early 90s, made famous by Nadia Magnenat-Thalmann, from the University of Geneva; famous for her work on the creation of a virtual world with virtual actors. The general procedure of this approach was that models would be sculptured from clay to achieve the desired surface detail for a specific face in order to construct an accurate polygonal mesh. Although the results appeared desirable, it was evidently not practical, seeing as this would require artistic skills and time to construct these models.

3.1.3 Image based method

The Image based method [Kurihara and Arai 1991; Akimoto et al. 1993; Ip and Yin 1996; Leeand Magnenat-Thalmann 2000] was a more efficient method than the previous attempts.

This approach utilizes existing 3D models and 2D information from a few images. Functionality was through the use of automatically detected feature points of the 2D images which would help project to 3D. The drawback to this approach was that on average, not enough feature points were identified, which meant that there was not a base robust enough to ensure that a 3D reconstruction would be accurate

Approaches (mid to late 90s)

In the mid to late 90s the Scattered Data Interpolation Method was utilized by a number of approaches brought into the limelight by researchers like Kolja Kähler, Jörg Haber , and Hans-Peter Seidel from Max-Planck-Institute for Information. The idea was to morph a generic face to a more specific one using scattered data interpolation technique. This depended heavily on a good and dense selection of feature points. The only way to achieve this is manually, and this is an arduous task

3.1.4 Direct3D digitization method

Accurate 3D reconstructions are created using Laser Scanning, stereo photogrametry, and active light projection. These models proved to be more suitable for static facial models and aren't ideal for facial deformation and animation for a couple of reason. The first one is that the only information gathered during the scanning process is the outward shape of the human face and there is no underlying mechanism for animation control (facial deformation, jaw control, etc). These controls would need to be manually inserted, which presents a time issue. The second problem is that these scanned models (created via triangle mesh) are generally made up of tens of thousands of triangles. In order to achieve real time animation, the number of triangles should fall to less than ten thousand. Due to the high polygon count, the cost of computation would be too high.

Of the aforementioned methods, some attempt an automated modeling process, but due to the various shortcomings described for each, most require significant manual intervention. A couple of approaches, which have yet to be discussed, show promise for overcoming the obstacles of their aforementioned predecessors. These techniques will be examined in detail over the next couple of sections and later on the advantages and disadvantages for each will be analyzed.

4 Yu Zhang, Terence Sim and Chew Lim Tan

Zhang et al, of the National University of Singapore, developed a relatively new Adaptation-based approach, which is based on adapting an existing low-resolution generic facial model to an acquired surface data to provide a control layer for animation.

4.1 The Generic Control Model

The generic control model contains a known topology, and already has a structure data in place for controlling facial deformation. The geometry and texture of the real face is obtained using a laser range scanner. For the tests conducted by Zhang, Sim and Tan, a Minolta Vivid 900 Digitizer was used. Three scans of the front view and 45 degree side profiles were taken for each subject.

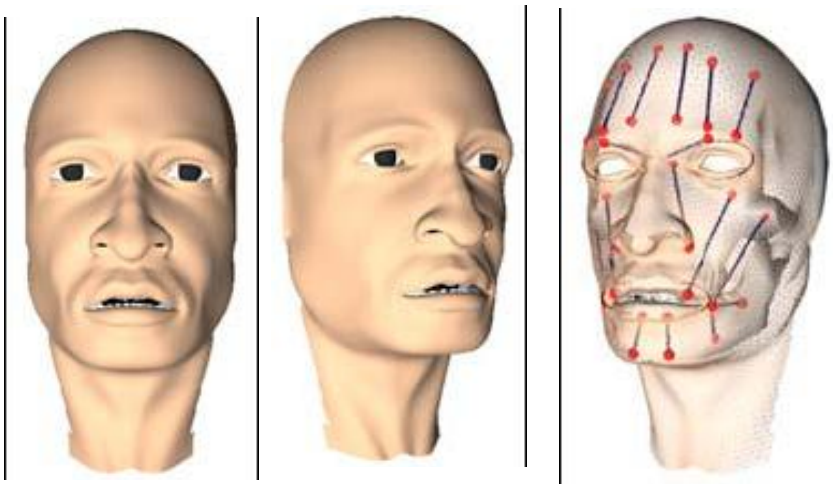


Figure 4.1: Generic Control Model

The Skin Layer is a triangle mesh which consists of 2,848 vertices & 5,277 triangles; well under the ten thousand triangle maximum requirement for real time rendering. To minimize the number of controllable artifacts the mesh edges are aligned to the facial features.

The skin deformation is represented by the skin mesh acting as a mass-spring system in which the vertices act as the point masses of the system, and the edges act as springs. One must note that this system is not perfectly elastic as that would take away any hint of realism in the animation.

The Muscles Layer is attached to the skin layer. 23 of the most important muscles found in the human face are modeled in order to enable a free range of deformation and contraction options during animation control

The Skull is a triangular mesh which is utilized as an anatomical framework for mapping the facial muscles during the construction and initialization process. During render time, the skull is not required.

To finish of the model, Eyes and teeth though not absolutely essential are added as separate components to enhance the overall realism

Facial Deformation or deformation of the facial skin is influenced by muscle contractions and jaw rotation. This is obtained by calculating the energy equilibrium state of the entire mass-spring system using LaGrange dynamics.

Control can be done in a number of ways. On the low level it could be done using an animation script whereby the muscle and jaw parameters are adjusted to get the desired expressions. There is also the option on the high level to implement expression commands.

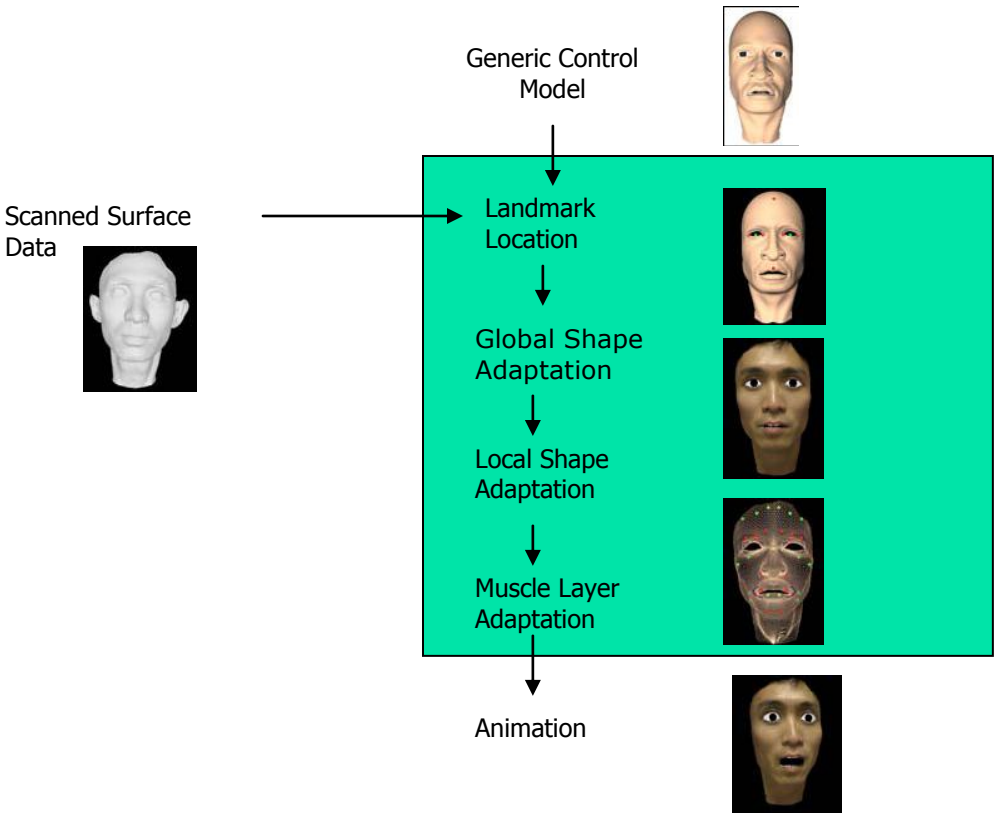


Figure 4.2: Pipeline for reconstructing animated facial model.

4.2 Adapting the Control Model

The Generic Control model denoted as *F* and is referred to as the *source* model. A Hi-res scan model denoted as *F** is referred to as *target* model. The goal in this approach, is to adapt *F* to *F** such that *F* takes on the texture and shape of the specific individual to be animated with predictability. For this to happen a series of steps will be taken as shown in the pipeline (Fig. 1)

The next step is for data from the scanned surface to be adapted to the generic model in the following 4 steps:

Landmark Location,

Global shape adaptation,

Local shape adaptation and

Muscle layer adaptation.

4.3 Landmark Location

This first step takes into account the anthropometric specification of landmarks on the 2D images of both the Generic Control Model and the scanned data. The 3D coordinates would later be recovered by using a projection mapping approach.

In order to achieve this, the image of the control model is stored as a bitmap. RGB values of the facial surface are stored. Zhang et al then specified on the bitmap, a set of landmarks following the anthropometric guidelines rules:

Eye Parameters:

p_l^{le} and p_r^{le} represent the left and right corners of the left eye

p_l^{re} and p_r^{re} represent the left and right corners of the right eye

Mouth Parameters:

p_l^m and p_r^m represent the left and right corners of the mouth

This constitutes the minimum set of landmarks based on their prominence on the face. Surprisingly the nose tip was not in this set.

After specification of the landmarks, the texture coordinates of each vertex on the generic/source model (*F*) are obtained via orthographic projection. The landmarks can be located on the 2d image via cylindrical projection of the 3D face, which maps onto the 2D image plane. The result was a 512x512 cylindrical image. Recall that the RGB values are stored for each pixel. These represent the surface color of the texture mapped surface with corresponding longitude and latitude.

3D positions of landmark points are obtained from mapping the 3d Mesh of *F* and *F** to 2D images using cylindrical projection. Each landmark *p* located within a triangle in the source model mesh can be obtained since each of the triangles have been accounted for in the 3D to 2D projection. They are recovered from normalized barycentric coordinates of *p* in the 2D triangle $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ as follows:

$$\mathrm{p} = \sum_{i=1}^{3} \alpha_i w_i \quad \text{with} \quad \sum_{i=1}^{3} \alpha_i = 1 \qquad 0 \le \alpha_i \le 1 \tag{4.1}$$

where:

p is the 3D position of the landmark *p* and

w_i is one of the vertices of the indexed triangle in which *p* is located

Four more key points were defined using the minimum set defined above:

Centers of the left and right eye :

$$p^{le} = \frac{1}{2}(p_l^{le} + p_r^{le}) \quad \text{and} \quad p^{re} = \frac{1}{2}(p_l^{re} + p_r^{re}) \tag{4.2}$$

Center between both eyes:

$$p^{c} = \frac{1}{2}(p^{le} + p^{re}) \qquad p^{m} = \frac{1}{2}(p_l^{m} + p_r^{m}) \tag{4.3}$$

The target Model (*F**) follows similar steps to that of *F* in terms of landmark specification:

Eye Parameters:

p_l^{*le} and p_r^{*le} represent the left and right corners of the left eye

p_l^{*re} and p_r^{*re} represent the left and right corners of the right eye

Mouth Parameters:

p_l^{*m} and p_r^{*m} represent the left and right corners of the mouth

Similarly, the four extra key points are calculated:

$$p^{*le} = \frac{1}{2}(p_l^{*le} + p_r^{*le}) \quad \& \quad p^{*re} = \frac{1}{2}(p_l^{*re} + p_r^{*re})$$

$$p^{*c} = \frac{1}{2}(p^{*le} + p^{*re}) \quad \& \quad p^{*m} = \frac{1}{2}(p_l^{*m} + p_r^{*m})$$

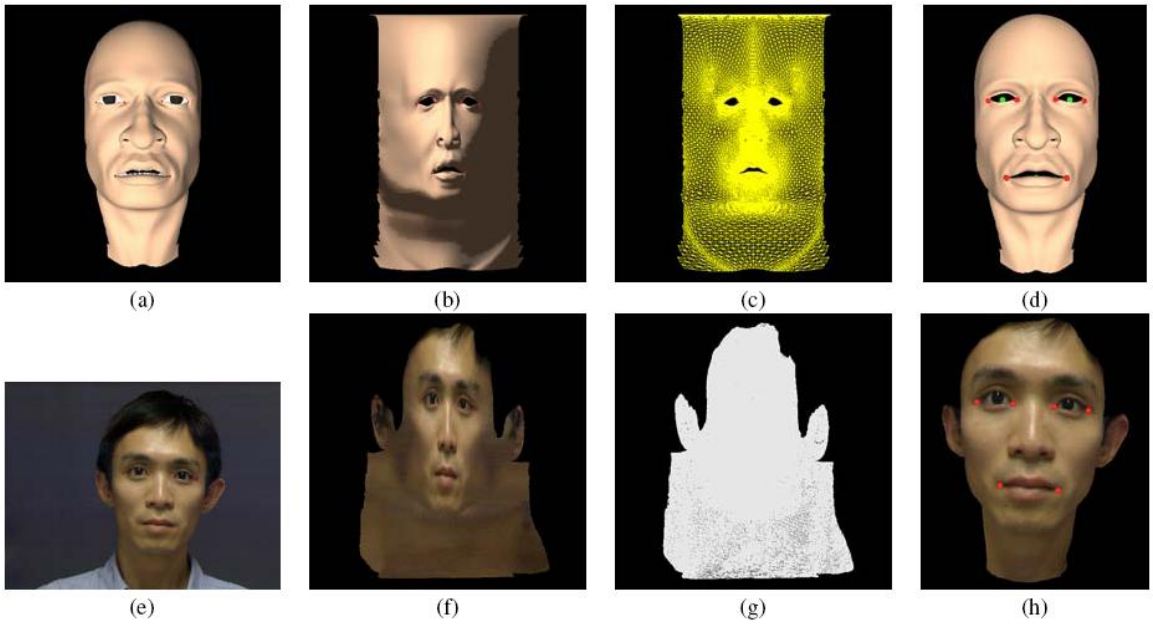


Figure 4.3: Specification of landmarks and recovery of their 3D positions based on a projection-mapping approach:
(a) landmark image of the source model; (b) projected textured-mapped source model; (c) projected surface mesh of the source model; (d) recovered 3D positions of specified landmarks (red points) and key points (green points) on the source model; (e) landmark image of the target model; (f) projected textured-mapped target model; (g) projected surface mesh of the target model; (h) recovered 3D positions of specified landmarks (red points) on the target model (key points are occluded).

4.4 Global Shape Adaptation

Here is where the generic model is aligned with the scanned face in 3D space using measurements between landmark locations obtained in step one.

This step involves some rotation and translation. The first step would be to transform F^* by aligning an imaginary line connection the eye centers ($p^{*le}p^{*re}$) with the world x-axis and is also have the y-z plane cut through the center of F^* (i.e. $p^{*c}p^{*m}$ should be on the y-z plane).

Zhang et al, go about this in the following manner:

Let an arbitrary vertex $x_i^* \in R^3$ on F^* move to a new position $x_i^{*\prime} \in R^3$. And let $\mathbf{R^*} \in R^{3\times3}$ be the rotation matrix, $\mathbf{T^*} \in R^3$ be the translation vector, and $C^{*0} \in R^3$ be the face model center. The transformation equates to:

$$x_i^{*\prime} = R^*(x_i^* - C^{*0}) + T^* \tag{4.4}$$

C^{*0} is defined as the center between both eyes of F^*

$$C^{*0} = p^{*c} \tag{4.5}$$

$\mathbf{R}^*$ is comprised of 3 rotation angles. Zhang et al define them as:

Face tilt (r_x^*) around the *x*-axis

Face rotation (r_y^*) around the *y*-axis

Face inclination (r_z^*) around the *z*-axis

They project $p^{*le}p^{*re}$ of F^* onto the *x*-*z* plane using orthographic projection as seen in Fig. 4.4.

r_y^* is calculated as:

$$r_y^* = \begin{cases} \arccos\left(\dfrac{\overrightarrow{p^{*re|xz}p^{*le|xz}} \cdot \overrightarrow{n_x}}{\| \overrightarrow{p^{*re|xz}p^{*le|xz}} \|}\right) & if\ (\overrightarrow{p^{*re|xz}p^{*le|xz}} \times \overrightarrow{n_x}) \cdot \overrightarrow{n_y} > 0 \\ -\arccos\left(\dfrac{\overrightarrow{p^{*re|xz}p^{*le|xz}} \cdot \overrightarrow{n_x}}{\| \overrightarrow{p^{*re|xz}p^{*le|xz}} \|}\right) & otherwise \end{cases} \tag{4.6}$$

where $\overrightarrow{p^{*re|xz}p^{*le|xz}} = p^{*re|xz} - p^{*le|xz}$ and $\overrightarrow{n_x}$ and $\overrightarrow{n_y}$ are unit vectors of the *x*- and *y*-axes

r_z^* is calculated as:

$$\left|r_z^*\right| = \arccos\left(\frac{\overrightarrow{p^{*re}p^{*le}} \cdot \overrightarrow{p^{*re|xz}p^{*le|xz}}}{\left\|\overrightarrow{p^{*re}p^{*le}}\right\| \left\|\overrightarrow{p^{*re|xz}p^{*le|xz}}\right\|}\right) \tag{4.7}$$

To determine the direction of rotation assuming $p^{*re'}$ and $p^{*le'}$ are the new positions of $p^{*le}p^{*re}$:

$$r_y^* = \begin{cases} \left|r_z^*\right| & if\ (\overrightarrow{p^{*re'}p^{*le'}} \times \overrightarrow{n_x}) \cdot \overrightarrow{n_z} > 0 \\ -\left|r_z^*\right| & otherwise \end{cases} \tag{4.8}$$

r_x^* is calculated using the angle between eye-mouth plane normal ($\overrightarrow{n^{em}}$), and the *z*-axis

$$r_x^* = \begin{cases} \arccos(\overrightarrow{n^{em}} \cdot \overrightarrow{n_z}) & if\ (\overrightarrow{n^{em}} \times \overrightarrow{n_z}) \cdot \overrightarrow{n_x} > 0 \\ -\arccos(\overrightarrow{n^{em}} \cdot \overrightarrow{n_z}) & otherwise \end{cases} \tag{4.9}$$

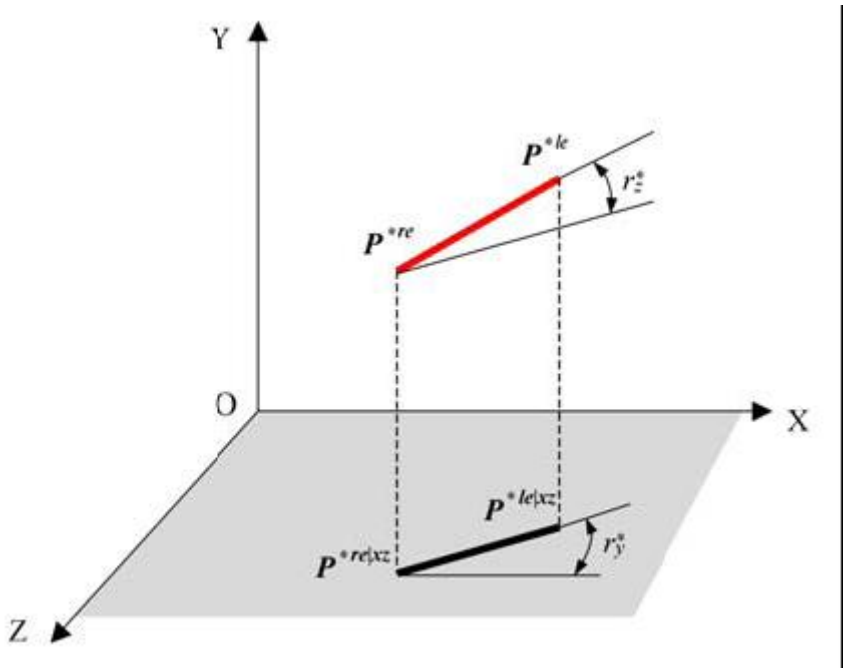


Fig. 4.5

Translation component **T*** = $(t_x^*, t_y^*, t_z^*)^T \in R^3$:

$$t_x^* = 0,\ t_y^* = p_y^{*c},\ t_z^* = p_z^{*c} \tag{4.10}$$

Zhang et al also needed to scaled, rotated and translated to match *F* to *F**

Let a vertex $x_i \in R^3$ on *F* move to a new position $x_i^{/} \in R^3$. And let $\mathbf{R} \in R^{3\times3}$ be the rotation matrix, $\mathbf{T} \in R^3$ be the translation vector, and $C^0 \in R^3$ be the face model center. The transformation equates to:

$$x_i^{/} = SR(x_i - C^0) + T \tag{4.11}$$

As with *F**, C^0 is defined as the center between both eyes of *F*

$$C^0 = p^c \tag{4.12}$$

$\mathbf{S} \in R^{3\times3}$ is the scaling matrix

$$S = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \tag{4.13}$$

Translation component **T** = $(t_x, t_y, t_z)^T \in R^3$:

$$t_x = 0,\ t_y = p_y^{*c},\ t_z = p_z^c \tag{4.14}$$

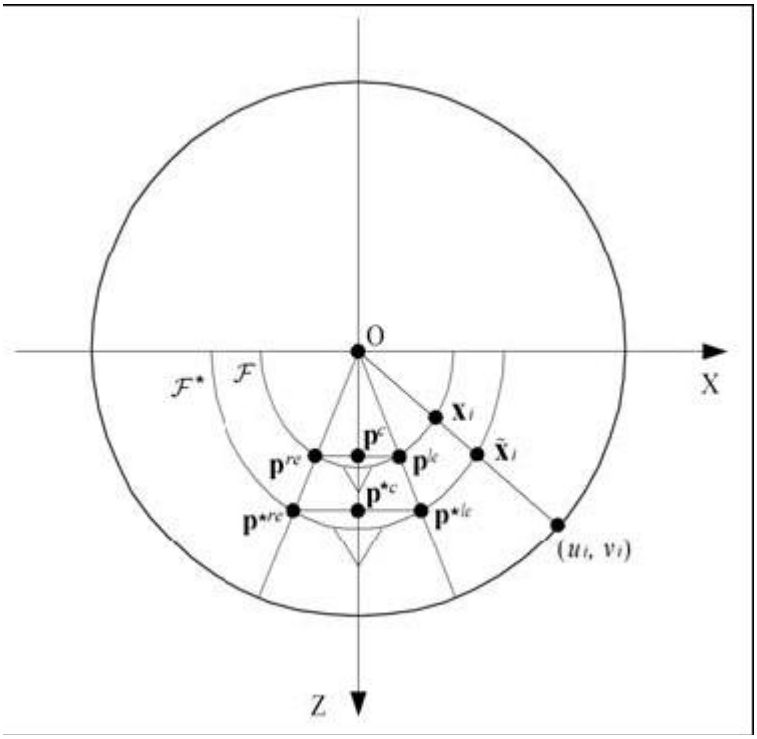


Figure 4.6: Local adaptation of the source model to the target model using cylindrical projection.

The scaling factors were calculated as follows:

$$s_x = \frac{p_z^c \left\| p^{*le} - p^{*re} \right\|}{p_z^{*c} \left\| p^{le} - p^{re} \right\|} \tag{4.15}$$

$$s_y = \frac{\left\| p^{*c} - p^{*m} \right\|}{\left\| p^{c} - p^{m} \right\|} \quad , \qquad s_z = \frac{1}{2}\left(s_x + s_y \right) \tag{4.16}$$

4.5 Local Shape Adaptation

This step is where the vertices of the generic model are mapped or fitted onto scanned surface data, by shifting vertices of *F* to match the scan data of *F**. The fitting is a tricky process in which there needs to be a manner of determining the corresponding 3D position of each vertex of *F* on *F**. This was achieved by Zhang et al using cylindrical projection as shown in Fig. 4.6

For the image coordinates (u_i, v_i) of a vertex $\vec{x}_i = (x_i, y_i, z_i)$ on F:

$$u_i = \arctan\left(\frac{x_i}{z_i}\right), \qquad v_i = y_i \tag{4.17}$$

Likewise, (u_i^*, v_i^*) is computed, and are found in the projected triangular mesh of F^*. Zhang et al define a point (u_i, v_i) in the triangle $\Delta((u_1^*, v_1^*), (u_2^*, v_2^*), (u_3^*, v_3^*))$ as a linear interpolation using barycentric coordinates as:

$$u_i = \sum_{j=i}^{3} \beta_j u_j^*, \qquad v_i = \sum_{j=i}^{3} \beta_j v_j^* \tag{4.18}$$

where β_j are barycentric coordinates.

Given $(\beta_1, \beta_2, \beta_3)$ in the 2D triangle, the 3D position of $\tilde{x}_i$ was calculated as follows:

$$\tilde{x}_i = \sum_{j=i}^{3} \beta_j \vec{x}_j^* \tag{4.19}$$

where $\vec{x}_j^*$ is a 3D position of one of the points that make up a given triangle on F^*

After adaptation, the texture coordinates of $\tilde{x}_i, (\tilde{s}_i, \tilde{t}_i)$ were computed:

$$\tilde{s}_i = \sum_{j=i}^{3} \beta_j \vec{s}_j^*, \qquad \tilde{t}_i = \sum_{j=i}^{3} \beta_j \vec{t}_j^* \tag{4.20}$$

Where $(\vec{s}_j^*, \vec{t}_j^*)$ are the texture coordinates associated with three vertices of the triangle on F^* that contains $\tilde{x}_i$.

4.6 Muscle Layer Adaptation

Here, the muscles defined beneath the skin of the generic model are transferred to new skin geometry. The muscle layer is essential for animation and resolves the issue of having no underlying control mentioned earlier with the Direct3D approach.

Zhang et al took into account 2 muscle groups: 1) Linear and 2) Sphincter Muscles. These muscles form a layer in between the skull and Skin layers and are the driving forces for the deformation of the skin mesh.

4.6.1 Linear Muscles

Each is defined by a fixed point on the skull referred to as the muscle attachment point, which is connected to a muscle insertion point that is attached to the skin

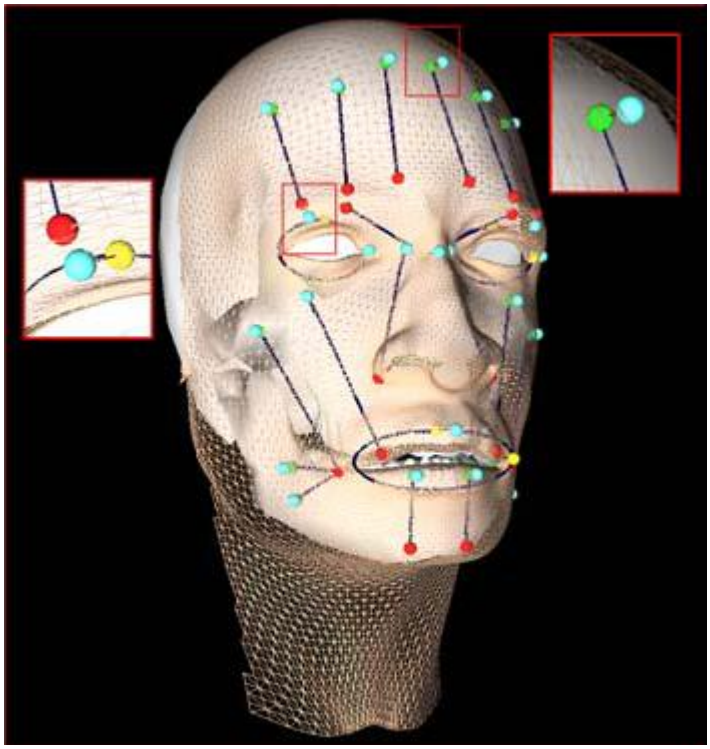


Figure 4.7: Muscle points and their corresponding skin points. Red and green points represent the insertion and attachment points of linear muscles respectively, yellow points represent sphincter muscle points, azure points represent the corresponding points of the linear muscle attachment points and sphincter muscle points on the skin mesh, and blue lines represent central muscle fibers.

4.7 Interpolation Computations

$$m_i^I = \sum_{j=1}^{3} \gamma_j^I x_j^k \tag{4.21}$$

where:

m_i^I is a muscle insertion point located on a skin triangle $t_k = (x_1^k, x_2^k, x_3^k)$

The value γ_j^I is said to be a barycentric coordinate of non negative scalar variables, but it is unclear what this value is required for; possibly some elasticity coefficient.

Compute the new position of the insertion point base on the new position of t_k, ($\tilde{t}_k = (\tilde{x}_1^k, \tilde{x}_2^k, \tilde{x}_3^k)$), we get:

$$\tilde{m}_i^I = \sum_{j=1}^{3} \gamma_j^I \tilde{x}_j^k \tag{4.22}$$

Computing the position each muscle attachment point m_i^A requires computing the corresponding skin point $m_i^{A(skin)}$ (point which when projected via the normal of m_i^A)

$$m_i^{A(skin)} = \sum_{j=1}^{3} \gamma_j^A x_j^l \tag{4.23}$$

where:

$m_i^{A(skin)}$ is the point which when ray cast from the normal at m_i^A, makes contact with the skin

x_j^l are points of the skin triangle in which m_i^A is in.

As illustrated in Fig. 4.7, the relationship between m_i^A and $m_i^{A(skin)}$ is defined by d_i

$$d_i = m_i^{A(skin)} - m_i^A \tag{4.24}$$

After mesh adaptation, $m_i^{A(skin)}$ is updated as follows:

$$\tilde{m}_i^{A(skin)} = \sum_{j=1}^{3} \gamma_j^A \tilde{x}_j^l \tag{4.25}$$

Since we now have $\tilde{m}_i^{A(skin)}$ and we know d_i calculating the new position of m_i^A is simply:

$$\tilde{m}_i^A = \tilde{m}_i^{A(skin)} - d_i \tag{4.26}$$

4.8 Sphincter Muscles

Unlike linear muscles, they are elliptical. Zhang et al define this type of muscle, with 3 parameters, $(\vec{o}_i, \vec{a}_i, \vec{b}_i)$ where

$\vec{o}_i$ is the virtual center of the muscle

$\vec{a}_i, \vec{b}_i$ are located at the half lengths of the axes of the elliptical shape of the muscle

Adaptation is computed by locating the end of the axes m_i^a and m_i^b. Skin surface points $m_i^{a(skin)}$ & $m_i^{b(skin)}$ are calculated similarly to the linear muscles. Like wise, the offsets are calculated as such:

$$d_i^a = m_i^{a(skin)} - m_i^a, \qquad d_i^b = m_i^{b(skin)} - m_i^b \tag{4.27}$$

Computing new positions after adaptation is also similar to linear muscles

$$\tilde{m}_i^a = \tilde{m}_i^{a(skin)} - d_i^a, \qquad \tilde{m}_i^b = \tilde{m}_i^{b(skin)} - d_i^b \tag{4.28}$$

$(\vec{o}_i, \vec{a}_i, \vec{b}_i)$ is determined as follows:

$$o_{i(x)} = \tilde{m}_{i(x),}^b \qquad o_{i(y)} = \tilde{m}_{i(y),}^b \qquad o_{i(z)} = \frac{1}{2}(\tilde{m}_{i(z)}^a + \tilde{m}_{i(z)}^b) \tag{4.29}$$

$$a_i = \left|\tilde{m}_{i(x)}^a - o_{i(x)}\right|, \qquad b_i = \left|\tilde{m}_{i(y)}^b - o_{i(y)}\right| \tag{4.30}$$

4.9 Results and Analysis

It becomes apparent that this adaptive approach possesses advantages over its predecessors. It has a fully functional generic model as a base. Manual intervention is kept to a minimum, so it is a more efficient process which contains a fully automated muscle adaptation technique.

The following are finished products of the Adaptive Approach:

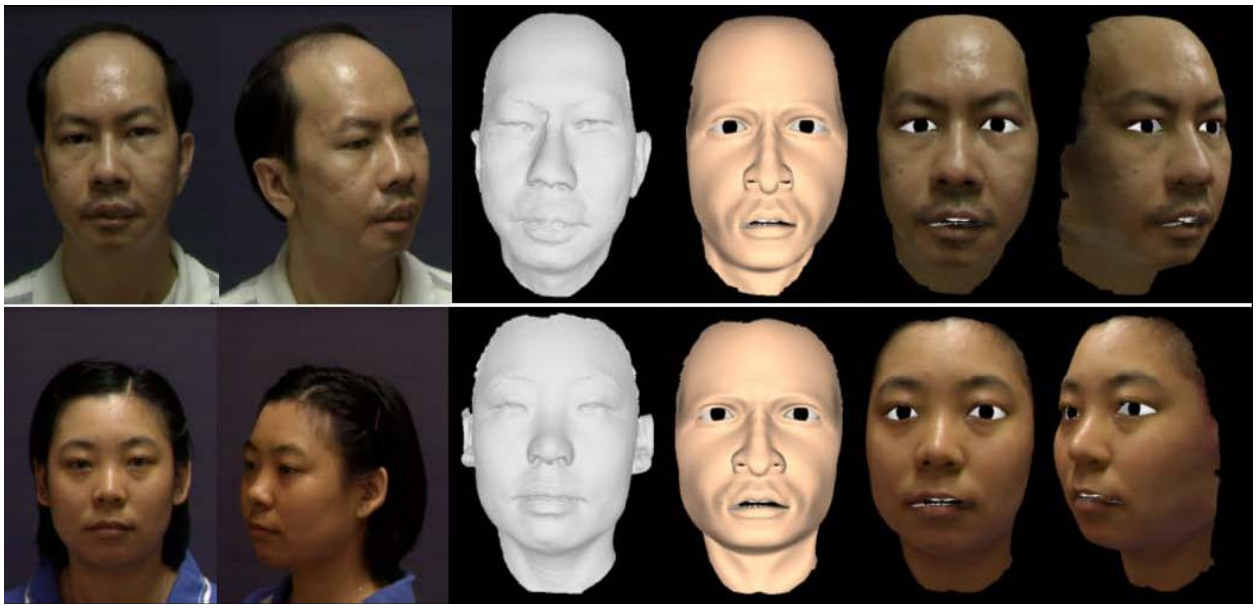


Figure 4.8: Sample results of the adaptation

5 Kolja Kähler, Jorg Haber and Hans-Peter Seidel

Similar to [5], this approach takes the approach of adapting underlying muscle, of a scanned facial surface. The muscles are grouped depending on the location of the face, and skin deformations under physically based conditions that mimic the human anatomy.

To facilitate their work, the authors of [6] developed a muscle editor capable of modifying the source model. Their reason for opting for muscle control as opposed to the previously tested parametric control was as a result of weighing out the benefits of each. Though parametric control isn't costly, interpolation becomes an issue without a physically based underlying structure, which with the commonly used mass-spring network system, would display a higher degree of accuracy in terms of modeling the (visco-)elastic properties of skin.

5.1 Approach Model

As with [5] based their model on three conceptual layers:

1. Skin / tissue layer representing the epidermis and subcutaneous fatty tissue;
2. A layer of muscles attached to the skull and inserting into the skin
3. Underlying bone structure, composed of immovable skull and rotating jaw.

The input data is composed of arbitrary triangle mesh representing the skin geometry and the skull geometry and facial muscle layout created semi-automatically. There is no mention on whether the skull was modifiable, but with the muscle editor, Kähler et al were able to tweak the layout of the muscle groups for the face. One would assume that that spring and dampening coefficients (to be discussed later) could be modified with this editor.

With the layer system in place, animation of the face is achieved by physics-based simulation of a mass-spring system that connects the three layers.

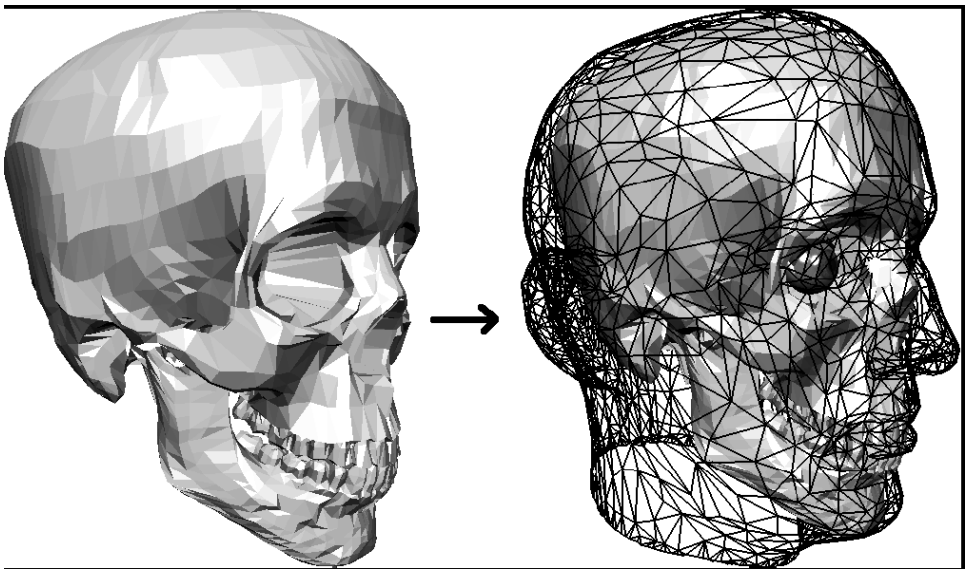


Figure #.1: Generic skull model fitted to head using affine transformation for estimating assignment of skin regions to skull and jaw

5.2 Skull and Jaw

Similar to [5], a skeletal underlying mesh was utilized. This comprised of the jaw and the skull. This was setup to determine the nature of the behavior of the skin mesh. For the skin covering the jaw area, their movement will be determined by the rotation of the jaw.

It is not pointed out in the article, but one would assume that the lower lip would still be under the influence of a muscle-based mass spring system.

These meshes (for the skull and jaw) are not rendered during the final stages of the animation process. They are used only while building muscles in the editor and during the startup phase of the animation system and also possibly for skull penetration constraint testing, though the authors of [6] only mention that this (skull penetration constraints) is handled internally to the mass-spring mesh.

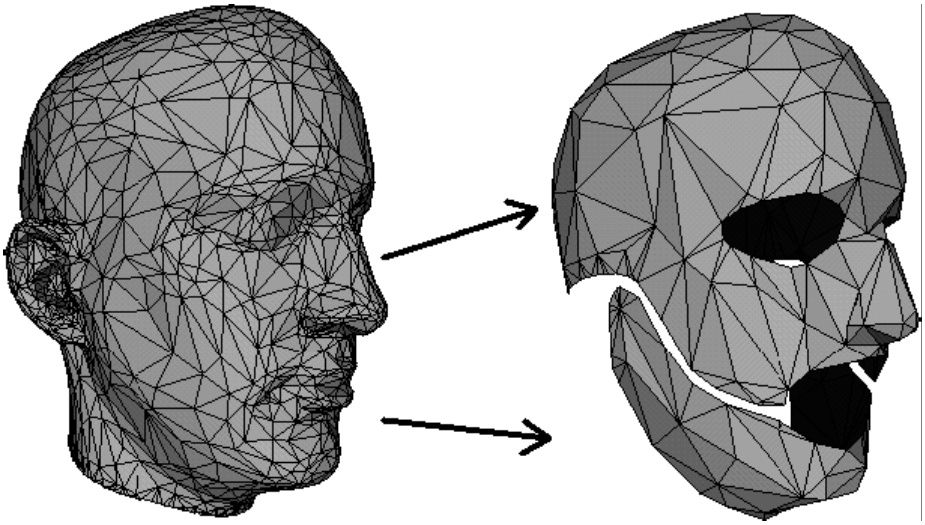


Figure #.2: Head model prepared from scan data (left), cut up and simplified to represent the skull (right).

5.3 Muscles

The muscles are made up of segments, which either contract towards the end attached to the skull (*linear muscle*) or towards a point (*circular muscle*), as was the case with [5]. However, Kähler et al go into more detail as to how the muscles interact with each other. Whether this extra step makes for an improvement over [5] remains to be seen, however it should be explored nonetheless

Each of the segments is represented by an ellipsoidal shape, and the muscles form a more complex system, in which they would be allowed to slide freely across each other

Muscle types are as follows:

(1) Linear

(2) Sheet

(3) Curved

(4) Sphincter

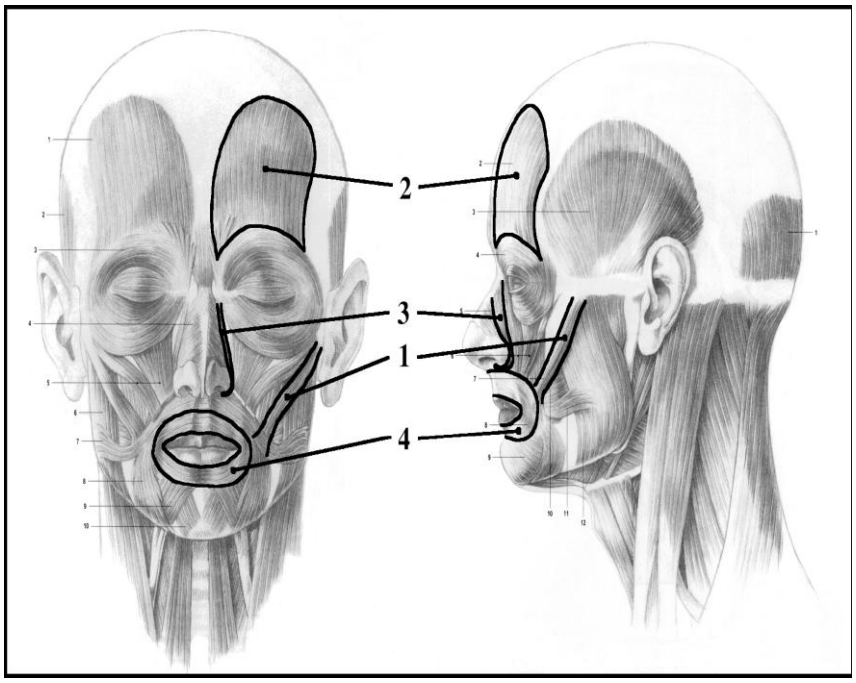


Fig. 5.3 Different Types of Muscles: (1) Linear (2) Sheet (3) Curved (4) Sphincter

Muscles (1) to (3) expand and contract in a similar manner, while the sphincter muscle (4) acts a little different in that its deformation is based on a virtual center, as touched on in [5]. From this point on, calculations for muscles (1) to (3) will be referred to as linear muscle calculations

5.4 Skin and Tissue Simulation Guidelines

- The top layer of model represents the skin
- Skin layer connects to muscles and bones
- Nodes and edges of the input triangle mesh comprise initial spring mesh.
- Each surface node is connected to either the bone layer or to an underlying muscle by a spring with low stiffness.

It is important to note that there is the risk of the skin penetrate skull, if it is treated as a simple membrane. There is also the issue of the skin folding over itself. To enforce this, Kähler et al employed methods for local preservation proposed by [7] They also attached

another outward pulling spring as illustrated in Fig. #.4. They are designed to mirror the muscle attachments to force the surface nodes to move tangentially to the skull and muscle surface, and thus preventing penetration.

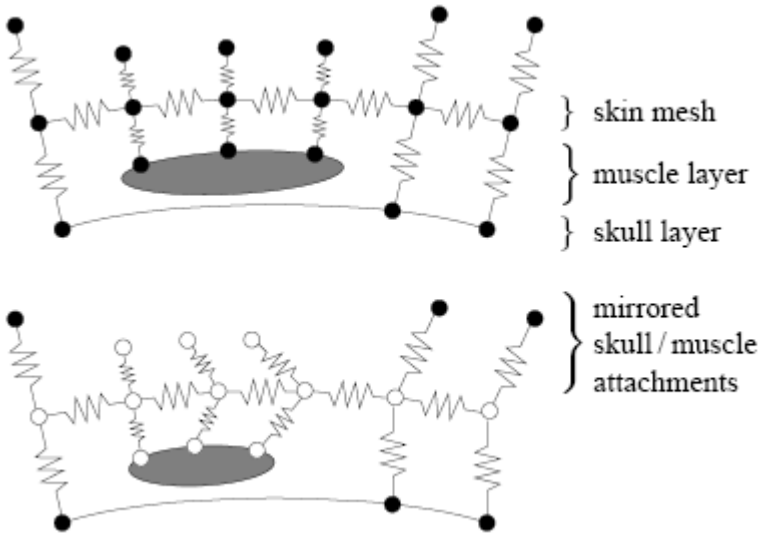


Fig. #.4: Mass-spring system in our model. Top: Relaxed muscle, outer springs mirroring skull and muscle attachments. Bottom: Contracted muscle, with mass points moving due to the contraction marked by ○

The equations for motion for the mass-spring system numerically integrated through time using explicit forward integration scheme.

5.5 Quick Verlet Integration Review:

$$\begin{aligned} x(t_0 + \Delta t) &= x(t_0) + (x(t_0) - x(t_0 - \Delta t)) + a\Delta t^2 \\ &= 2x(t_0) - x(t_0 - \Delta t) + a\Delta t^2 \end{aligned} \tag{5.1}$$

where

t_0 is the current time and Δt is the time step.

The velocity at each time step is then not calculated until the next time step.

$$v(t_0) = \frac{x(t_0 + \Delta t) - x(t_0 - \Delta t)}{2\Delta t} \tag{5.2}$$

5.6 Muscle Model Calculations

Kähler et al base the muscles on individual fibers that are composed of piecewise linear segments. These segments were represented by quadric shapes (ellipsoid) aligned with them, and scaled to the length of the muscle segment.

They further define the structure of these fibers as having *n* control points $p_i \in R^3 (i = 0,1,...,n-1)$ as illustrated in Fig. 5.5. These control points make up a control polygon *P*

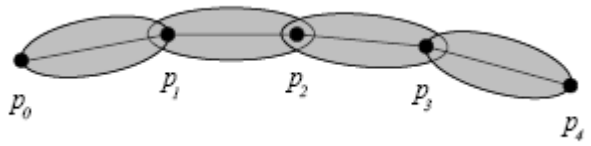


Fig. 5.5 Muscle fiber with control polygon P and per-segment ellipsoids.

5.7.1 Contraction

On contraction of a control polygon $P = \{p_i\}_{i=0}^{n-1}$ to a resized control polygon $Q = \{q_i\}_{i=0}^{n-1}$, each parameter is assigned a parameter $t_i \in [0,1]$:

$$t_i = \begin{cases} 0 & ,if\ i = 0 \\ \dfrac{\sum_{j=1}^{i} \left\| p_j - p_{j-1} \right\|}{\sum_{j=1}^{n-1} \left\| p_j - p_{j-1} \right\|} & ,else \end{cases} \tag{5.3}$$

Scaling is performed using a contraction factor *c*:

$$\tilde{t}_i = \max\{(1-c)t_i, 0.01\} \tag{5.4}$$

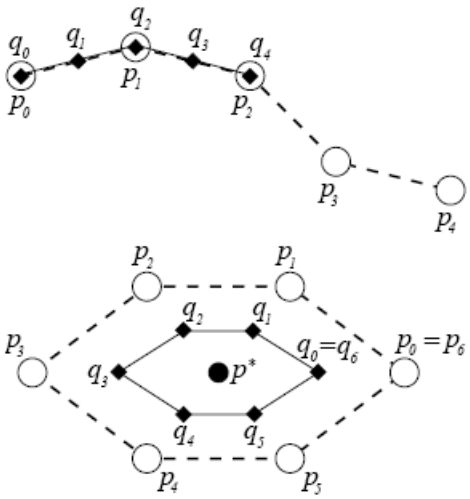


Figure 5.6: Contraction (c = 1/2) of a linear (top) and a sphincter (bottom) muscle fiber.
The control points $\boldsymbol{p}_{i=0\ldots6}$ and $\boldsymbol{q}_{i=0\ldots6}$ represent the relaxed and contracted muscle.

Next $\tilde{t}_i$ gets mapped to a index $k_i \in \{0,\ldots,n-2\}$ of the starting point of $\tilde{t}_i$:

$$k_i = \begin{cases} 0 & ,if\ i = 0, \\ m : t_m < \tilde{t}_i \le t_{m+1} & ,else \end{cases} \tag{5.5}$$

The value *m* is however not defined. This is possibly an oversight in the article.

With that (5.3), (5.4) & (5.5) the new control Polygon *Q* can now be computed.

For linear muscles:

$$q_i = p_{k_i} + (p_{k_i+1} - p_{k_i}) \frac{\tilde{t}_i - \tilde{t}_{k_i}}{\tilde{t}_{k_{i+1}} - t_{k_i}} \tag{5.6}$$

For sphincter muscles:

$$q_i = p^* + (1-c)\left\| p_i - p^* \right\| \tag{5.7}$$

where:

p^* is the center point of the muscle

There is no mention by [Kähler et al] as to whether clamping is performed for the contraction factor, but based on earlier calculations, this will be implied.

5.7.2 Bulge

To simulate realistic muscle behavior for linear muscles, muscle should get thinner on expansion and thicker on contraction, with the center of the muscle exhibiting the biggest bulge. Sphincter muscles will act in a slightly different manner, in the sense that the muscle will bulge evenly

[Kähler et al] achieve this by introducing a scaling factor $s_i \in [0,1]$, computed from l_i^r and l_i^c whereby it scales the height of each muscle segment $\overline{p_i p_{i+1}}$ by $(1+2s_i)$ where:

$l_i^r = \|p_{i+1} - p_i\|$ is the rest length

$l_i^c = \|q_{i+1} - q_i\|$ is the current length

So then:

$$s_i = 1 - l_i^c / l_i^r \tag{5.8}$$

If $n > 3$ then

$$s_i = (1 - l_i^c / l_i^r)\left[1 - \left(\frac{2i}{n-2} - 1\right)^2\right] \tag{5.9}$$

5.8 Building Muscles from Geometry

1. Load a face mesh and display it in the editor.
2. Lay out the fixed end (i.e. the origin) of a muscle by specifying at least two grid points.
3. Sketch the basic muscle grid row by row.
4. For a sphincter muscle: specify the center of contraction.
5. The muscle grid is refined automatically to fit the geometry and the muscle is inserted, making it fully functional for immediate testing and re-editing.
6. Goto step 2 until all muscles are specified.

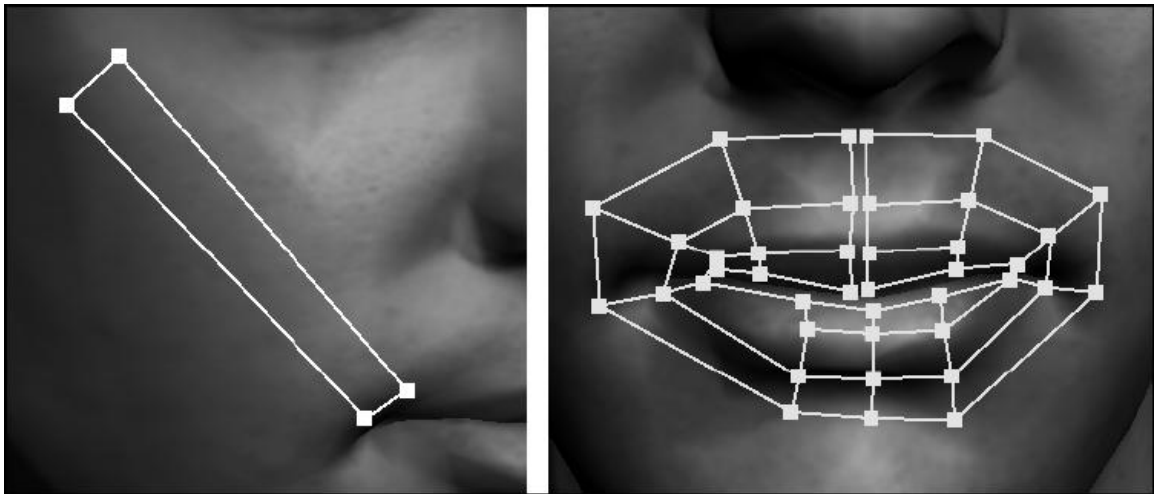


Figure 5.7: A simple grid (left, zygomatic major) and a non-uniform complex grid (right, orbicularis oris).

5.9 Optimizing Muscle Shape Guidelines

- The surface of a muscle must lie within a prescribed distance range below the skin surface.
- Muscles that are well-adapted to the resolution of the skin mesh
- Given the skin mesh and a muscle grid, optimization step determines the following parameters that are needed to create the muscles:
    - the number of muscle fibers;
    - the number of segments per fiber;
    - width, height, and length of each segment;
    - position of the muscle fiber control points;
    - alignment of the quadrics' coordinate systems.

5.9.1 Initializing the grid:

- The initial outline is converted into a regular grid, i.e. all rows are assigned the same number of grid points. The grid points are then projected onto the face mesh and placed slightly underneath the skin surface.

5.9.2 Refining the grid:

- The grid is adaptively refined until a decent approximation has been found.

5.9.3 Creating the muscle:

- Muscle fibers are created and aligned to the refined grid.

5.9.4 Attaching the muscle:

- The muscle is attached to the spring mesh, and the control points of the muscle segments are attached to either the skull or the jaw.

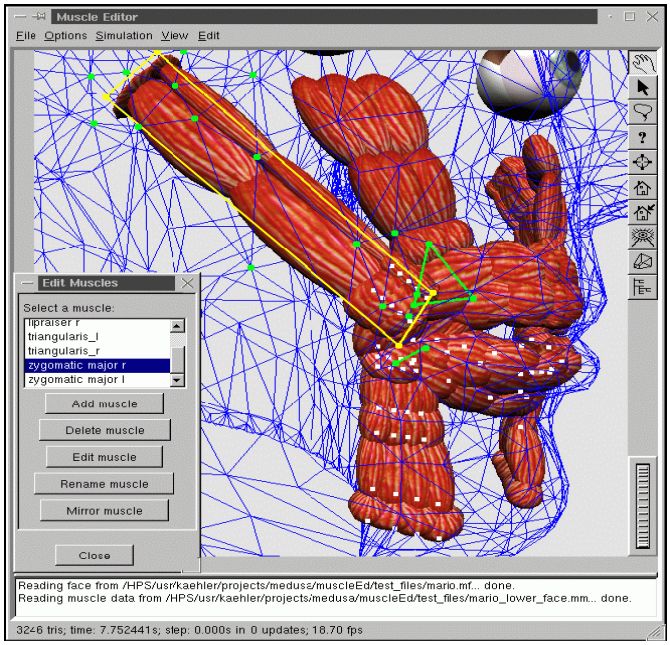


Figure 5.8: Visual information while editing a muscle: the muscle grid of the currently edited muscle(yellow); the skin vertices influenced by this muscle (green dots); muscle control points attached to the jaw (white dots); merged muscle segments (connected by green lines).

5.10 Results and Analysis

- **Pros:**
  - The muscle model itself performs well with low computational overhead.
  - Once a rough layout for a muscle has been specified, the muscle can be automatically rebuilt from this data for many different head models.
- **Cons**
  - Polygon count may be undesirable

- Skin thickness must remain constant

6 Yuencheng Lee, Demetri Terzopoulos, and Keith Waters

This technique improves on the physics-based facial modeling approach proposed by Terzopoulos and Waters. It promises to reduce the amount of work that must be done by the animator. Also present in this installment is the ability to articulate the neck.

The process begins with scanning and creation of the facial mesh, and as with other approaches care must be taken not to over-sample the surface because there is a trade-off between the number of nodes and the computational cost of the model

The set of key/landmark points differ slightly from that of [5]. Positions of the nose, eyes, chin, ears, and other facial features with respect to the generic mesh are taken into account. Facial muscle emergence and attachment points are also known relative to the generic mesh and are adapted automatically as the mesh is conformed to the scanned data.

6.1 Image Processing

The technique applied to image processing is similar to that employed by the authors of [5]. Image resolution on of these scans however was 512x256.

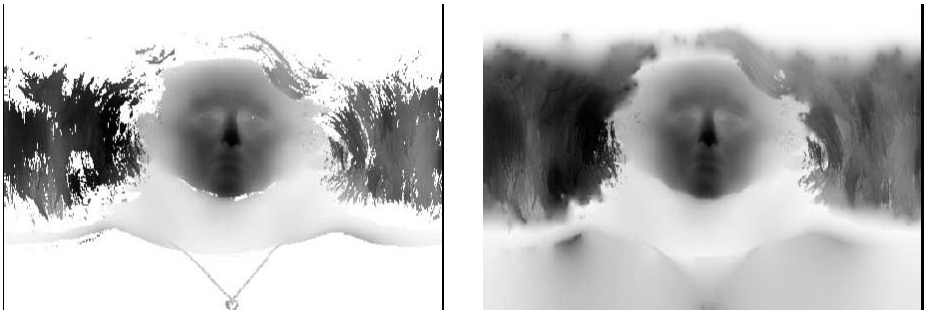


Figure 6.1: (Left) Range data of "Grace" from a Cyberware scanner.
(Right) Recovered plain data.

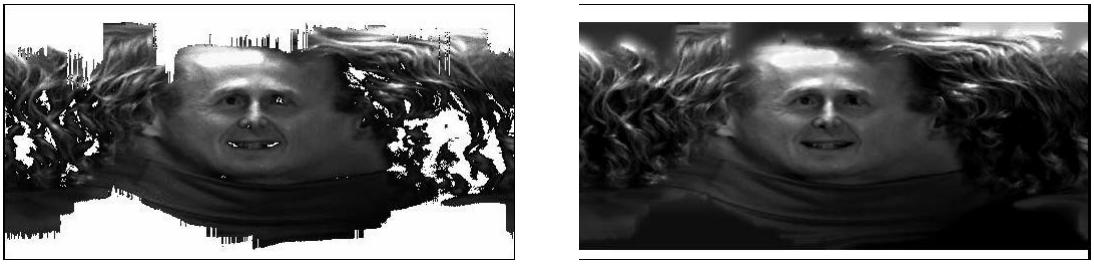


Figure 6.2: (Left) Texture data of "George" with void points displayed in white and (Right) texture image interpolated using relaxation method.

6.2 Generic Face Mesh and Mesh Adaptation

Automatically produces an efficient triangulation, with finer triangles over the highly curved and/or highly articulate regions of the face, such as the eyes and mouth, and larger triangles elsewhere.

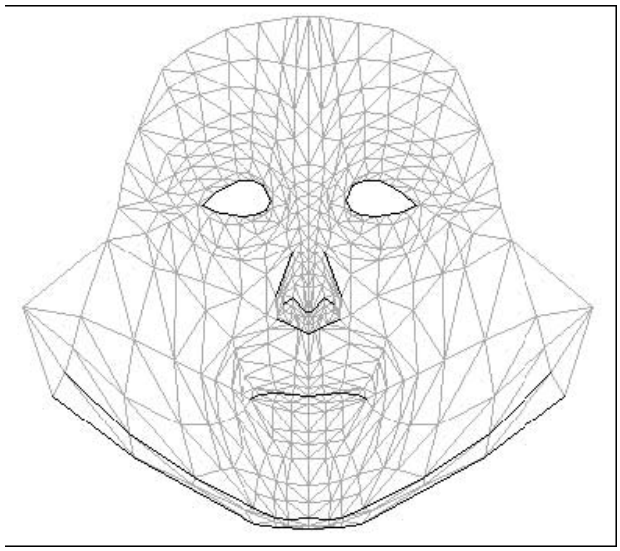


Figure 2: Facial Portion of Generic Mesh in 2D cylindrical coordinates

6.2.1 Mesh Adaptation Procedure

Lee et al outline the mesh adaptation procedure as follows

1. Locate nose tip
2. Locate chin tip
3. Locate mouth contour
4. Locate chin contour

5. Locate ears
6. Locate eyes
7. Activate spring forces
8. Adapt hair mesh
9. Adapt body mesh
10. Store texture coordinates

Lee et al claim to be able to normalize the expression of a given scanned face, should it not be in a neutral state:

1. Perform mesh adaptation procedure without step 3.
2. Store nodal longitude/latitude into adapted face model.
3. Perform lip adaptation in step 3 of the mesh adaptation procedure
4. Store nodal range values into adapted face model.

6.3 The Dynamic Skin and Muscle Model

Upon adaptation a dynamic model of facial tissue is developed. For this model as with other techniques we estimate a skull surface, and insert the major muscles of facial expression into the model. The following sections describe each of these components.

6.4 Layered Synthetic Tissue Model

The skull is covered by deformable tissue which has five distinct layers. Four layers make up the epidermis, dermis, sub-cutaneous connective tissue, and fascia comprise the skin. The fifth layer consists of the muscles of facial expression. Following the work of D. Terzopoulos and K. Waters (Physically-based facial modeling) and in accordance with the structure of real skin, have designed a new, synthetic tissue model

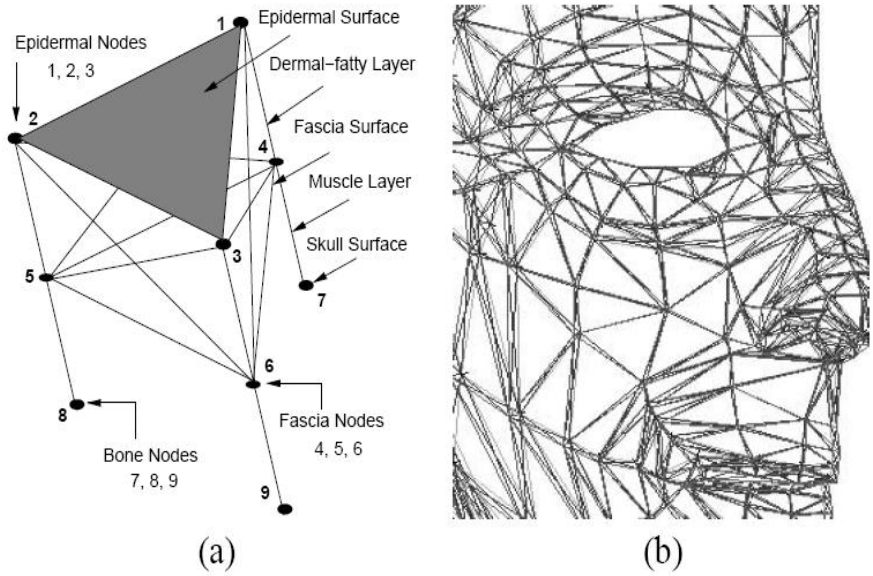


Figure 6.3: (a) Triangular skin tissue prism element.
(b) Close-up view of right side of an individual with conformed elements.

The tissue model is composed of triangular prism elements which match the triangles in the adapted facial mesh.

- The epidermal surface is defined by nodes 1, 2, and 3, which are connected by epidermal springs.
- The epidermis nodes are also connected by dermal-fatty layer springs to nodes 4, 5, and 6, which define the fascia surface. Fascia nodes are interconnected by fascia springs.
- They are also connected by muscle layer springs to skull surface nodes 7, 8, 9.

6.5 Discrete Deformable Models (DDMs)

The DDM is composed of a node-spring-node structure. It is a uni-axial finite element. The data structure for the node consists of:

1. the nodal mass: m_i
2. position: $x_i(t) = [x_i(t), y_i(t), z_i(t)]^t$
3. velocity: $v_i = dx_i / dt$
4. acceleration: $a_i = d^2 x_i / dt^2$

5. net nodal forces: $f_i^n(t)$

The data structure for the spring in this DDM consists of:

1. pointers to the head node *i* and the tail node *j* which the spring interconnects
2. the natural or rest length l_k of the spring
3. the spring stiffness c_k

6.6 Tissue Model Spring Forces (TMSFs)

Lee et al, employ a simplified tissue model (as seen in Fig. 6.3a). Each layer has its own stress-strain relationship c_j and the dermal-fatty layer uses biphasic springs (non-constant c_j)

They define a force spring *j* which exerts on node *i* as follows:

$$g_j = c_j(l_j - l_j^r)s_j$$

where:

l_j^r and $l_j = \| x_j - x_i \|$ are the rest and current lengths for spring j

$s_j = (x_j - x_i)/l_j$ is the spring direction vector for spring j

6.7 Linear Muscle Forces

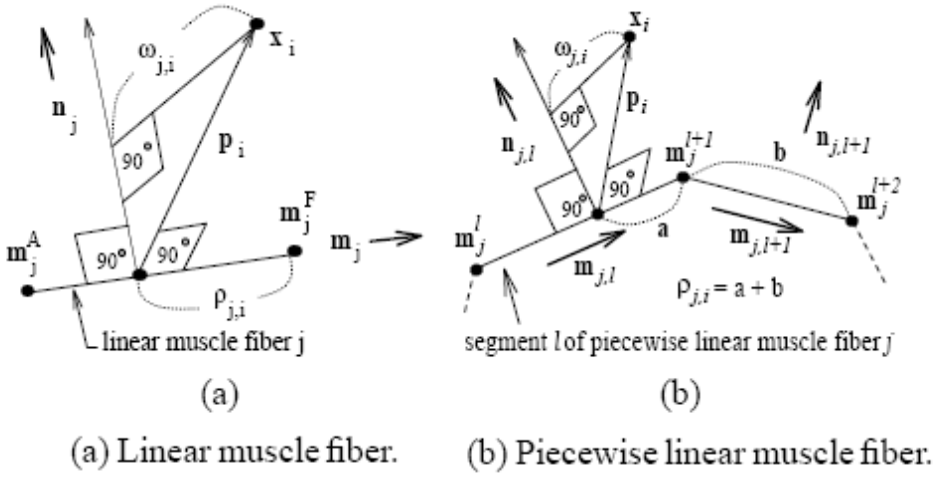
Despite the extra layers in this model, it acts in a similar fashion to the one proposed in [5]. They connect to the skin tissue by short elastic tendons at many fascia points and are fixed to the skeleton at only a few points

Contractions of muscles cause movement of the facial tissue. Face construction algorithm determines the nodes affected by each muscle in a pre computation step.

6.8 Applying muscle forces to the fascia nodes

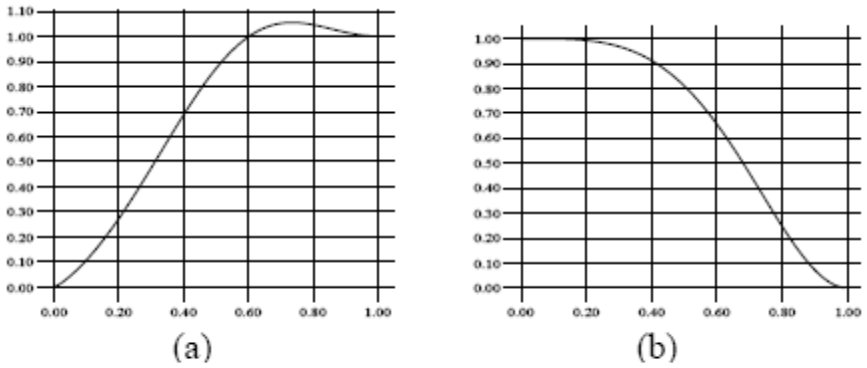
Lee et al calculate a force for each node by multiplying the muscle vector with a force

length scaling factor and a force width scaling factor.



(a) Linear muscle fiber. (b) Piecewise linear muscle fiber.

Figure 6.4



(a) Muscle force scaling function Θ_1 wrt $\varepsilon_{j,i}$, (b) Muscle force scaling function Θ_2 wrt $\omega_{j,i}/w_j$

Figure 6.5

Muscle *j* exerts a force *f* on node *i* as follows:

$$f_i^{\,j} = \Theta_1(\varepsilon_{j,i})\Theta_2(\omega_{j,i})m_j \tag{6.1}$$

where

Θ_1 scales the force according to the distance or length ratio $\varepsilon_{j,i}$,

d_j = length of muscle *j*,

$\varepsilon_{i,j} = \rho_{j,i}/d_j = ((m_j^F - x_i)\times m_j)/(\| m_j^A - m_j^F \|)$

Θ_2 scales the force according to the width ratio $\omega_{j,i}/\omega_j$

ω_j = width of muscle *j*, and

$\omega_{i,j} = \| p_j - (p_i \times n_j) n_j \|)$

m_j is the normalized muscle vector for muscle *j*

6.9 Piecewise Linear Muscles

As in [Kähler et al] other types of muscles (e.g. sphincter, sheet) build on the most basic form (linear).

Figure 4(b) illustrates two segments of an N-segment piecewise linear muscle j showing three nodes m_j^l, m_j^{l+1} and m_j^{l+2}. The unit vectors $\vec{m}_{j,l}$, $\vec{m}_{j,l+1}1$ and $\vec{n}_{j,l}$, $\vec{n}_{j,l+1}$ are parallel and normal to the segments, respectively. The figure indicates fascia node *i* at x_i, as well as the distance $\rho_{j,l+1} = a + b$, the width $\omega_{i,j}$, and the perpendicular vector p_i from fascia node *i* to the nearest segment of the muscle. The length ratio $\varepsilon_{j,i}$ for fascia node *i* in muscle fiber *j* is:

$$\varepsilon_{j,i} = \frac{(m_j^{l+1} - x_i) \cdot \vec{m}_{j,l} + \sum_{k=l+1}^{N} \left\| m_j^{k+1} - m_j^k \right\|}{\sum_{k=l+1}^{N} \left\| m_j^{k+1} - m_j^k \right\|} \tag{6.2}$$

6.10 Volume Preservation forces

The volume preservation force element e exerts on nodes i in element e is defined as:

$$q_i^e = k_1 (V^e - \tilde{V}^e) n_i^e + k_2 (p_i^e - \tilde{p}_i^e) \tag{6.3}$$

where:

$\tilde{V}^e$ and V^e = rest volumes for *e*

n_i^e = epidermal normal for epidermal node *i*

$\tilde{p}_i^e$ and p_i^e = rest and current nodal coordinates for node *i* w.r.t. the center of mass of e

k_1 and k_2 = force scaling constants

6.11 Skull Penetration Constraint Forces

The reasons for these forces have already been covered in the previous paper. The force S_i to penalize fascia node *i* during motion is:

$$s_i = \begin{cases} -(f_i^n \cdot n_i)n_i & if\ f_i^n \cdot n_i < 0 \\ 0 & otherwise \end{cases} \qquad (6.4)$$

where

f_i^n = net force on fascia node *i*

n_i = nodal normal of node *i*

6.12 Equations of Motion for Tissue model

This obeys Newton's law of motion for tissue model response to forces, which results in 2nd order Equations relating to position, velocity & acceleration of nodal forces

Equation for node *i:*

$$m_i \frac{d^2 x_i}{dt^2} + \gamma_i \frac{dx_i}{dt} + \tilde{g}_i + \tilde{q}_i + \tilde{s}_i + \tilde{h}_i = f_i \qquad (6.5)$$

where:

m_i = nodal mass

γ_i = dampening coefficient

$\tilde{g}_i$ = is the total spring force at node *i*

$\tilde{q}_i$ = total volume preservation force at node *i*

$\tilde{s}_i$ = total skull penetration force at node *i*

$\tilde{h}_i$ = total nodal restoration force at node *i*

$\tilde{f}_i$ = total applied muscle force at node *i*

At time ***t:***

Acceleration of node *i* = *(net nodal Force at time t) / (mass of node i)*

The Velocity and distance/length calculated through integration and extended to calculate at next time step $t + \Delta t$:

$$a_i^t = \frac{1}{m_i}\left(f_i^t - \gamma_i v_i^t - \tilde{g}_i^t - \tilde{q}_i^t - \tilde{s}_i^t - \tilde{h}_i^t\right) \quad (6.6)$$

$$v_i^{t+\Delta t} = v_i^t + \Delta t a_i^t \quad (6.7)$$

$$x_i^{t+\Delta t} = x_i^t + \Delta t v_i^{t+\Delta t} \quad (6\#.8)$$

For their tests, Lee et al used the following as default parameters:

Mass (m)	Time step (Δt)	Damping (γ)
0.5	0.01	30

	Epid	Derm-fat 1	Derm-fat 2	Fascia	Muscle
c	60	30	70	80	10

6.13 Eyes, Teeth and Other artifacts

The eyes are independent of the muscle-system, and are kinematically operated. Teeth were created using NURBS.

7 My Implementation

While Muscle based approaches have their advantages over parametric approaches to facial animation, there is still some work involved in determining morph targets for muscles, be it for a generic model, as used in **[5]** and **[6]** or simply for a single model. Some manual specification would still be required. Furthermore, computation overhead is not as favorable as with non-physically based approaches.

With parametric approaches like **[2]** on the other hand, it is faster to interpolate between morph targets. Yet this approach still suffers the problem of being specific to an individual model. Looking in a big picture sense, several unique faces would then need to

be defined individually, which basically implies $N \bullet M$ sets of data, where N is the number of specific models and M is the number of Morph Targets to be used for each Face Model.

Looking at the trade offs between the muscle-based and parametric approaches; some manual input is still going to be required for defining the morph targets, to achieve realistic Facial Animation.

My approach is a parametric one which utilizes some of the basic idea in [5], whereby a generic model is used as a source of animation for multiple models. It will focus mainly on Speech animation (Animation of emotions being secondary work), so the majority of morph targets will be visemes. Though not physically based as in [5], vertex positioning for the morph targets of target models will be defined by a generic model, using vertex offset technique. Most of the manual input in this approach would apply to the generic model, thus making several models a less involved process. The catch to this though, is that the models must all be of equal polygon count for this approach to work.

7.2 General Approach

The general approach is to define all morph targets and their (absolute) model space vertex positions, for a generic model. Once this has been established, we would then be able to use this set of data to define our vertex offsets.

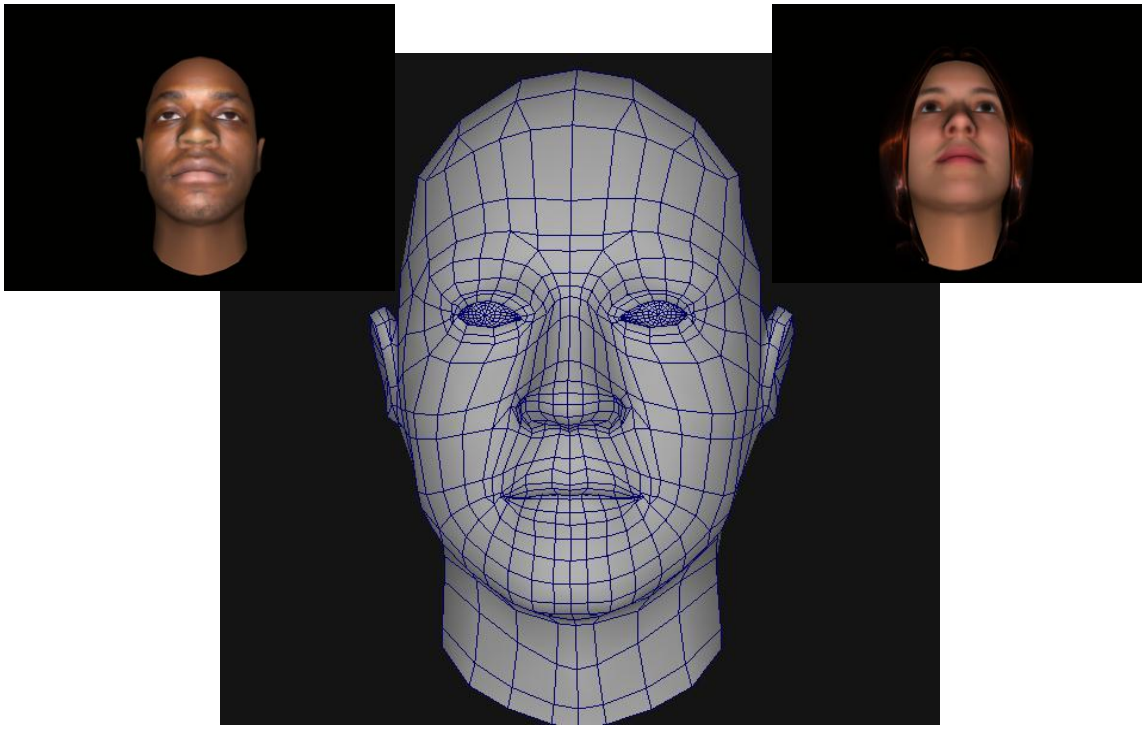


Figure 7.1: Generic Model (Middle) and Specific Models (Left & Right)

Transitioning from one morph target to the next could be determined using a number of interpolation techniques. The most time-efficient option would be linear interpolation, but it is yet to be determined how smooth an animation this would produce with this facial animation technique. Another option though not as efficient would be cubic-spline interpolation, but this would only be required depending on how smooth the linear approach appears to be.

Since we are focusing mainly on speech animation, accompanying the interpolation/animation would be a mechanism for generating or recording speech and having this audio played in synch with the speech animation. For robustness, we would need a way to convert text to speech, speech to text. To go even a step further, user recorded voices could be used to synch visemes generated determined by the audio input.

7.3 Tools

Some questions still need to be answered as to how these models are to be created and

whether they could be done so in a time efficient manner. And since we will be focusing on Speech animation, it would be nice to have an engine to facilitate creating a dialogue, also with little time overhead. If we were to generate all the visemes that make up speech animation, even for a single sentence, manually, this would be too time consuming. To answer these questions, a number of tools were required and leveraged to make this a possible. These tools are treated as integral components for the overall animation system.
The tools used are:

1. FaceGen Modeller (3.1)
2. 3ds Max / Panda Exporter
3. Microsoft SAPI (5.1)
4. Microsoft DirectX / Visual Studio

7.3.1 FaceGen Modeller (3.1)

FaceGen is a stand-alone parametric face modeling middleware for Windows (XP or Vista). For artists, it is fast and flexible way for creating realistic 3D faces, exportable in a number of popular formats. For developers, it is a powerful and easy to use tool for (re)creating realistic 3D faces.

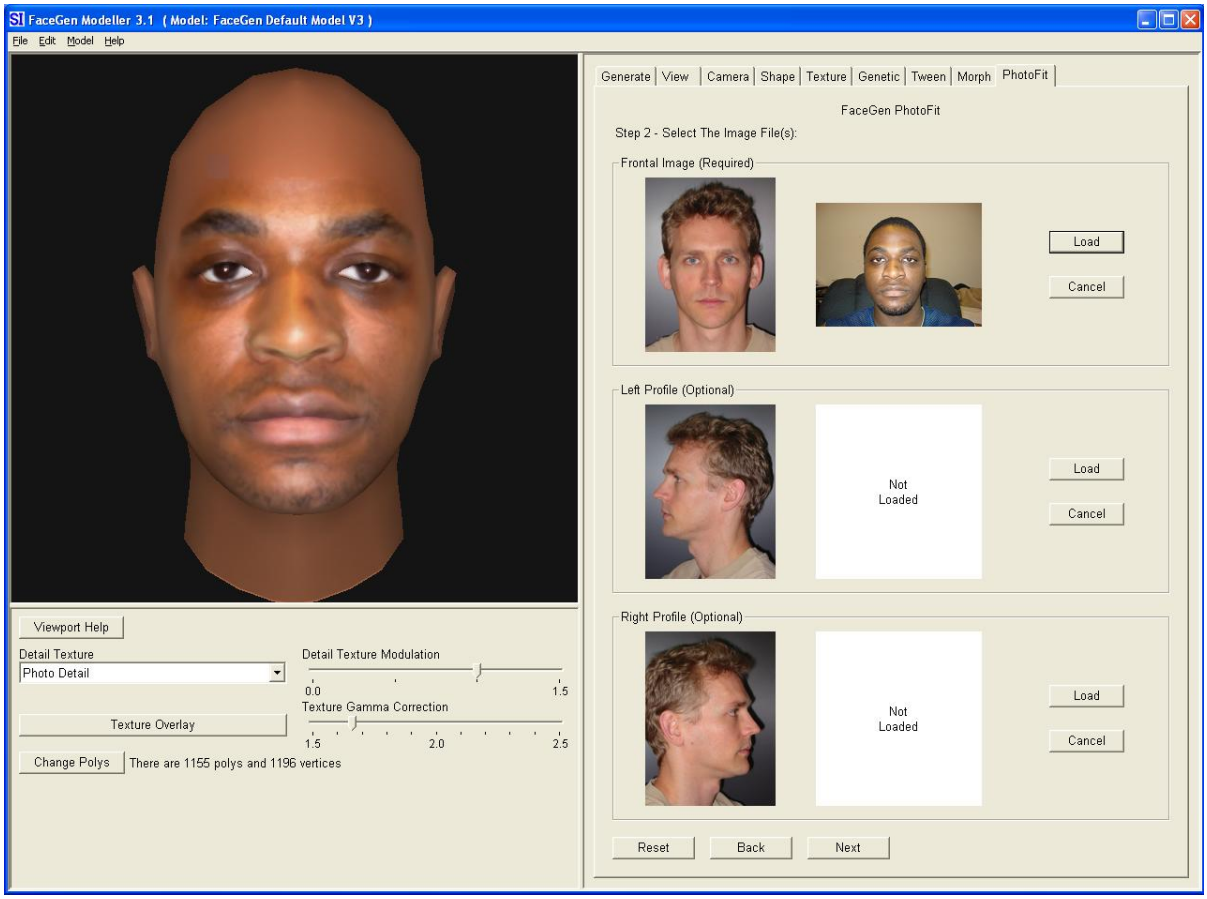


Figure 7.2 Creating a Model using PhotoFit

There are 2 ways to create faces. They could either be generated at random, or by *PhotoFit*. As the name suggests, *PhotoFit* is a feature which offers the option to customize your own model, using source photoraph(s).

To use *PhotoFit*, you will need to provide a front view photograph (mug shot) and two optional side view photos, as shown in Figure 7.2. You would also have the option to define 11 feature points: eyes, cheekbones, mouth, jaw, nose and chin as shown in figure 7.3

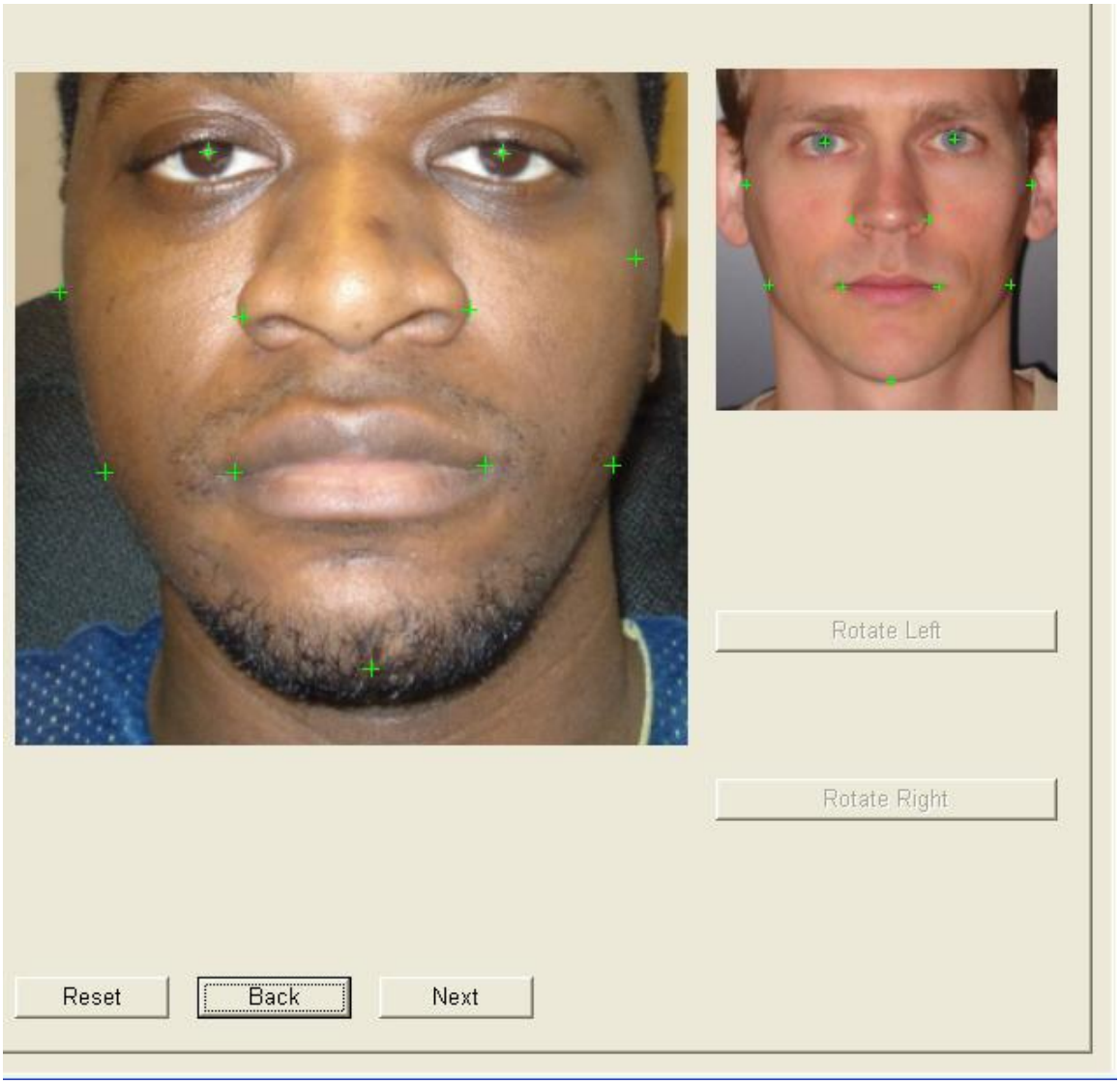


Figure 7.3 Creating a Model using PhotoFit

Once these are defined, the next step is to process the image to create a 3D representation

The process takes about 10 to 15 minutes and the result is a set of interchangeable model parts of 3 different qualities:

- Low Resolution - 790 Quads & 775 Vertices
- Medium Resolution - 1476 Quads & 1439 Vertices
- High Resolution - 5746 Quads & 5832 Vertices

For best results, the Low resolution face mesh will be used. One advantage of using this software is that it uses an equal polygon count for all models. Further fine tuning can also

be performed on the model once the processing is completed. The resulting model can then be save in FaceGen's native **.fg** format, for future use.

7.3.2 3ds Max / Panda Exporter

In addition to the native format, FaceGen models can also be exported to various standard formats, including wavefront .obj. The format of choice for the animation engine is DirectX, so we would need 3ds Max with an exporter plug-in to convert .obj files to .x. The exporting to .x file process is where Panda Exporter comes into play.

7.3.3 Microsft SAPI (5.1)

Microsoft's Speech API will be used for converting Text-to-speech and Voice Recognition. This is done through a series of events which will be discussed in further detail later on

7.3.3 Microsoft DirectX / Visual Studio

The Animation Engine Application as a whole is written in C++ in the Visual Studio environment. This is seamlessly integrated with DirectX and SAPI to as they are all part of the Microsoft family

7.4 Overview

The general flow of the Animation process is illustrated in Figure 7.4. It outlines the aforementioned components, showing how each would relate to each other on a grand scheme.

The SAPI Engine determines listens for events to determine whether information passed to it is text or audio. In the case of audio input the engine first performs a speech-to-text conversion before converting the text to speech by one of the Microsoft voices. Use of the Microsoft Voices is necessary to ensure proper synching of audio with the speech animation.

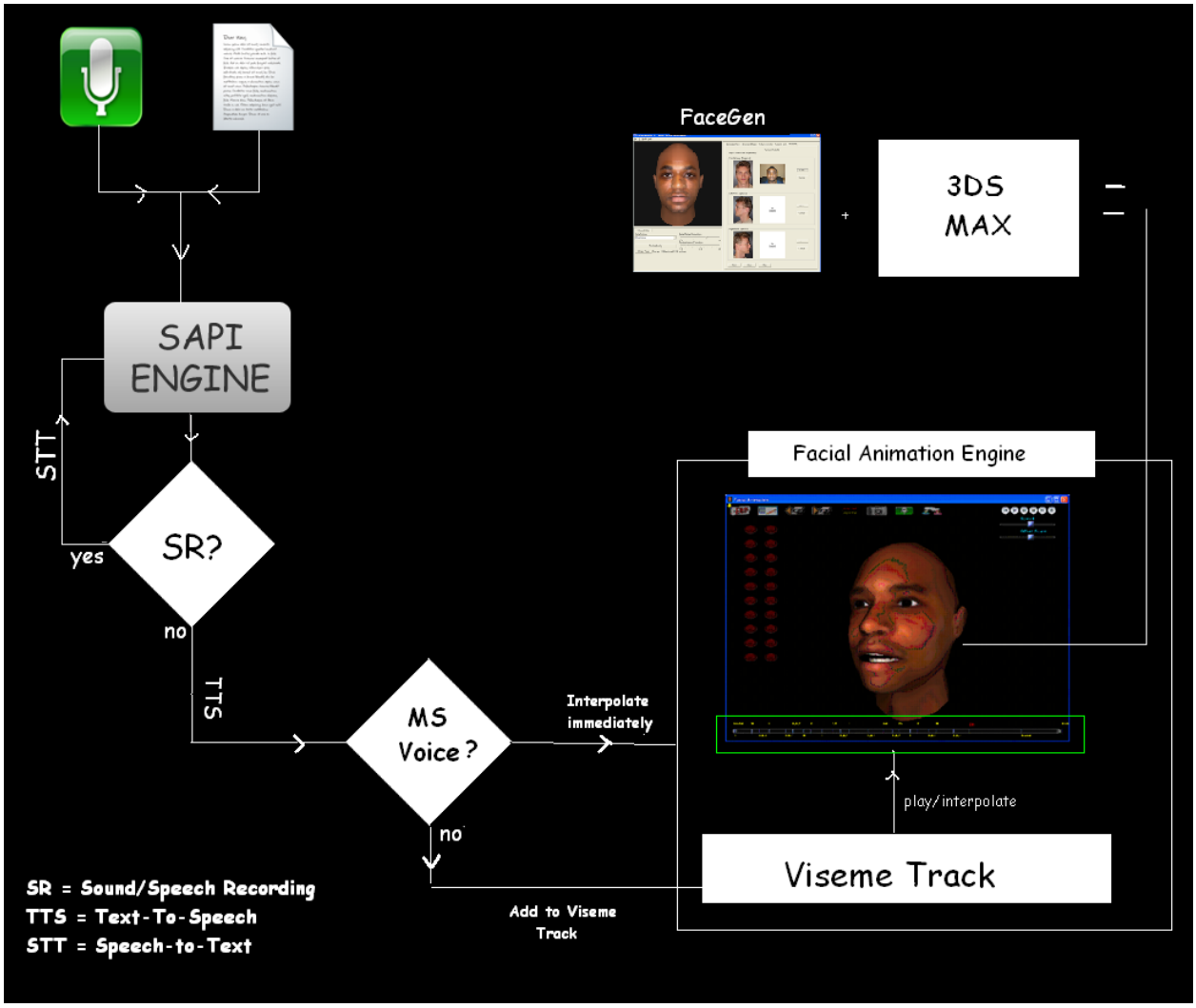


Figure 7.4

Model Creation, performed offline is the model creation. It is done before the animation application is run. Upon application initialization, a model is loaded via an XML script into the scene of the Facial Animation engine.

7.5 Facial Animation Engine

To get a better understanding of how components are used it is important to take a closer look at the animation engine. It is essentially comprised of 3 major components:

- SkinManger
- Speech Wrapper
- VisemeTrack

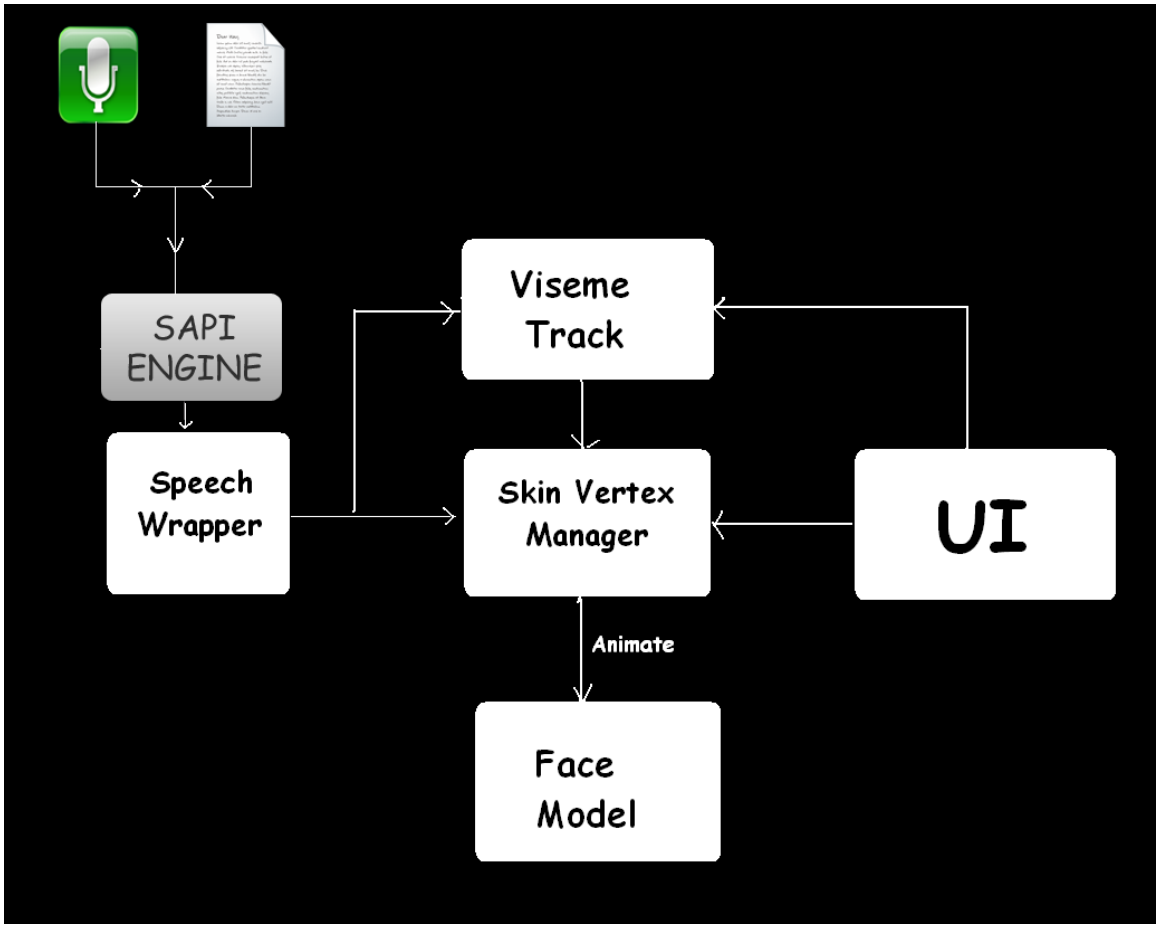


Figure 7.5: A Closer Look at the Facial Animation Engine

7.6 Skin Manager

All objects created in FaceGen are made up of eye, teeth, tongue and skin meshes. The skin manager, the core of the animation focuses on manipulating the skin mesh of the face model. It has access to the vertex buffer of the skin mesh and control over the vertices that define the skin mesh. All animation via interpolation between Visemes and other Morph Targets, are performed here. As it was mentioned earlier, this is a parametric approach which makes use of a vertex data of a source generic model. But it has yet to be discussed, the process of defining this source data.

7.6.1 Defining Generic Model Vertex Data

In total, FaceGen has 38 Morph Targets, but since the focus is on speech animation, half of these can be ignored.

Defining the generic data is a one time process. 17 visemes are created to define the 16 viseme morphs and a neutral state. These morph targets can be previewed under the morph Tab as shown in figure 7.6 below

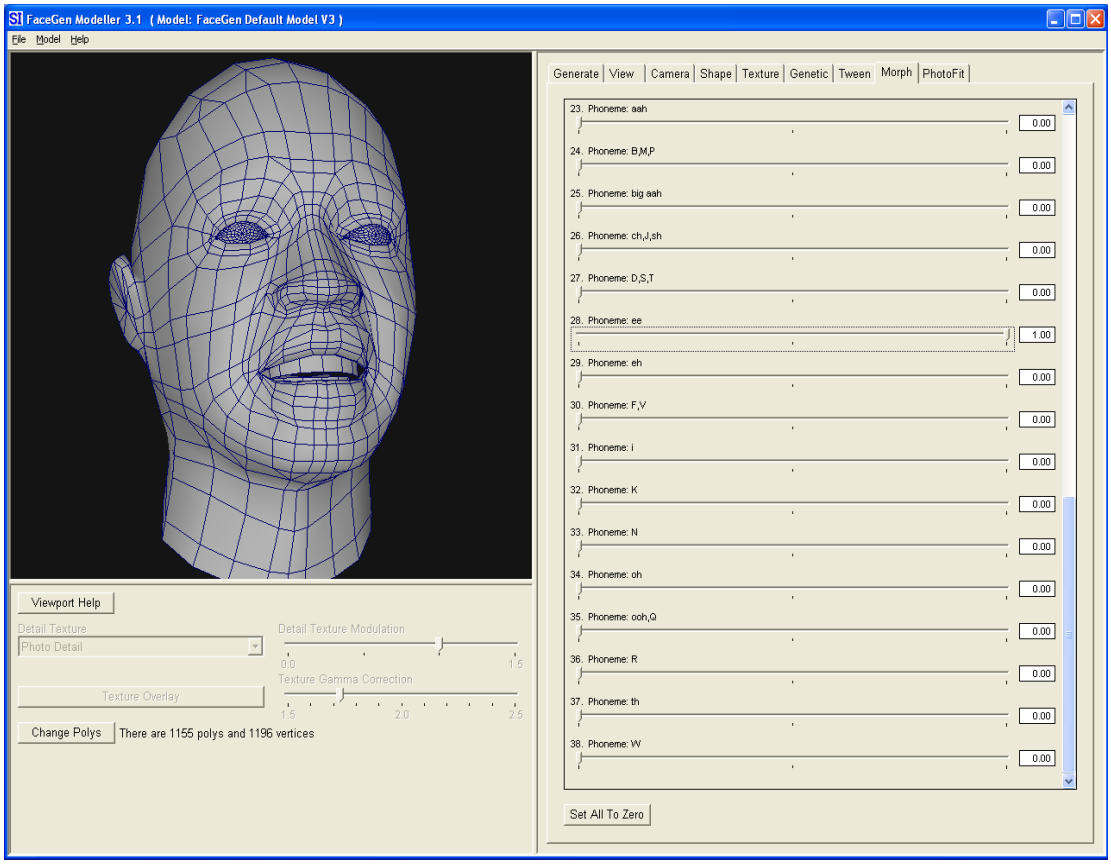


Figure 7.6: Morph (Viseme) Targets

These visemes are exported to .obj files which each contain vertex data of the different meshes that make up the model. As mentioned earlier, we only need data for the skin mesh. For convenience and also to save space, all the Skin mesh vertex data for each viseme is compiled into one data file. It can be thought of as a collection of tables, each named according to the viseme file they originated from. These tables contain per row, information on a vertex index number, its position and normal in model space. With this data it is now possible to interpolate between targets of the generic model.

But how does one apply this to a specific model. With the generic model's vertex data,

the skin manager has enough information to generate viseme offsets. This can be done on initialization of the Facial Animation Application. The offset for a given vertex is determined by computing the vector defined by its position for a given viseme, and its position on the neutral target.

To compute a vertex offsets **(*V*):**

$$V_{G,p}[i] = \left(V_{G,p}[i] - V_{G,N}[i]\right) \qquad (7.1)$$

Where:

i:	a given vertex index
G:	generic Model
p:	one of the 16 viseme targets
N:	Neutral viseme

In order to keep track of all the data, a collection of **Viseme** objects is required. A Viseme object is comprised of all the vertices that make up the skin mesh of a morph target. Also included in this object, are generic model offsets determined using equation (7.1), and normals:

```
class Viseme
{
   std::map<UINT, D3DXVECTOR3>  m_v3DVerts; // Model Space Position
   std::map<UINT, D3DXVECTOR3>  m_vNeutralOffset; //from Neutral Viseme
   std::map<UINT, D3DXVECTOR3>  m_vNormals;
public:
   Viseme ();
   ~Viseme ();
};
```

The skin manager stores all of objects in a Viseme Container which is basically a table of all of Viseme Objects. In addition to the 16 viseme targets + the neutral target (of the generic model), we also need to include a viseme object for the specific model. This additional viseme object represents the Neutral state of the specific model.

7.6.2 Determining vertex offsets of the Specific Model

Viseme Targets for the specific models are determined by applying the vertex offsets computed in (7.1) to the vertices of the specific model. It should be pointed out that these specific models can be thought of as the Neutral morph target for a given specific model

The equation to compute a Viseme target **(V_S)** for a specific model**:**

$$V_{s,p}[i] = V_{s,N}[i] + (V_{G,p}[i] - V_{G,N}[i]) \qquad (7.2)$$

Where:

i:	a given vertex index
s:	specific model
G:	generic Model
p:	one of the 16 viseme targets
N:	Neutral viseme

This method works seamlessly with models of equal polygon count and vertex indexing. We can go one step further and include an offset modifier (0.0 – 1.0), to control degree of articulation:

$$V_{s,p}[i] = V_{s,N}[i] + (V_{G,p}[i] - V_{G,N}[i]) \bullet M \qquad (7.3)$$

Where:

***M*:**	Offset Modifier

The lower value of *M*, the less articulation occurs. By default, the value of *M* is 1.0 which reduces equation (7.3) back to (7.2)

7.6.3 Interpolation Technique used

Calculating vertex positions between 2 Visemes p and q is done by linear interpolation:

$$V_{sx}[i] = \mathbf{Lerp}(V_{sp}[i], V_{sq}[i], t)$$

$$= (1-t) \bullet V_{sp}[i] + t \bullet V_{sq}[i] \qquad (7.4)$$

Where:

i:	a given vertex index
s:	specific model
t:	normalized time between visemes
p:	one of the 16 viseme targets
q:	also one of the 16 viseme targets

$V_{sp[i]}$ and $V_{sq[i]}$ are computed using equation 7.3. Linear interpolation turns out to be the best option in this case as the resulting animation appeared as smooth as some higher order interpolation techniques, such as cubic spline interpolation.

7.7 Speech Wrapper

The Speech Wrapper leverages functionality of the speech API, Microsoft SAPI 5.1. It offers a variety of options, some of which are beyond the scope of this project, but the 2 main areas of interest for integration into the Facial Animation Engine, are the Text-to-Speech (TTS) and Speech-To-Text (our Speech Recoding (SR). For both, a series of events are used to determine Visemes parameters, for animation. In order to capture these events, three interfaces will be required: ISPVoice, ISpRecognizer and ISpRecoContext.

7.7.1 ISpVoice

Text synthesis operations are performed using this interface. Sources could be either a text string, text file, or an audio file. These operations may be performed synchronously or asynchronously.

This can also be used to modify the state of the voice (pitch, rate, volume etc), either in via SAPI XML tags in input text, or in real time using the various *Set* methods ISPVoice provides. Our facial Animation engine provides an interface to leverage these functions

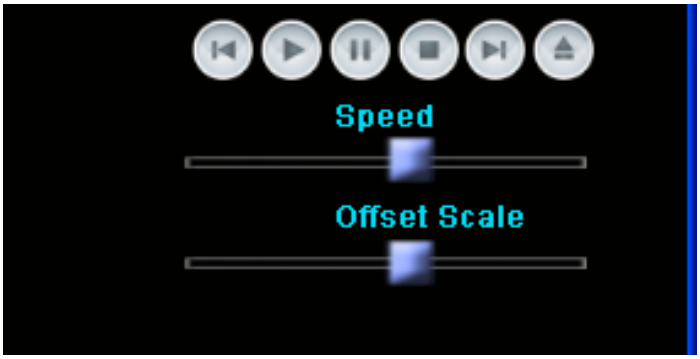


Figure 7.7 SpeechWrapper User Interface

ISpVoice inherits from ISpEventSource and ISPNotifySource, methods to specify how notifications are received and, which events should trigger notifications for the SpeechWrapper to take action. For this project, the events of interest are as follows:

SPEI_START_INPUT_STREAM: On notification, the SpeechWrapper notifies the Skin Manager to begin animation

SPEI_END_INPUT_STREAM: On notification, the SpeechWrapper notifies the Skin Manager to stop animation

SPEI_WORD_BOUNDARY: Useful for displaying text on screen one word at a time

SPEI_VISEME: When text is spoken via the **ISPVoice::Speak** method, a series of events is triggered, including all the visemes that make up the input text. It should be noted that the SAPI, assumes a set of 21 visemes. In order to make use of them effectively, an intermediate conversion is required to map any one of these 21 visemes to the 16 generated in FaceGen. Included in the viseme events are the current Viseme, duration of the current viseme and the next visemes.

With these 3 pieces of information, interpolation may be performed immediately. In fact, since we're using the default TTS Voices, it makes most sense to perform immediate interpolation, concurrent with the audio playback. The Speech wrapper passes on the current Viseme information to the Skin Manager:

```
SkinSystem::SetCurrentVisemeInfo(currentVisID, fDuration, nextVisID)
```

The downside to immediate interpolation is that there is not enough information provided for Phoneme Reduction. The Facial animation provides a Viseme Track that aims to resolve the problem.

7.7.2 ISpRecognizer & ISpRecoContext

The ISpRecognizer is used to control aspects of the Speech Recognition Engine. Attached to it is a recognition context (ISpRecoContext), through which we will be able to control and handle recognition events. Like the ISpVoice object, we set the events we are interested in for the recognition context, using the *ISpRecoContext::SetInterest* method.
e.g.

```
const ULONGLONG ullInterest = SPFEI(SPEI_RECOGNITION)| SPFEI(SPEI_HYPOTHESIS) |
SPFEI(SPEI_SR_AUDIO_LEVEL);

//Set intereted in the Events defined above
m_cpRecoCtxt->SetInterest(ullInterest, ullInterest);
```

Events of interest (from the recognition context) are:
SPEI_RECOGNITION: On notification, this event contains a **RecoResult** object which may be converted to text.
SPEI_FALSE_RECOGNITION: On notification, do nothing

SPEI_SR_AUDIO_LEVEL: Audio levels ranging from 0 – 100 can be gathered for each phoneme event.

7.7.3 Recognition Accuracy

Voice training sessions are required for improved sound recognition results. These training sessions are available as part of the Microsoft Speech SDK

7.8 Viseme Track

The viseme track is a container made up of markers which represent visemes interpreted by the SpeechWrapper. Markers or Marker objects consist of the following data:

1. ID: unsigned integer ranging from (0-16), representing visemes
2. Duration: time in seconds representing the difference between the start time of the current viseme to be entered, and the next viseme.
3. UI render components: to be displayed on screen (as shown in Fig 7.8). They are displayed as bars and text, showing the Phonemes a given marker's viseme ID represents

Each time we convert Text-To-Speech, Viseme events (SPEI_VISEME) are triggered, during which markers may also be placed on the track. For every new entry, the track automatically rescales according to the updated total time, computed by the total of the duration values of all the markers.

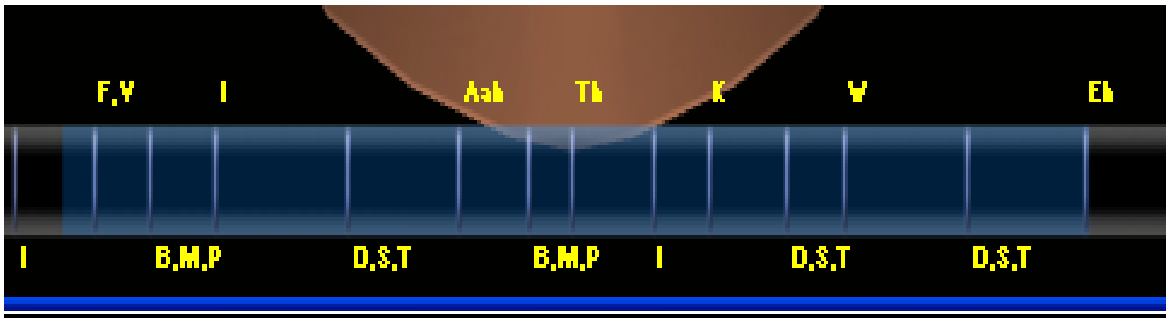


Figure 7.8 Viseme Track Segment

Markers may also be manually modified or deleted, opening up the possibility for manual phoneme reduction, and possibly lip syncing with human voice playback

7.9 User Interface

Figure 7.9 UI Interface

Figure 7.9 illustrates the user interface for the entire application. Users can toggle face models, play and replay recorded input, load text files for text-to-speech conversion, toggle voice recognition mode. Also included are the options edit the viseme track or control the offset scale introduced in equation 7.3.

8 Summary

We investigated a couple of different animation techniques, identified areas of concern in

each case and examined a number of relatively new approaches which apparently solved a number of the problems brought about by its predecessors.
Whether these techniques are practical for use in real time gaming remains to be seen, but they are innovative approaches nonetheless which could be applied in other areas such as animated movies or video game cut-scenes.

My implementation was introduced to address the issues and concerns of both muscle based and parametric animation techniques, and the result proved to be very positive, with lower computational overhead than muscle based approaches and an average frame rate exceeded 200 FPS. Like the new batch of physically based implementations, my parametric technique is applicable to multiple faces using an adaptation technique.

There are however, some limitations. Specific models must be same resolution as the generic model for adaptation to be successful. Provided that a common modeling package, is used, this should not be an issue

9 Future Work

Future implementation include more focus on animation of emotion, currently models have 3 non viseme morph targets: blink and eyebrow raising both of which currently occur at random, and a closed mouth smile, which can only be triggered manually Currently only smile, blink, eyebrow raise. Enhancement would involve the addition of more expressions and tying some of them to with speech, to visually reflect a change in tone.

Another area of interest is the automation of phoneme reduction, using the data gathered by the viseme track. Manual reduction is currently possible, but this quickly becomes an arduous task as the amount of speech increases

Currently, the engine has only been thoroughly tested with low resolution, and it is yet to be seen if using higher resolution models would result a significant dip in frame rate.

References

[1] Selma Rizvic, Zikrija Avdagic. *Phoneme Reduction in Automated Speech for Computer Animation.* 2004

[2] RIZVIC S, AVDAGIC Z. 2003. *Model for Speech Animation Based on MaxScript Scripting Language,* IKT Sarajevo, 2003.

[3] FLEMMING B, DOBBS D. 1999. *Animating Facial Featuresand Expressions,* Charles River Media INC, Rockland Massachussets

[4] MADSEN R. 1969. *Animated film: Concepts, Methods, Uses.* Interland, New York

[5] Y. Zhang, T. Sim, C. L.Tan. 2004. *Rapid Modeling of 3D Faces for Animation Using an Efficient Adaptation Algorithm.*
<http://www.comp.nus.edu.sg/~face/paper/GRAPHITE2004.pdf>

[6] K. Kähler, J. Haber , H. P. Seidel. 2001. *Geometry-based Muscle Modeling for Facial Animation* <http://www.graphicsinterface.org/proceedings/2001/103/file103-1.pdf>

[7] Yuencheng Lee1, Demetri Terzopoulos, and Keith Waters. 2001. *Realistic Modeling for Facial Animation.* <http://www.graphicsinterface.org/proceedings/2001/103/file103-1.pdf>

[8] Wikipedia. April 16, 2006. *Verlet Integration*. <http://en.wikipedia.org/wiki/Verlet>