# AN AUTOMATED RIGGING SYSTEM FOR FACIAL ANIMATION

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Jacobo Bibliowicz

January 2005

## ABSTRACT

We present a system for the automated rigging of human face models, providing a significant time savings for this arduous task. Our system takes advantage of previous work which deforms a reference facial surface to conform to new face models. In particular, we have matched our reference model to digitized human faces. We parameterize the construction of the reference rig on the surface topology of the reference model; thus, the rig can be procedurally reconstructed onto deformed copies of the model.

The animation parameters of our rig are based on the expressive muscles of the face. We present a novel skin/muscle model, based on wire deformers and point constraints, to implement this parameterization. The model supports any number of muscle contractions about a particular skin patch, yet gives the artist ultimate control over the shape of the deformation.

Finally, we present a flexible environment for loading, rigging, and animating new models quickly and effectively.

# Biographical Sketch

Jacobo Bibliowicz, known as Jacky to his friends and family, is proud of his Jewish and Colombian heritage. He was born in Bogotá, Colombia, in May of 1976, and lived there until he completed his high school degree at Colegio Nueva Granada. He moved to Ithaca, NY, to obtain his undergraduate degree in Applied and Engineering Physics at Cornell University, and has resided there ever since. Before joining the ranks of the Program of Computer Graphics, he worked at a small branch of Autodesk, Inc., as a software engineer.

In addition to his academic life, Jacky enjoys having fun and trying out new and novel projects. In his ten years at Ithaca, he directed the world premiere of *Una tal Raquel*, DJ'ed the radio show *Ritmo Latino* on WICB, and volunteered as a skier at Greek Peak Sports for the Disabled. He also enjoys riding his bike, playing squash, dancing salsa, tasting wine, and traveling.

After completing his Masters degree at the PCG, he will move to Canada to begin a Ph.D. at the University of Toronto.

He will miss Ithaca very much.

# Acknowledgements

This work would not have been possible without the encouragement and support of my advisor, Don Greenberg. His excitement over my work was contagious, and his enthusiasm over all things graphics related is laudable. I look forward to many future meetings with Don as a PCG alumnus.

The time spent at the Program of Computer Graphics was enjoyable due to the charisma, friendliness, and support of its faculty and staff. I thank my minor advisor, Kavita Bala, for lending a helpful hand and a friendly ear when I needed them, and for ensuring that my work is presented in the best possible way. Steve M. and Fabio provided invaluable comments on some of the chapters in this thesis and answered some of my toughest questions. Bruce, Steve W., and Jim were always available to help me and to chat whenever I popped into their office. Maintaining a computer lab that runs smoothly is no small task; Hurf and Martin provided first-rate support for all my computing needs. Linda was crucial in ensuring that my work did not get buried in Don's in box, Peggy entertained me with many interesting conversations about politics, upholstery, and skiing, and Mary was not too upset about the spilled popcorn.

Life at the PCG was not always work; lunch breaks and fun times were shared with other students. Ryan, Jeremy, and Spf ensured that our transition into the

lab was a smooth one. Adam proved himself a great office mate and a strong athlete. Mike provided great entertainment with hallway sports and "DyD," while Will entertained us with his juggling and candy consumption. Henry is just an overall great guy, fun to talk and listen to, and his office mate Vikash is the pillar of calmness and the guru of all thing Linux and Subaru. John Mollis was the connection to A&EP at Cornell, and Hong Song completed the club of international students. To the students who remain: Jeremiah has been a great office mate, providing artistic tips in exchange for technical ones. Jeff B. is the Houston connection and a good buddy, and Jeff W. provided hockey tickets and rides home at key times. And Nasheet and Mark will quietly take over the lab when nobody is looking. That is, of course, unless the Ph.D. students (Jon, Andy, Ganesh, Adam, Milos, Piti) do it first.

My mom and dad have been strong supporters of all my endeavors, each in their own way. I will never be able to thank them enough for calling me so often (even when I don't want to talk) and for inviting me to visit far away places. I love them very much. My sister Yael is also a source of strength and inspiration, as well as a true friend. I wish all three of them could have seen me present this work. I also want to thank the rest of my family for their love and support—they are too many to mention in these short pages.

My love goes out to my girlfriend Juyun. She has mothered me over the last few weeks and helped me jump over the stumbling blocks I lay down for myself. Rocío is a fun and cheerful friend and a great President. Her family adopted me for the Christmas holiday when I was not able to visit my own. Mauricio "Guasacaca" has been a my partner in squash and in crime. Pati cheered me up when I was down and increased my consumption of bubble tea. Miguel and Jenny Gómez invited

me into their home during my last few months in Ithaca, for which I am eternally grateful. I partied Colombian style with Camilo, Miguel, Hernando, Luz Marina, Juliana, Tomás, León, Juan, and many others.

In the ten years I have spent in Ithaca, I have met many wonderful people who I do not have space to mention, but who will never be forgotten.

This work would not have been possible without generous software donations from Alias, Inc. Funding was provided by Program of Computer Graphics, the Department of Architecture, and the National Science Foundation Grant CCF-0205438.

# Chapter 1

# Introduction

Recent years have seen an explosion of 3D computer generated animation. The entertainment industry has played a crucial role, pushing computer graphics technology to the limit. Studios producing computer animated films are releasing at least one movie per year, while digital characters are becoming commonplace in traditional films. Animation is an important ingredient in computer games as well, and cinematic shorts are becoming more commonplace. The public is enjoying it: seven of the ten top grossing films in the United States[1] have scenes with some form of computer animation, and the computer gaming industry is one of the fastest growing industries of recent times.

This high demand has resulted in a need for improving the efficiency of the computer animation process. Although one should not truncate the time spent in the initial artistic process, such as the design and construction of appealing characters, there are several technical steps in the animation pipeline that offer the opportunity of automation.

[1] According to figures provided in November, 2004, by Exhibitor Relations Co, Inc. [ERC].

One such step is the *rigging* step. Rigging is the process of defining and implementing the possible motions of a virtual character and providing controls to execute them, much like tying strings onto a marionette. This is a complex and time-consuming procedure requiring both artistic and technical savvy. However, when we restrict ourselves to a particular type of character—say, human beings—then every new character will have the same basic motions. This correspondence provides an opportunity for automating the rigging process. Such an automated system must be flexible to account for variations in the characters' physical characteristics, such as height and weight.

In practice, this problem is more complex than it seems, especially when dealing with the human face; the most expressive part of the body, communicating both language and emotion. Human beings have developed an acute sensitivity to facial expressions and artifacts in facial movement are immediately noticeable. Additionally, although humans all share the same underlying facial structure, the range of face types is immense.

Current facial animation techniques, discussed in the following chapters, attempt to overcome some of these issues. For example, parameterized models provide ready-made, configurable and animatable faces, but the number of characters which can be represented are limited by the number of conformal parameters. Morphing systems, on the other hand, can represent virtually any character, but they depend on facial expression libraries which must be built for each character. Additionally, these popular morphing algorithms do not simulate the motions in the face correctly.

In this work, we present a system for automating the rigging process of human faces, allowing the artist to immediately concentrate on the animation phase.

Our approach is to parameterize the rig construction on the surface topology of a reference model which is optimized to support the deformations of a human face. This model can then be deformed to match new faces using previously developed techniques. The reference rig is then easily reconstructed onto deformed copies of the reference model. The animation parameters of our rig are based on the underlying expressive musculature of the face. To implement the rig, we have developed a novel, procedural skin/muscle model which supports any number of muscle contractions about a particular region of skin. This model is flexible in that it allows the artist to control the final shape of the skin deformation. Additionally, we present a flexible environment for animating new models quickly and effectively.

The remainder of this thesis is organized as follows: Chapter 2 provides background information on the structure and movement of the human face, the perception of emotion, and the computer animation pipeline. Related work is covered in Chapter 3. Chapter 4 provides a description of the reference head model and rig, also developing the skin/muscle model. The procedure for deforming the reference model to match scanned geometry is the subject of Chapter 5, while Chapter 6 describes the final overall system, including the animation user interface. Conclusions and future work are provided in Chapter 7.

# Chapter 4

# A Generic Face Model and Rig

The goal of this research is to speed up the rigging process of a human face by applying a predefined rig to new geometric face models. We accomplish this by deforming a reference model and rig to match models generated by a facial digitizer. The reference model is expressive, capable of a large number of facial movements and emotions, and flexible, working in a large number of geometric configurations.

We have chosen muscle contractions as the driving parameters for the reference rig since, as explained in Chapter 2, they define the basic motions of the human face. Additionally, other, more intuitive interfaces can be constructed on top of these low-level controls.

To implement the rig, we have developed a novel skin/muscle model based on wire deformers and point constraints. The wire deformers intuitively act as skin attachment points for the muscles and the point constraints average out the skin displacement due to various muscle contractions. Furthermore, the point constraint weights allow the artist to fine tune the shape of the deformation, providing more control over the appearance of the face.

This chapter describes the generic model and rig in detail. We first cover

the basic concepts required to understand the deformation chain employed by the reference rig in Section 4.1. In Section 4.2, we introduce the skin/muscle model developed for the rig, and we conclude with a detailed description of the reference model in Section 4.3.

## 4.1 Fundamental Concepts

### 4.1.1 Constraints

Computerized surface models specify the position of their vertices with respect to an arbitrary origin, also known as the *pivot* point. By changing the position and orientation of the pivot coordinate system, the entire surface is repositioned within the scene. A system of hierarchical transforms works by applying the transform chain to the pivot point, thus altering the locations of surfaces.

However, it is sometimes desirable to constrain the behavior of the pivot point's coordinate system. For example, a rig can require the animator to specify the orientation on an object while constraining both eyes to match this orientation. Another example constrains the orientation of the eyes to point toward the control object. These control systems are illustrated in Figures 4.1 and 4.2. In both cases, the eye shapes are referred to as *constrained* objects and the controlling object is called the *target* object.

Our generic rig uses three types of constraints.

**Point Constraint:** Forces the position of the constrained object's pivot to match the position of the target object's pivot. The constrained object's orientation remains unaffected.

**Orientation Constraint:** Matches the orientation of the constrained object's co-

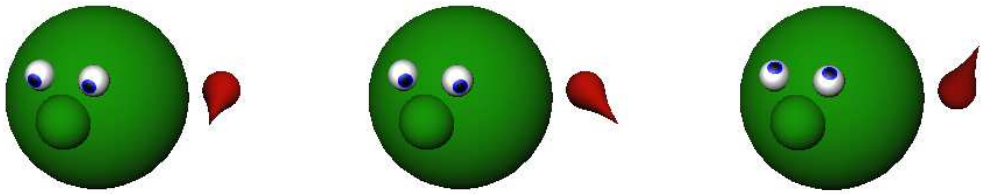


**Figure 4.1: Example of an orientation constraint. In the above images, the character's eyes are constrained to match the orientation of the red control figure.**

**Figure 4.2: Example of an aim constraint. In the above images, the character's eyes are forced to look at, or aim toward, the red diamond on the control figure.**

ordinate system to the target object's coordinate system. The position remains unaffected.

**Aim Constraint:** Forces the constrained object's coordinate system to orient itself so that it looks at, or aims toward, the pivot point of the target object. The position of the constrained object remains unaffected.

Note that these constraints support more than one target object. In this case, the constrained object's final position or orientation is calculated as if each target object was the only target, and the results are blended using user provided weights. These weights are normalized to ensure that the final position and orientation are bounded by the target positions and orientations. A point constraint with multiple

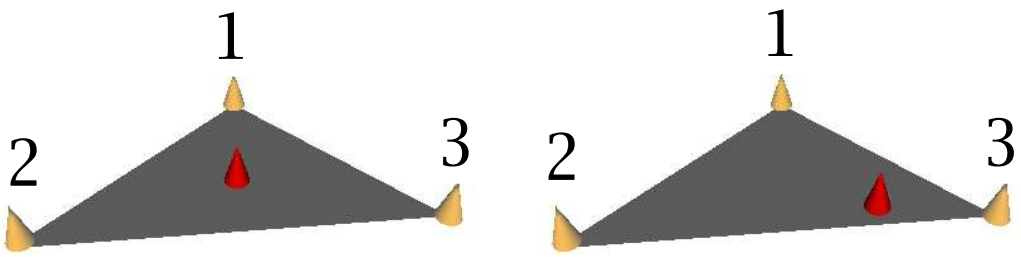



**Figure 4.3: An example of a weighted point constraint. The red cone is point constrained to the orange cones. In the first image, the weights for cones 1, 2, and 3 are all equal to 1.0, whereas in the second image the weights are 1.0, 2.0, and 6.0, respectively. By normalizing the weights, the pivot point of the red cone is constrained to the triangle defined by the orange cones.**

targets is illustrated in Figure 4.3.

### 4.1.2 Deformers

As mentioned previously, deformers are procedural manipulations which change the positions of a surface's vertices or control points, thus deforming it. Note that a deformer need not change the positions of all of a surface's vertices. In fact, it is possible to define a *deformation set* to specify those points which a certain deformer should act upon. The benefits are two-fold. First, the artists is given greater control over the extent of the deformation. Second, the number of computations in a high-resolution model is greatly reduced. This is analogous to the definition of regions in Pasquariello and Pelachaud's Greta model [PP01].

#### Cluster deformers

*Cluster* deformers provide a way to attach a second pivot point to a selection of vertices or control points of a curve or surface. This new pivot point is generally different from the original pivot which defines the curve or surface's position in

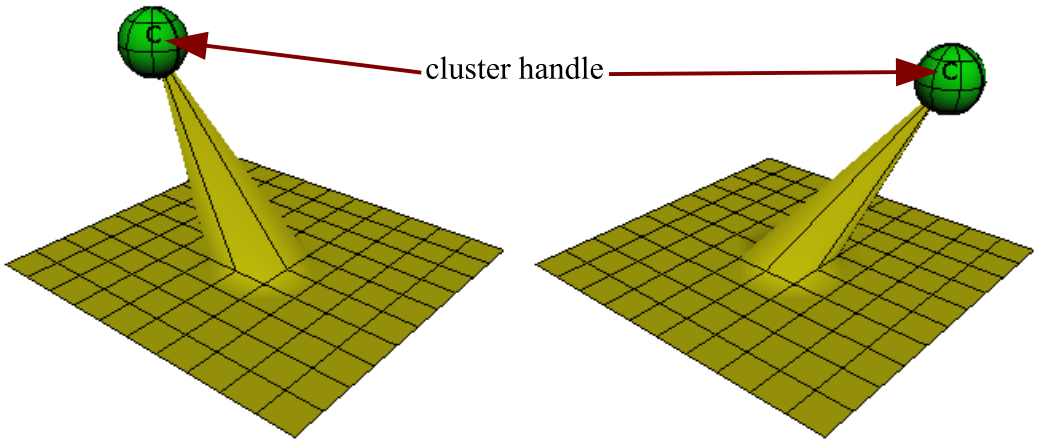



**Figure 4.4: An example of how cluster deformers can be used to constrain individual vertices of a surface. In the images above, a cluster has been applied to a single vertex of the polygonal plane. The cluster handle, shown by a black letter "C," has been point-constrained to the pivot point of the green sphere. Therefore, as the sphere is moved, the cluster handle follows, bringing along the vertex. The cluster handle, therefore, acts as a second pivot point for the vertex, independent of the surface pivot point.**

space. Transformations applied to the cluster handle (representing the new pivot point) are also applied to the vertices in the cluster's deformation set. This is useful when the artist wants direct vertex manipulation during animation.

Clusters are also used for constraining regions of a surface. Since constraints affect surface pivot points only, the artist can use a cluster to provide an alternate pivot for part of a surface, as shown in Figure 4.4.

Another degree of freedom is provided by using *cluster weights*. These user defined weights control the degree to which the cluster transform is applied to the elements of the deformer set. Figure 4.5 shows the results of changing cluster weights on an object.

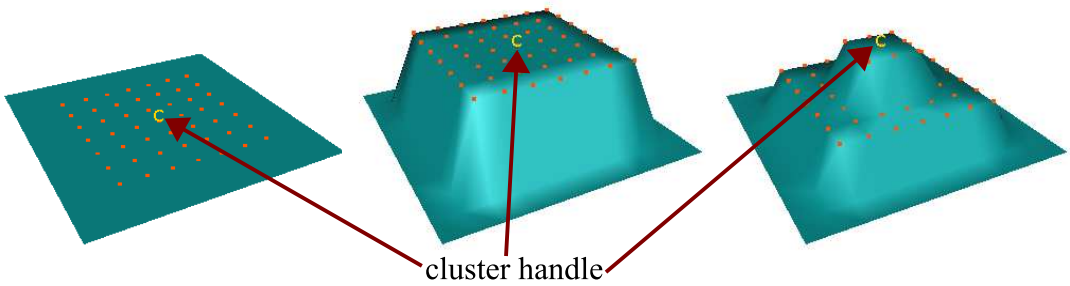



**Figure 4.5: The images above show the effects of cluster weights on a deformed object. The left image shows the polygon surface before deformation. The orange vertices represent the elements of the cluster deformer set. At center, the cluster has been displaced upward, moving the vertices in the deformer set along with it. The image on the right shows the deformed surface after changing the cluster weight on the outer rows of vertices to 0.5 (the default weight is 1.0).**

### Joints and skins

Imagine for a moment that you are playing with a hand puppet. As you open and close your hand, different parts of the puppet move, achieving expressive poses. Consider each finger in your hand as an animation control and let the puppet represent the surface of your character. As you articulate your hand, you are altering the shape of the puppet surface. This is the essence of the *skin* deformer. The name is derived from an analogy to nature: the driving elements (your fingers) are considered "bones" surrounded by a "skin" (the deformable surface).

The skin deformer cannot exist without a corresponding set of driving transforms. These transforms are usually linked together to create a rigid articulation system, such as a human skeleton, but this is not entirely necessary. The transforms are commonly referred to as *bones* or *joints*[1].

There are two ways to bind a skin deformer to a surface using a particular

[1]The term "bone" is used in discreet 3ds max and the term "joint" is used by Alias Maya.

set of joints. In *rigid* binding, each vertex on the surface is associated with a single joint. On the other hand, *smooth* binding allows more than one joint to determine the final vertex position. User-defined weights determine how much each joint's transform affects a vertex. This finer control over the deformation allows the surface to deform more smoothly. Note that in both cases, the deformation is dependent on the original configuration of the joint system when the skin is attached. This configuration is called the *bind pose* or the *preferred orientation*. Examples of both skin types are shown in Figure 4.6. The interested reader can find a description of the deformation algorithm in [LCF00].

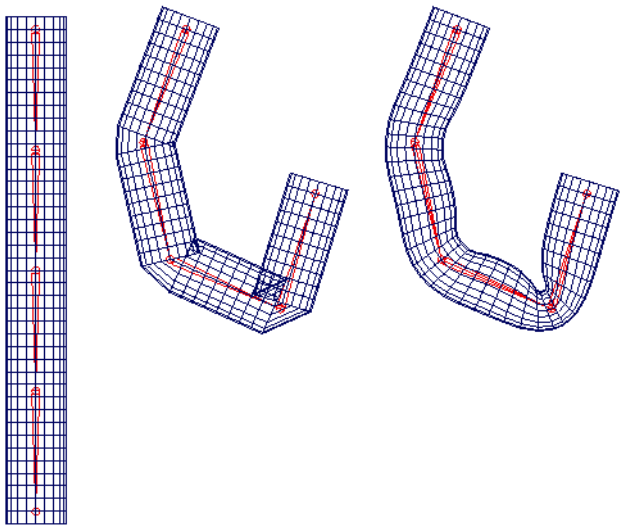


**Figure 4.6: Examples of smooth and rigid skin deformers are shown above. Pictured from left to right are the undeformed surface, the surface deformed by a rigid skin, and the surface deformed by a smooth skin with a maximum of five influence joints. The joint systems are highlighted in red.**

### Wires

*Wire* deformers were inspired by sculpting armatures [SF98]. Using a wire curve, the artist can deform surface geometries directly, almost as if creating a rough

sketch of the 3D model. A simplified version of the wire deformer, as used in this work, is described here. The reader is referred to [SF98] for more comprehensive treatment.

The wire deformer is completely specified by two curves of equal parametric range (a *base* wire and a *deforming* wire), a dropoff distance value, and a dropoff function. The dropoff function is a decreasing function defined on the interval $[0, 1]$, where the derivatives at the interval endpoints are usually zero. This work uses the dropoff function $f(x) = (1 - x^2)^2$, for $x \in [0, 1]$.

Intuitively, the deformer measures the vector difference between corresponding points on the base and deforming wires. This deformation is then applied to the vertices within the dropoff distance from the base wire, weighted by the dropoff function.

More specifically, consider the point $P$ in the deformation set, and let $B$ and $D$ be the base and deforming wires, respectively (see Figure 4.7). Additionally, let $f(x)$ be the dropoff function and let $d_0$ be the dropoff distance. Finally, let $u$ be the curve parameter of the point on $B$ closest to $P$, and denote this point as $B(u)$. The wire deformer finds the corresponding point $D(u)$ on the deforming wire and calculates the vector displacement $\vec{r}$ between $B(u)$ and $D(u)$. The final position $P'$ of $P$ is found by adding $\vec{r}$ to $P$, weighted by the value of $f(d/d_0)$. Here, $d$ is the distance between $P$ and $B(u)$. Note that if $d > d_0$, the weighting function is not defined. In this case, the wire deformer has no effect on the position of $P$. Also note that unless the base wire or the dropoff distance are modified, the calculation of the parameter $u$ can be cached to speed up computation.

Figure 4.8 shows the effects of the wire deformer on a surface and the effects of varying the dropoff distance.

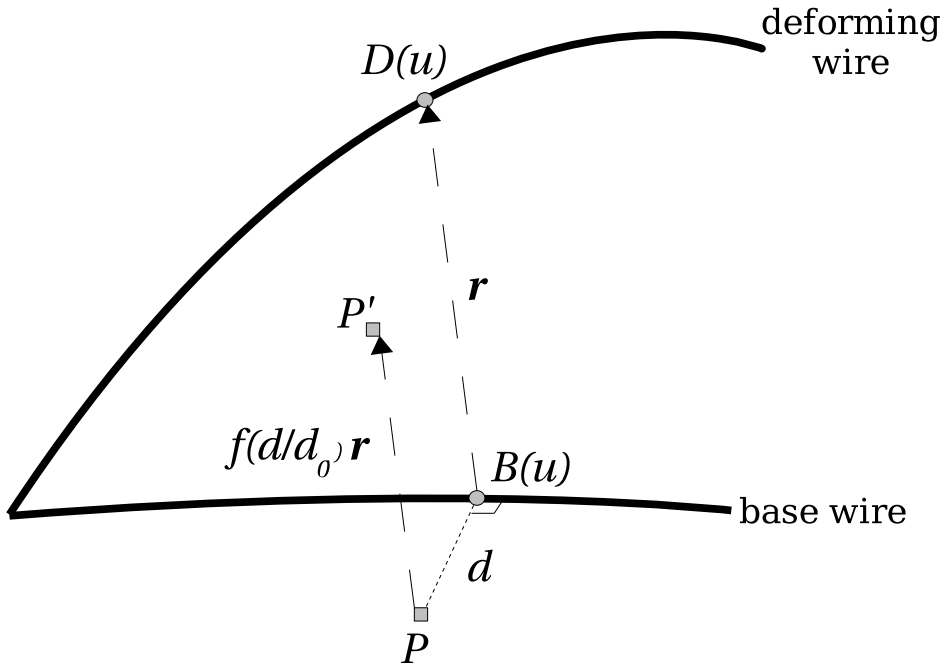



**Figure 4.7: This diagram illustrates the wire deformer algorithm, as described in the text. Note that $\vec{r}$ is shown as a bold letter '$r$' above.**

### Push-to-surface

The name of the *push-to-surface* deformer gives its functionality away. This deformer moves vertices behind a reference surface to a position on the surface. The definition of "behind" is determined by a user-specified reference direction.

The deformation algorithm is very simple. For each vertex in the deformation set, we define a ray with its origin at the vertex and pointing in the reference direction. If the ray intersects the reference surface, the vertex is moved to the point of intersection; otherwise, its position is unchanged. An example of this deformer in action is shown in Figure 4.9.

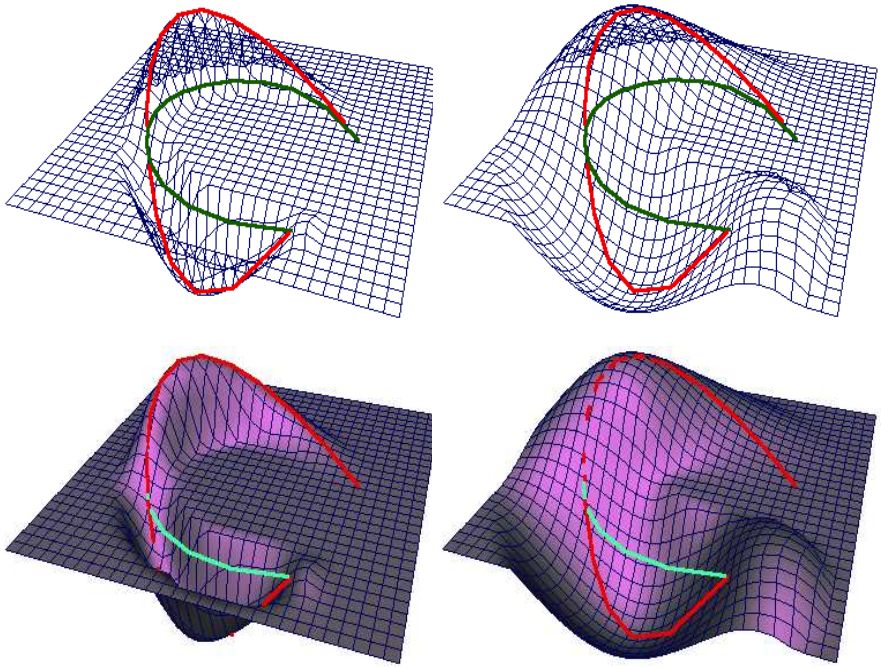


**Figure 4.8: The effects of a wire deformer are shown above. The base wires are highlighted in green and the deforming wires in red. The two surfaces on the left have the wire dropoff distance set to 1.0, while the ones on the right have a dropoff of 3.0.**

## 4.2 Skin/Muscle Model

The generic rig used in our animation system was developed using a mixture of structural and ad hoc components. Since the rig will animate a variety of geometrically different faces, we have based the animation parameters on the common human musculo-skeletal structure. However, there is no formal structural basis for the deformations employed by the skin/muscle model used in the rig; these simply provide good results. Nevertheless, there are a few intuitive principles which led to the development of this model.

From our discussion in Chapter 2 we know that a number of muscles attach to neighboring facial areas. Therefore, simultaneous contractions of these muscles will pull the same patch of skin in more than one direction. Additionally, the dis-

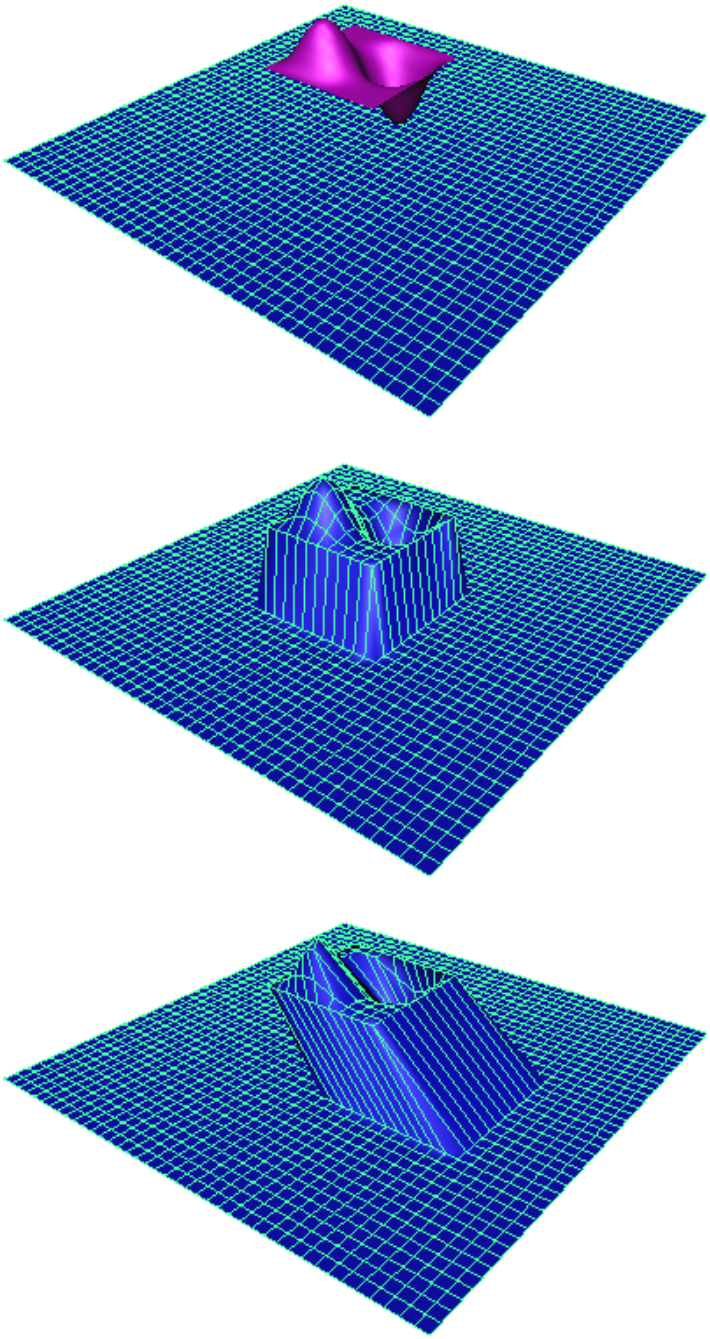


**Figure 4.9: An example of the push-to-surface deformer. The top image shows the reference surface in a purple color, before the deformation is applied to the blue surface. The middle image shows the effects of the deformation, and the bottom image shows the effects of altering the reference direction.**

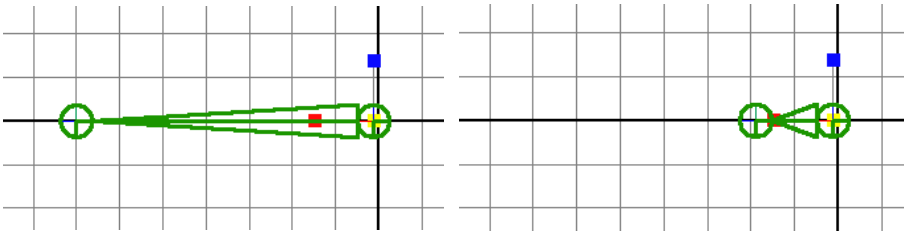


**Figure 4.10: Joints were chosen to model muscle contractions due to their visual feedback. The images, from left to right, show a relaxed and contracted muscle joint, respectively.**

placement of skin due to muscle contraction is strongest near the muscle's insertion point, dropping off as one moves away from this region. Our model takes these properties into account.

The basic elements of the model are muscles and skin. Muscles are defined by an origin and an insertion point, and the skin is simply the deformable surface. These elements are coupled together by a deformation chain, which is triggered by muscle contractions. To simplify the description of this deformation chain, we will describe its components as we piece it together.

The muscle contraction is modeled as a scaling transformation about the muscle's origin in the direction of the insertion point. Although any object can be used to represent this muscle transformation, we have chosen to use joints due to their intuitive visual feedback, shown in Figure 4.10.

The muscle contraction affects the skin surface indirectly using wire deformers. As the muscles contract, the wires change shape, thereby affecting the skin. This procedure is shown in the simple diagram of Figure 4.11. You can think of the wire as skin tissue being pulled by the muscle. Figure 4.12 shows a sample setup with two joints affecting a wire, which we will use to illustrate the construction of the deformation chain.

As stated above, a muscle contraction does not affect all points on the skin

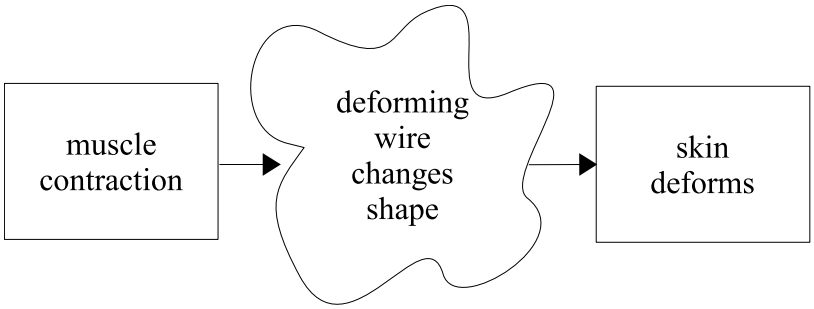



**Figure 4.11: Simple version of the skin/muscle deformation chain.**

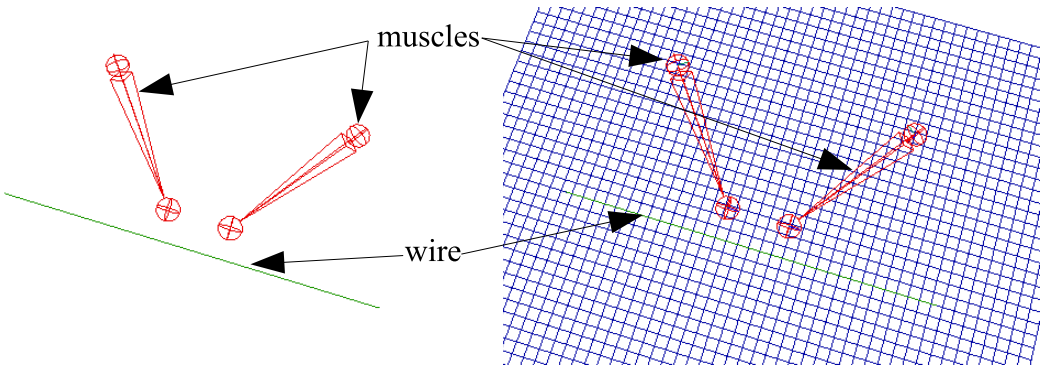



**Figure 4.12: A sample skin/muscle setup. The two joints, shown in red, represent two muscles in this system. The skin surface, shown in blue, will be deformed by the green wire, which is currently in its base position.**

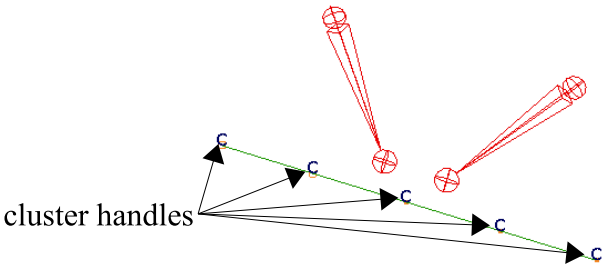



**Figure 4.13: A cluster is constructed for each of the control vertices of the deforming curve in the wire deformer. The cluster handles are indicated above by the letter "C."**

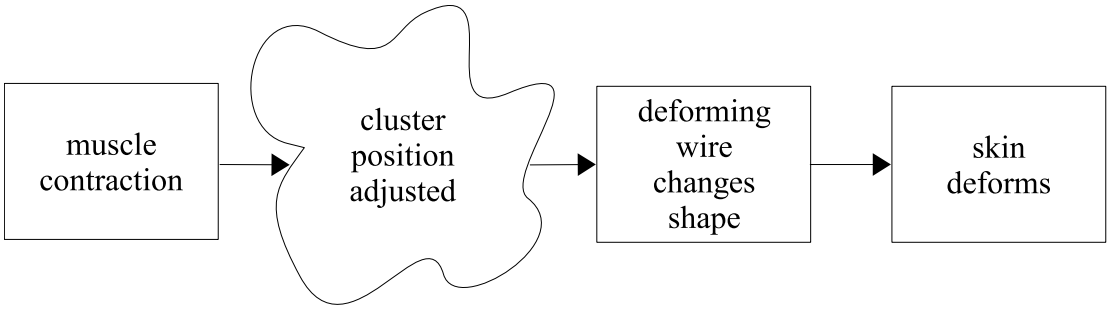



**Figure 4.14: Modified version of the skin/muscle deformation chain.**

equally. Therefore, the muscle must pull on the wire deformer's individual vertices rather than on its pivot point. To enable this, we build a cluster for each of the control vertices of the wire, as shown in Figure 4.13. The expanded deformation chain is shown in figure 4.14.

One way to tie the muscle joint contraction to the wire cluster is via parenting. This simply requires making the cluster transform a child of the joint transform in the scene hierarchy. There are two drawbacks to this approach. First, parenting is absolute—the cluster is either a child of the joint or it is not. This does not provide the flexibility to control a given joint's influence over a wire cluster. Second, since the cluster can only have one parent, it is unclear how more than one muscle joint

can modify a single cluster.

Instead, we tie the joints and clusters together using point constraints. We use *locators*, also called *points*, to help specify the constraint targets. Locators are simple, non-renderable shapes which provide coordinate frames in the transform hierarchy for other objects to use as reference. They are usually rendered in the viewport as a set of axes. We create a locator for every joint affecting a given cluster, placing the pivot point at the cluster's pivot, as shown in Figure 4.15. Each locator is then parented to its corresponding joint so that joint contractions will affect its position. An additional, unparented locator is placed at the cluster pivot to keep track of the clusters original position. Finally, the cluster is point constrained to the set of locators. By altering the constraint weights, the user can control the degree to which the contraction of a given joint affects the displacement of the cluster, thereby controlling how much the wire deformer changes. An example of the chain in action is given in Figures 4.15 and 4.16, and the final deformation chain is illustrated in Figure 4.17.

By tying the wires and joints together using point constraints, we ensure that the wire displacement can respond simultaneously to more than one muscle contraction. Because the clusters' final positions are controlled by a weighted average of the targets, the effect of multiple contractions is dampened and thus the skin behavior is not too erratic. Finally, point constraint weights provide the flexibility which allows the artist to control the effects of a given muscle contraction on a patch of skin. This flexibility is the key to the expressiveness of the reference model.

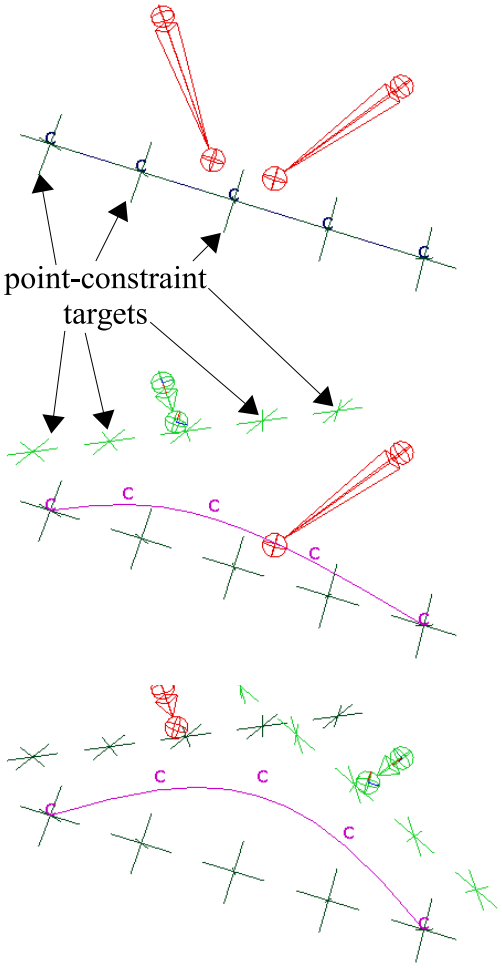



**Figure 4.15: Use of point constraints in the skin/muscle model. In this example, there are three target locators per cluster: one for each joint and one unparented locator. Each locator is rendered as three intersecting lines. The top image shows the initial position of the target locator shapes and the lower images show the effects of the muscle contractions on these targets. The contracting muscle and its associated locator targets are highlighted in green. As the targets move, the point constraint adjusts the position of the clusters, deforming the wire. Note that the constraint weights have been specified to deform the center of the wire more than the endpoints. The effects of these contractions on the skin appear in Figure 4.16.**

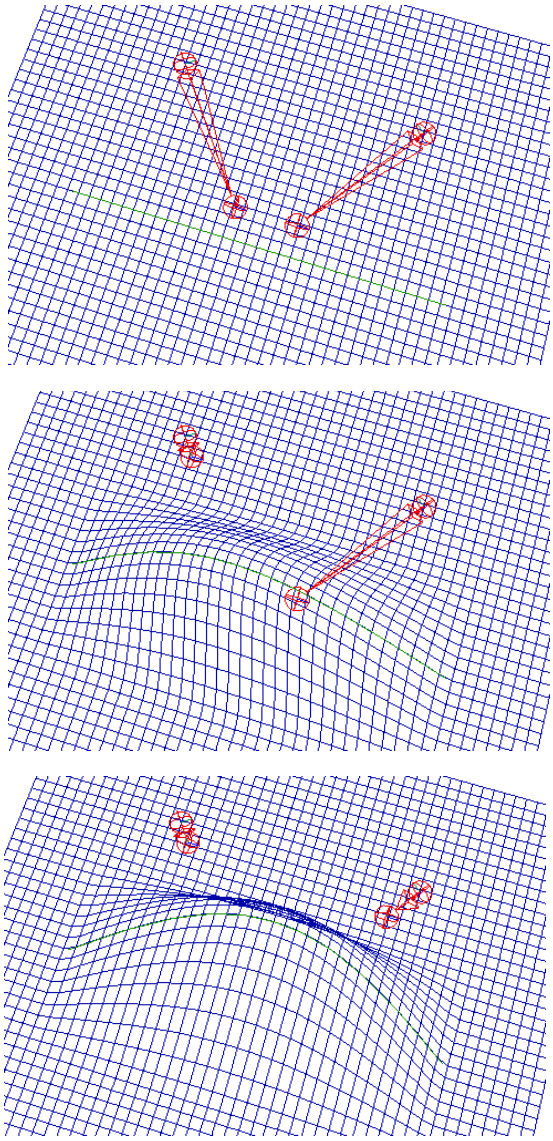


**Figure 4.16: An example skin/muscle system in action. As a muscle joint contracts, it alters the positions of the cluster point constraint targets associated with it. This forces the clusters to move, thereby altering the shape of the wire deformer and thus deforming the skin. The images show the skin deformation caused by the muscle contractions shown in Figure 4.15.**

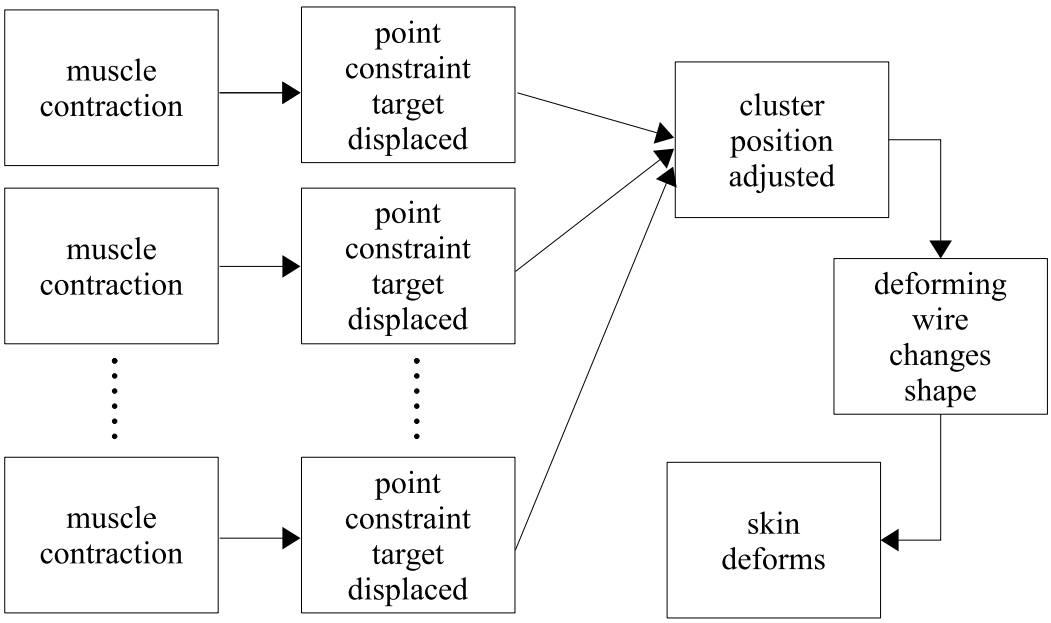



**Figure 4.17: The final deformation chain in the skin/muscle model.**

## 4.3 Reference Model

The reference head model, affectionately named "Murphy," is a generic human character capable of a wide variety of facial movement and expression. It is comprised of two separate components: a polygonal face model and a muscle-based rig.

### 4.3.1 Polygonal Face Model

The reference head model was sculpted using Alias Maya 5.0 [Ali] software following the procedure outlined in [Osi03]. Initially, the mouth, eye, and ear regions where roughly sculpted using NURBS surfaces (see Figure 4.18). More detail was added after converting these surfaces to polygons. A single face mask was created by combining the polygonal surfaces into one and filling in the holes. In order to create a symmetric model, a copy of half of the face was mirrored about the meridian and

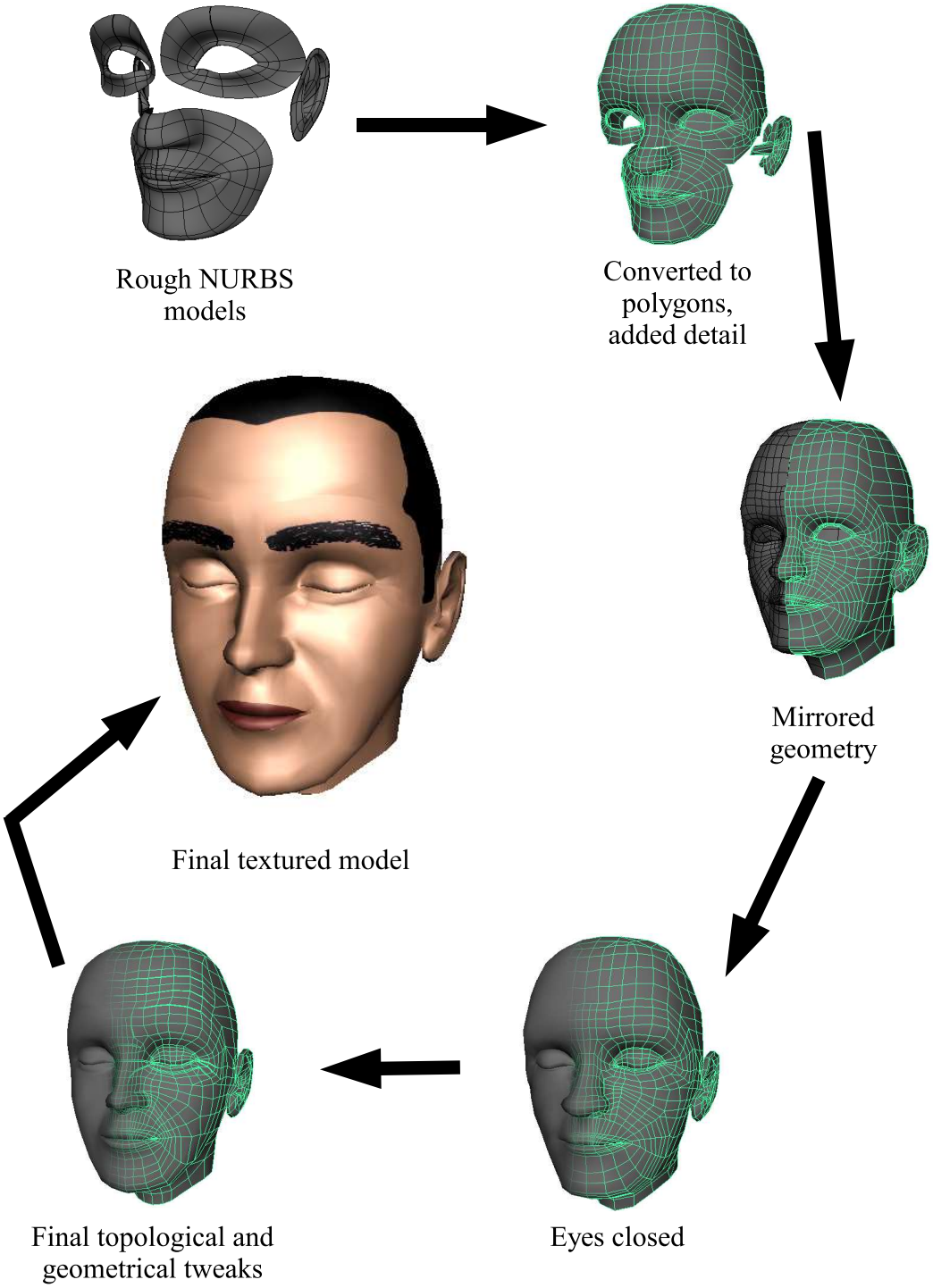



**Figure 4.18: Construction of a generic head model.**

combined with the original half, producing a full face model. The eyes were then closed to aid in the scan-fitting process (see next chapter). Additional tweaks were performed on Murphy to remove the cartoonish look and improve the topology of the surface for deformation. Finally, we applied a simplistic texture to provide a reference for the eyebrow positions and to enhance the human quality of the character.

Murphy's topology was optimized for facial animation, as shown in Figure 4.19. We used a radial topology for the mouth and eye regions to facilitate the radial movement of the orbicularis oris and orbicularis oculi muscles. To aid the deepening of the nasolabial fold, the radial point layout around the mouth is continued from the lips to the region above the nostrils. Furthermore, extra edges were added to expressive regions of the face where creases form. These regions include the forehead, the nose, and the region below and lateral to the eyes. Note also that Murphy's bounding box is centered about the origin and he is looking down the positive $z$ axis, with the $y$ axis pointing up.

Figure 4.20 shows the eyes, teeth, and tongue objects, which were modeled as separate surfaces.

### 4.3.2 Muscle-Based Rig

The facial rig provides the controls that bring Murphy to life. The controls are based on human facial structure and make use of the skin/muscle model introduced above. The rig itself was implemented in Alias Maya 5.0 and can be decomposed into three components: the facial mask rig, the eye rig, and the tongue rig.

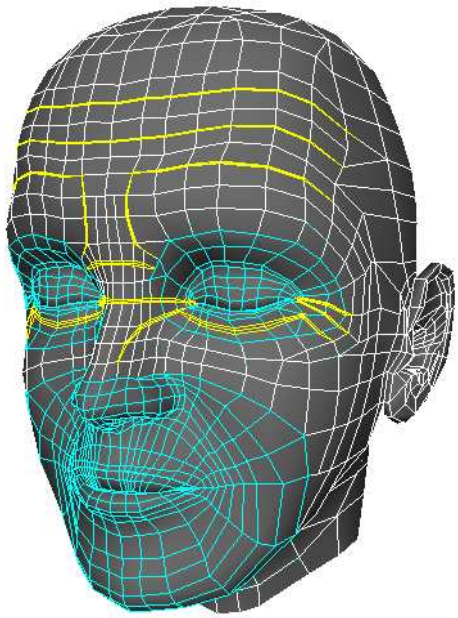


**Figure 4.19: Features of Murphy's topology are shown in this image. The area highlighted in light blue shows the radial topology used around the mouth and eye regions. The edges highlighted in yellow are additional edges included for creating creases.**

**Figure 4.20: Models for the eyes, teeth, and tongue. Note that the tongue shape, which is attached to the lower teeth, is hidden by the upper teeth.**

### Facial mask

The facial mask rig accounts for 95% of all movement in the face. It is responsible for simulating all muscle contractions, skin creases, and neck and jaw rotations. Therefore, it is the most complex component of Murphy's rig.

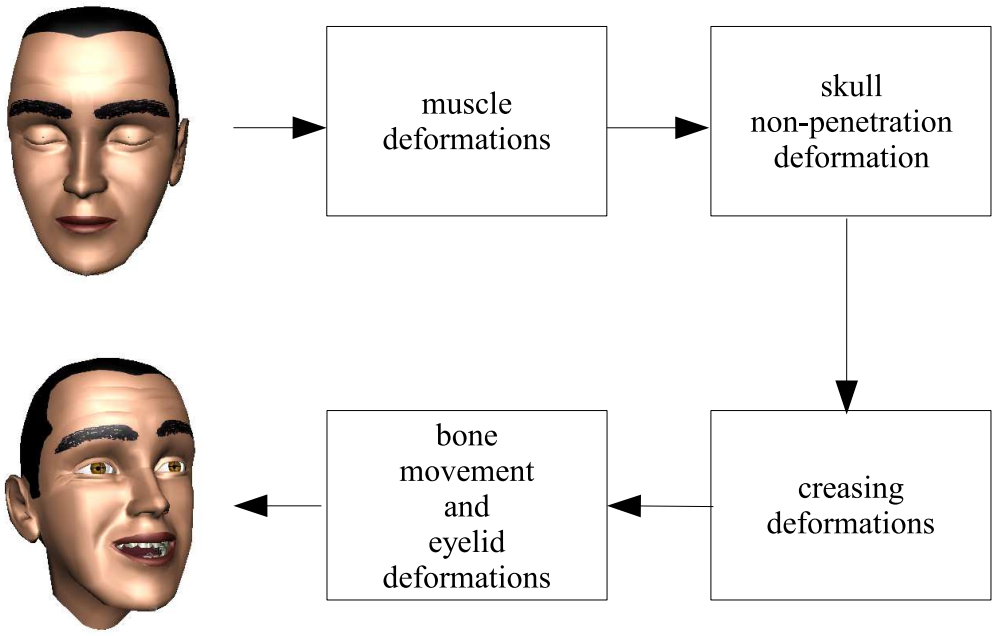



**Figure 4.21: Deformation chain for the facial rig.**

A simplified deformation chain diagram for the facial mask rig is shown in Figure 4.21. Conceptually, the chain is divided into four components. Initially, the deformers representing muscle contractions are performed using the skin/muscle model. Since this model does not account for skin sliding over bone, a skull non-penetration deformation follows. Next, a skin creasing deformation, triggered by certain muscle contractions, further enhances the expressions of the forehead and eye regions. Finally, deformations controlling head and jaw rotations (termed *bone movements*) and eyelid opening/closing are applied. The reasons for placing the gross motor movements of the head and jaw at the end of the chain will be

explained later in this section.

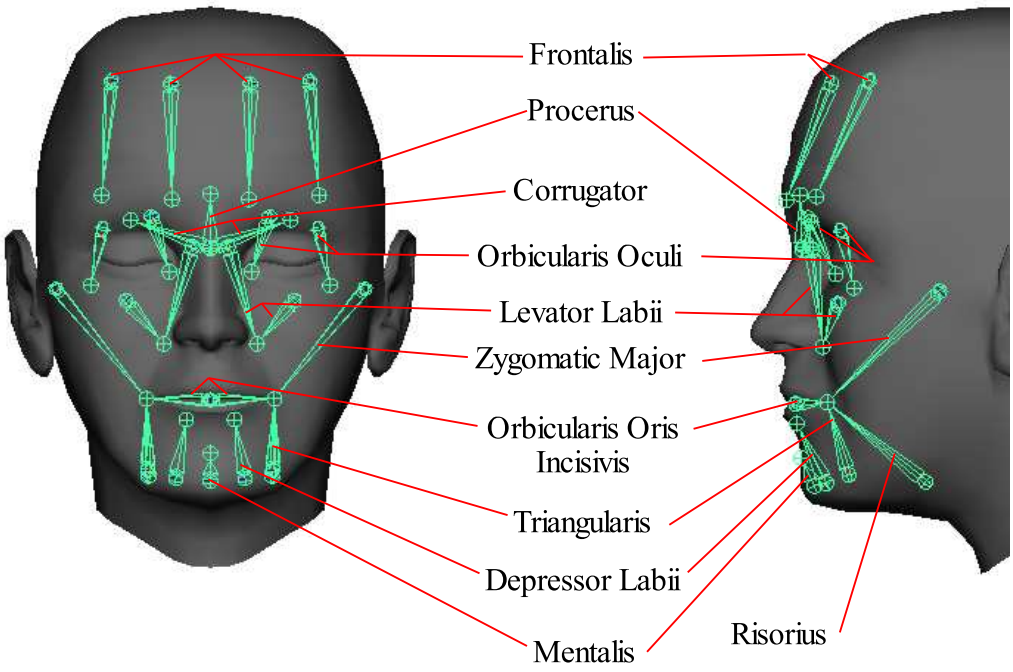



**Figure 4.22: Muscles used in Murphy's rig.**

Figure 4.22 shows the muscles implemented in the rig. Vertices on Murphy's polygonal mesh were selected to act as origin and insertion points for each muscle, thereby parameterizing the muscle construction to the model topology. Additionally, the muscle origin position was scaled uniformly by 0.95 about the center of Murphy's bounding box before creation of the joint to simulate the subcutaneous attachment to the skull, as illustrated in Figure 4.23. With the exception of the procerus and mentalis, all muscles have symmetric joints on the left- and right-hand sides of the face. The frontalis muscle has two components per face half, allowing for contractions of the inner and outer portions of the muscle.

Another feature of note on the mentalis muscle is that the origin and insertion points are switched, thereby simulating a muscle contraction by scaling the joint *outward*. This orientation worked better in practice than the standard treatment

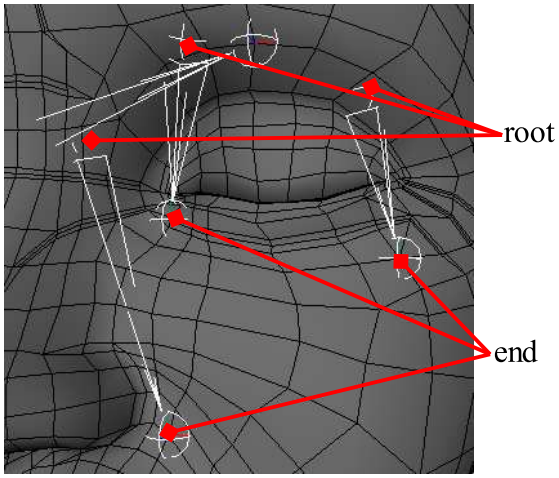



**Figure 4.23: This view of Murphy's left eye shows how the muscle joints are constructed from the vertices in the topology of the face mask. Note that the ends of the joints are positioned at the vertex locations—the roots of the joints are displaced to simulate attachment to the skull (see text).**

since, in contrast to the other muscles of the mouth, the mentalis pushes on the lips instead of pulling.

Observe that the sphincter muscles are notably absent from this muscle system (see Figure 2.7 for reference). In case of the orbicularis oculi, two joints have been used to simulate the skin rising in the cheek region due to contractions (see Figure 4.22). Only the incisivis fibers of the orbicularis oris have been included in this rig. The levator palpebrae muscle is also excluded from this system, as it is implemented in the final deformation component of the rig. A comparison of the muscle joints to the human musculature is shown in Figure 4.24.

The rig muscles attach to sixteen wires on the skin, illustrated in Figure 4.25. Intuitively, these wires represent the areas of the skin that are most affected by muscle contractions. Once again, with the exception of the upper and lower central wires around the mouth, each wire has a symmetric twin on the opposite side of the face. The wires are constructed by interpolating specific vertex positions on

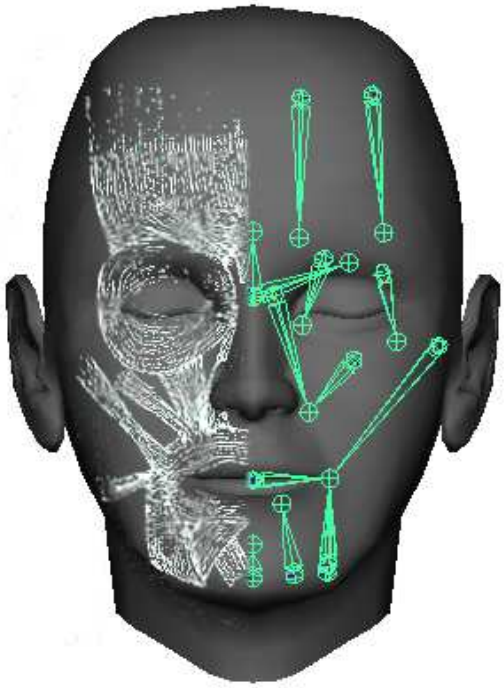


**Figure 4.24: Comparison of Murphy's rig muscles to the human musculature. Note that the risorius muscle, which appears to be missing, is actually hidden by the triangularis muscle, as shown in Figure 4.22.**

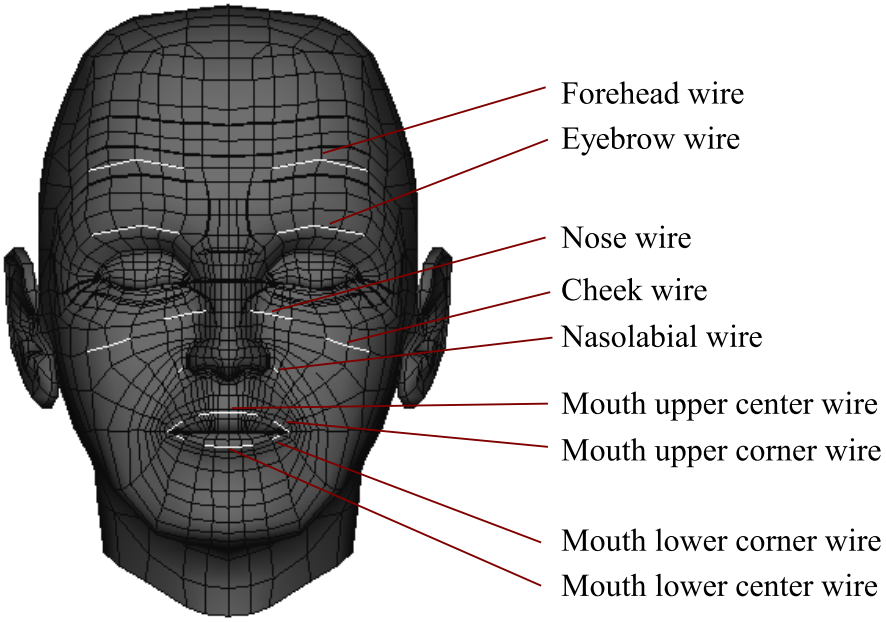



**Figure 4.25: The wires used in Murphy's rig.**

the face mesh, maintaining the topological parameterization. Although we used cubic NURBS curves initially for the wires, experiments showed that linear curves provided similar results with lower computational costs[2]. Table 4.1 shows which muscles are connected to each wire using the skin/muscle deformation chain. The weights on the cluster point constraints in each wire were adjusted manually until the desired facial contraction effects were obtained.

Due to the proximity of the upper and lower lips in the generic face mesh, vertices in the lower lip are within the dropoff distance of the upper lip wires and vice versa, which is a source of undesirable deformations. To correct this problem, the region around the mouth was carefully partitioned into deformation sets specific to each wire. These deformation sets are illustrated in Figure 4.26. An additional benefit of this partitioning is the reduction of computation in each wire deformer. The deformation sets for the wires around the eyes and cheek regions were also specified, as shown in Figure 4.27.

As briefly stated previously, the skin/muscle model has no provisions for simulating skin sliding over a skull. This is not a problem in the mouth region since the face geometry does not follow the skull lines closely there. However, this does create a problem in the forehead region, specially when the eyebrows are raised (contraction of the frontalis muscle), as shown in Figure 4.28. As the muscle joints pull on the skin, the region of the eye representing the eyebrow ridge bulge rises with the wire, giving the face an unnatural, stretched-out look. The second stage of the facial deformation chain fixes this using a push-to-surface deformer, which pushes out the skin in the skull region and restores the natural look of raised eyebrows.

[2]Linear curves use a simpler algorithm for closest distance to a point (used in the wire deformer) and have a smaller number of control points.

**Table 4.1: Muscle to wire connections in Murphy's rig. Unless otherwise noted, each muscle and wire correspond to the same side of the face mesh.**

| Wire | Muscles |
|---|---|
| Forehead | Frontalis (both components)<br>Procerus<br>Corrugator |
| Eyebrow | Frontalis (both components)<br>Procerus<br>Corrugator |
| Nose | Orbicularis Oculi (inner) |
| Cheek | Orbicularis Oculi (outer) |
| NasoLabial | Levator Labii (nasal)<br>Levator Labii (mid) |
| Mouth Upper Center | Levator Labii (nasal, both sides)<br>Levator Labii (mid, both sides)<br>Zygomatic Major (both sides)<br>Orbicularis Oris Incisivis (both sides) |
| Mouth Upper Corner | Levator Labii (nasal)<br>Levator Labii (mid)<br>Zygomatic Major<br>Risorius<br>Triangularis<br>Orbicularis Oris Incisivis |
| Mouth Lower Corner | Zygomatic Major<br>Risorius<br>Triangularis<br>Depressor Labii<br>Mentalis<br>Orbicularis Oris Incisivis |
| Mouth Lower Center | Zygomatic Major (both sides)<br>Risorius (both sides)<br>Triangularis (both sides)<br>Depressor Labii (both sides)<br>Mentalis<br>Orbicularis Oris Incisivis (both sides) |

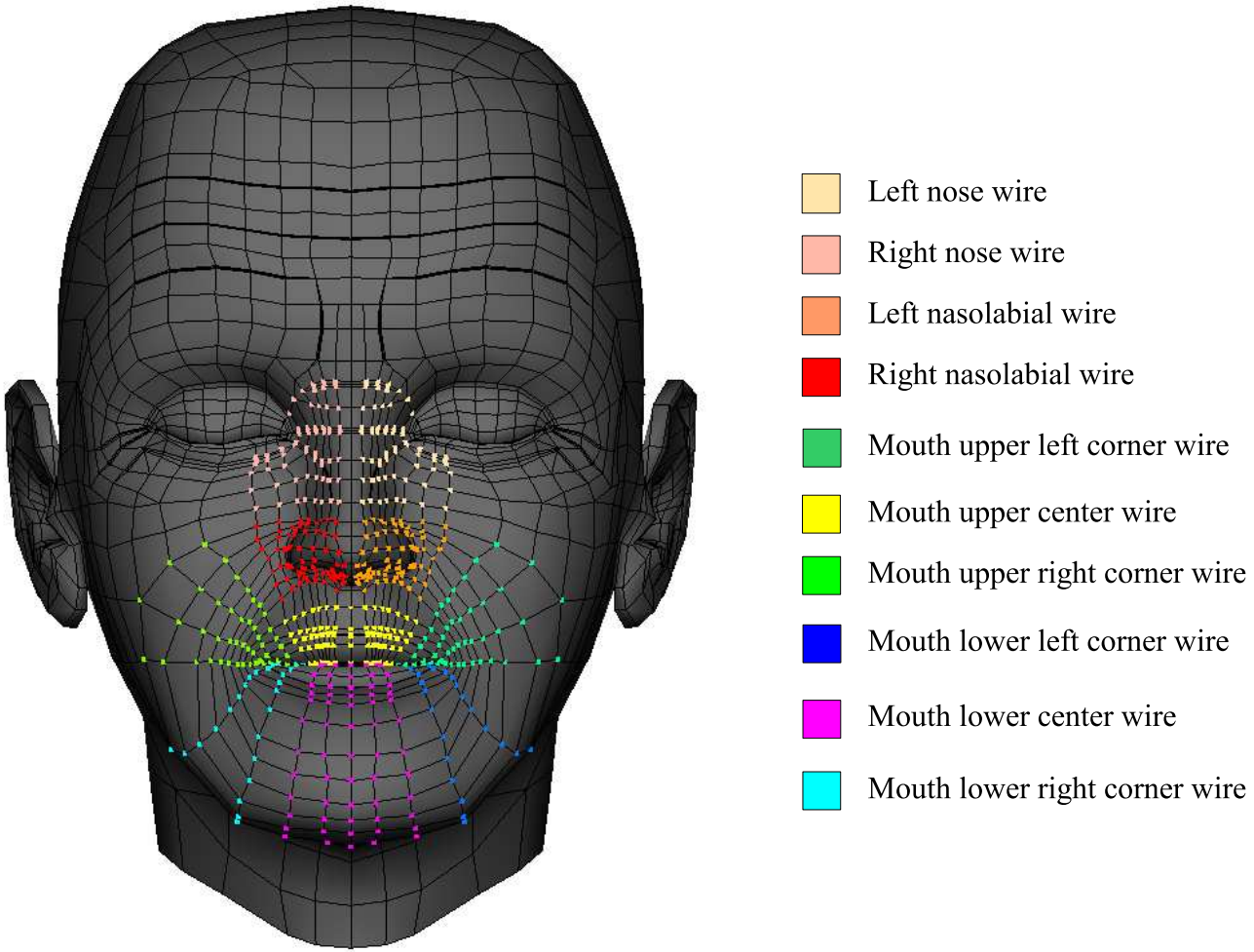



**Figure 4.26: Deformation sets corresponding to the wire deformers around the mouth region.**

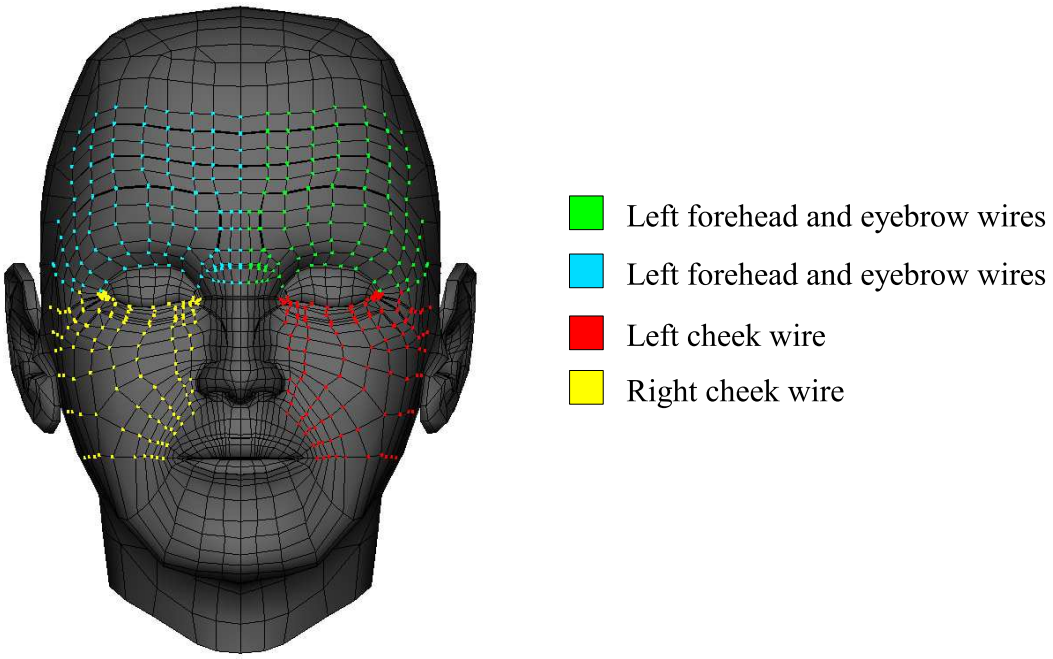



**Figure 4.27: Deformation sets corresponding to the wire deformers around the forehead and cheek regions.**

Figure 4.29 shows the procedure we use to generate the skull surface directly from Murphy's topology. First, we generate a set of cubic NURBS curves over the forehead surface. Each curve is constrained to pass through a series of face mask vertices contained within a vertical section of the forehead. The curves are then lofted to create a bicubic NURBS surface used by the push-to-surface deformer. We use the $z$ axis vector ($[0\ 0\ 1]^T$) to define the forward direction of the deformer, which is the same direction Murphy looks toward.

The push-to-surface deformer is the bottleneck of the facial deformation chain. The slowdown occurs from the number of ray/surface intersection tests performed against the bicubic NURBS skull surface. Therefore, we have attempted to trim down the skull deformation set as much as possible. The final set is shown on Figure 4.29.

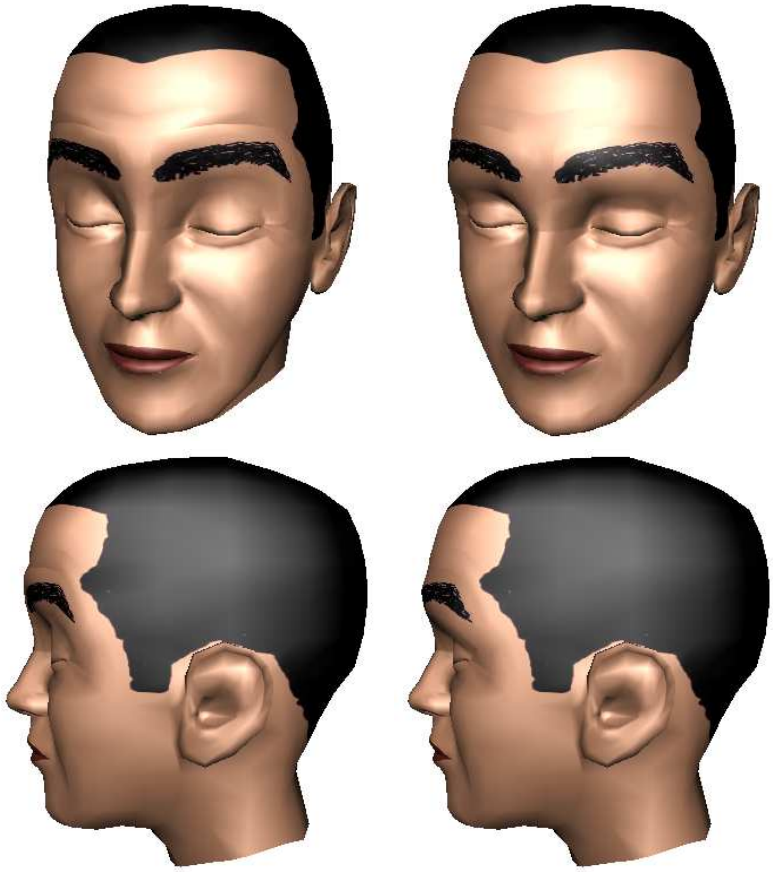


**Figure 4.28: Simulation the skull using a push-to-surface deformer. The images on the left show Murphy raising his eyebrows without the skull deformations. The face looks unnaturally stretched out. On the right, the face regains its natural look by use of a push to surface deformation.**

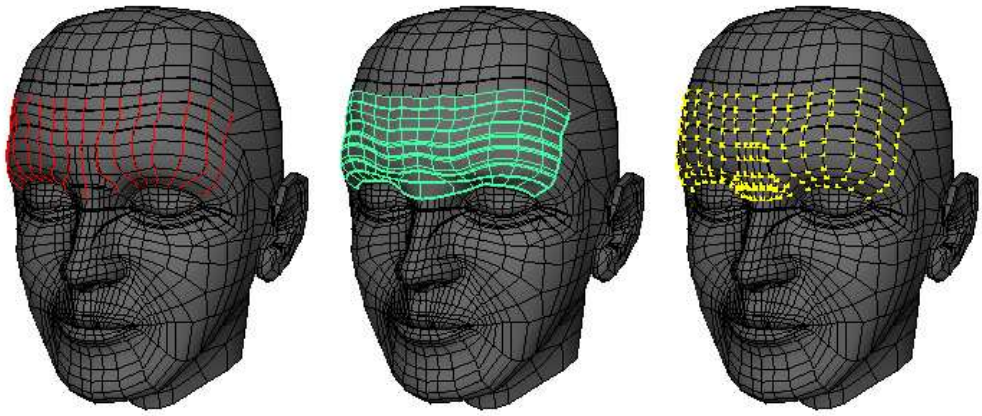


**Figure 4.29: Details of the skull surface generation and deformation. Pictured on the left are the NURBS curves used to define the skull surface. These curves are constrained to pass through a series of face mask vertices contained within a vertical section of the forehead. The curves are then lofted to produce the skull surface shown on the center image. Finally, the vertices of the skull push-to-surface deformation set are highlighted in yellow on the far right.**

Once the skull deformations are applied, the forehead appears fairly flat. However, when the eyebrows are raised, the skin of the forehead should crease. The job of the next set of deformations in the facial rig chain is to deepen the skin creases in the face. Figure 4.30 illustrates the results of this step.

Note that we do not solely apply creasing deformations to the forehead region. Creases are an important component of facial expression, appearing also in the areas around the eye and nose. We have added extra edges in these regions to enhance Murphy's expressions.

The creases themselves are implemented by creating cluster deformers on selected vertices. The clusters are displaced slightly, usually in the negative $z$ direction, to deepen the skin crease. The degree of displacement is controlled by the muscle contraction parameter. Cluster weights are used to blend creases together when the corresponding regions of influence of the trigger muscle overlap

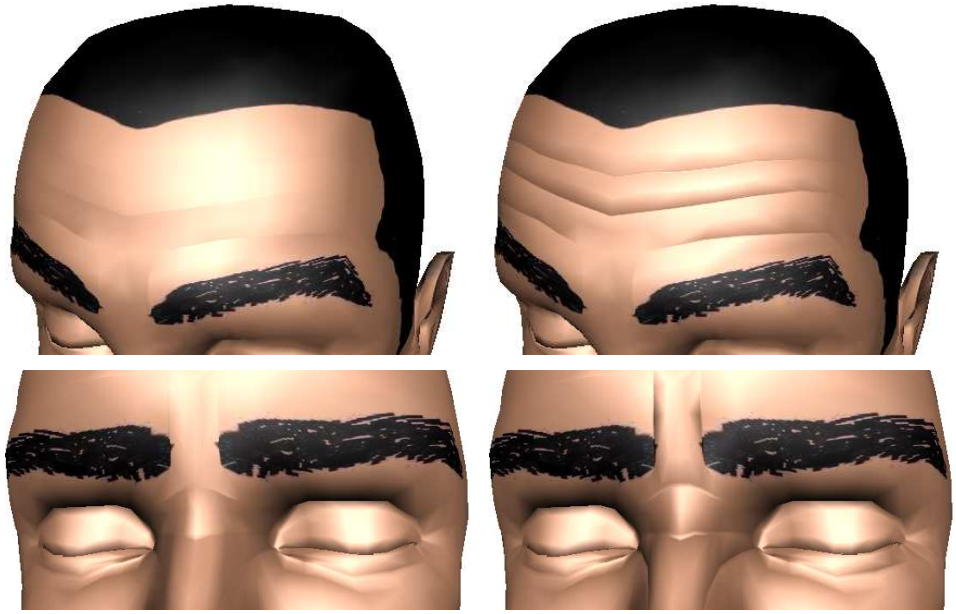


**Figure 4.30: The effects of the creasing deformations in the third step of the facial rig deformation chain. The images on the left show Murphy before this step is applied, and the results of the creasing deformations are shown on the right. Note how the deep creases enhance the facial expression.**

(for example, in the forehead). Figure 4.31 shows the vertices affected by creasing deformations. The colors correspond to individual cluster deformers and show how the cluster weights blend creases together.

Once the creases are in place, we arrive at the last step in the facial mask deformation chain. This step implements deformations that simulate the head rotating, the jaw opening, and the eyelids opening and closing. The unifying factor behind these movements is their rotational character: neck vertebrae rotate one over the other, the jaw rotates as the mouth opens, and the eyelids "rotate" about the center of the eye. Joints and skinning provide an intuitive implementation for this type of deformation.

Figure 4.32 shows the joints used to skin Murphy. In order not to confuse these joints with the muscle joints described previously, we call these *bone* joints. We use

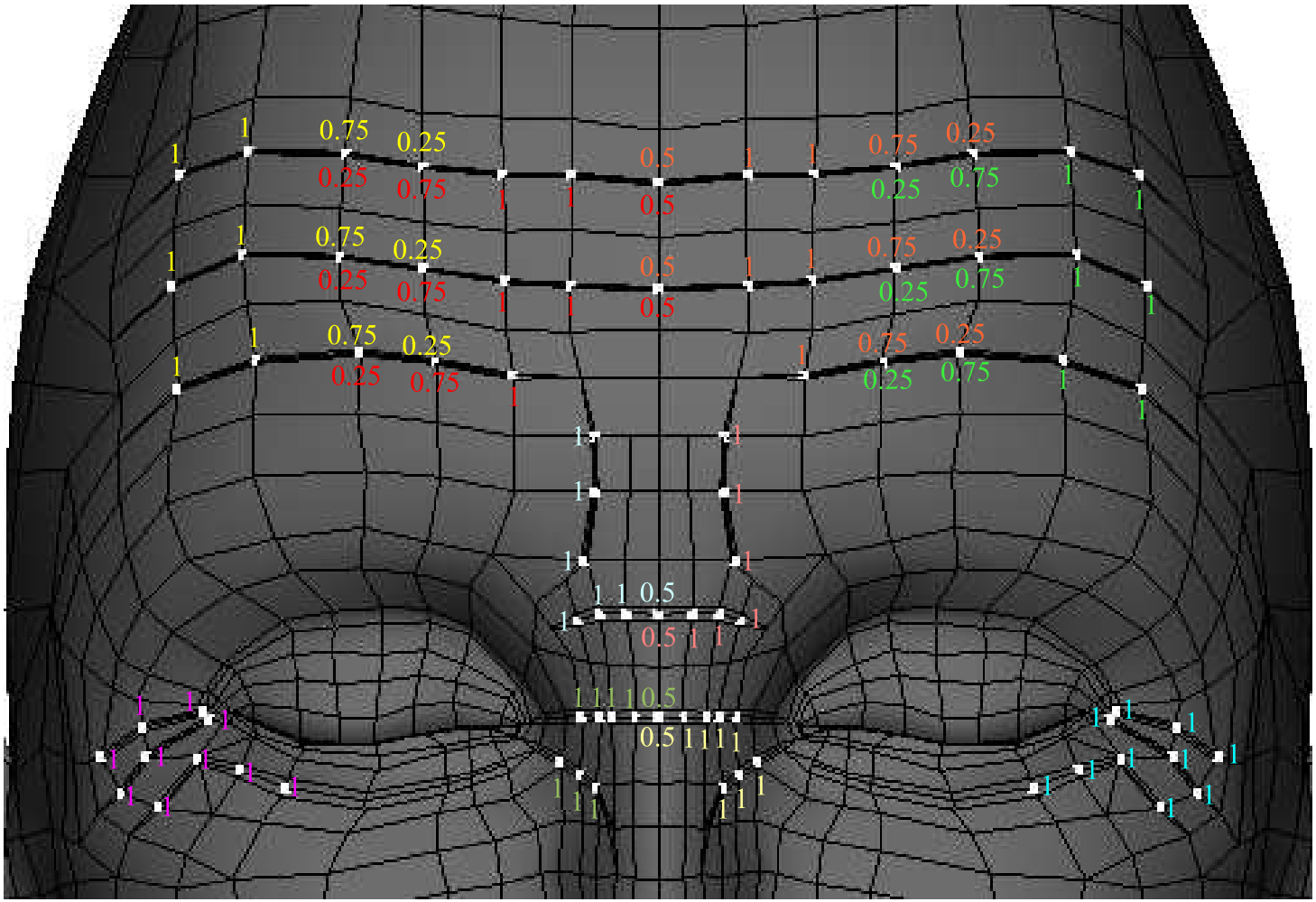


**Figure 4.31: The creasing deformation clusters used in Murphy's rig. The vertices involved in these deformations are highlighted in white. Each color corresponds to an individual cluster, and the number values represent the weight of the vertex in the cluster of that color. Note how the weights blend the creases together in the forehead and down the meridian.**

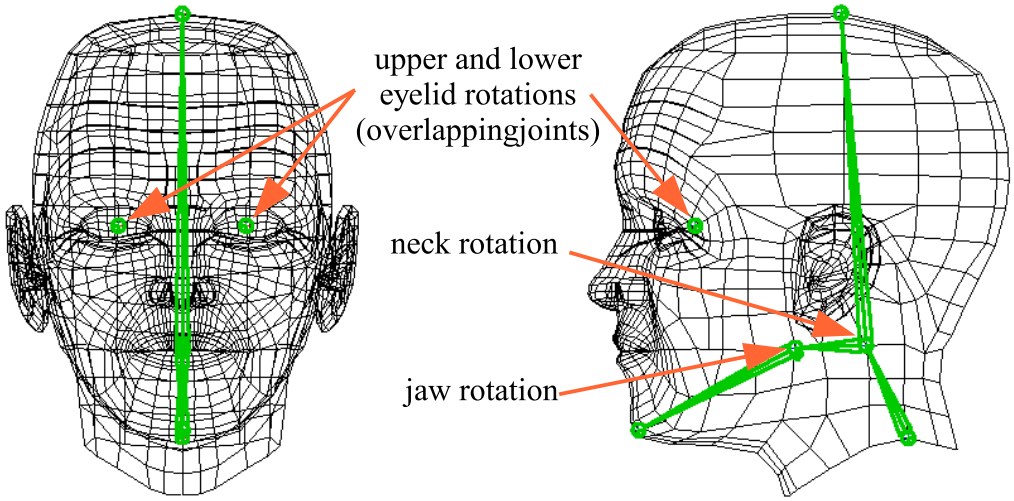



**Figure 4.32: Joints used for Murphy's skinning deformation.**

a single joint for the head rotations and another for the jaw rotations. There are also a total of four eyelid joints, one per eyelid. We use smooth binding to attach the skin to the joints, limiting the maximum number of joint influences to two per vertex. Although this is a small number, in practice we obtain good results due to the simplicity of our joint skeleton.

With the exception of the eyelid joints, the bone joints are constructed by taking the midpoint of selected vertices on the face mesh. The eyelid joints themselves are positioned at the pivot points of the eye shapes. To maintain the topological parameterization, we store the skin weights per joint based on the vertex indices in the face mesh. Figure 4.33 shows the results from applying the smooth skin deformation on Murphy.

The inquisitive reader may wonder why this deformation comes last. Since the skin and muscles stretch as bones rotate (for example, when opening the mouth), would it not make more sense to place this deformation at the beginning of the deformation chain? Indeed, although this approach would increase the structural

**Figure 4.33: Effects of the skinning deformation.**

and anatomical correctness of our model, in practice the complexity of the system would increase dramatically. In order to produce the correct skin/muscle deformations, the muscle joints and skin wires must rotate and deform along with the facial mask, as illustrated in Figure 4.34. Consequently, the cluster/point constraint systems must be relocated as well. However, the rigging software does not allow the application of skinning deformers to locators. To work around this limitation, we opted for the simpler approach of pushing the skinning deformation to the end of the chain, which provides equally satisfactory results.

As a final detail, we note that eyelids do not really rotate about the center of the eye. As shown in Figure 4.35, this simplified approach can sometimes fail and cause intersections of the eyelid and eye shapes. We use a *sculptor* deformer to solve this problem. A sculptor deformer is essentially a push-to-surface deformer, with two differences: the surface is always a sphere and the displacement direction in not constant. Instead, all rays are traced from the same user specified location. We place the sculpting origin at the center of the eye shape, and the sculpting

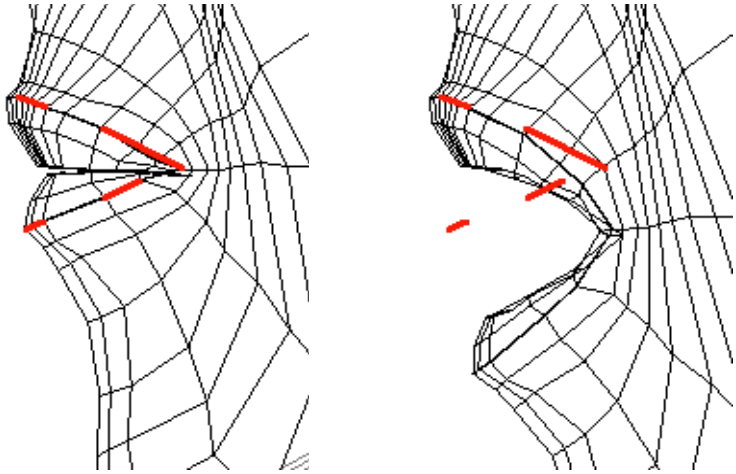


**Figure 4.34: The images above illustrate the problems of applying the skinning deformation before the muscle deformations. As the bone joints move, they must pull the muscle joints and wires along with them, maintaining the relative positions of the wires to the skin surface. Otherwise, as shown on the right, the wire deformations will not be acting at the correct skin surface locations.**

sphere is placed to match the sphere defined by the cornea. The deformation sets for the sculptors and the improved results are also shown in Figure 4.35.

### Eyes

The eye rig, which simply controls the eyes' gazing direction, is implemented using aim constraints. Given that the eyes naturally diverge between 5 and 15 degrees away from the meridian [Osi03], we do not directly aim the eye axis to the controls. Rather, we place the divergent eye shape as a child to a locator. This locator is then constrained to aim at the control shape, which is initially placed directly in front of the locator. To move the eyes together, the aim control shapes are made children to a larger control shape. Thus, the artist can move the larger shape to move the gaze direction of both eyes at once, using the smaller control for fine tuning. This is useful when the character is looking at an object near the face or to give the character a cross-eyed appearance. Figure 4.36 illustrates the details

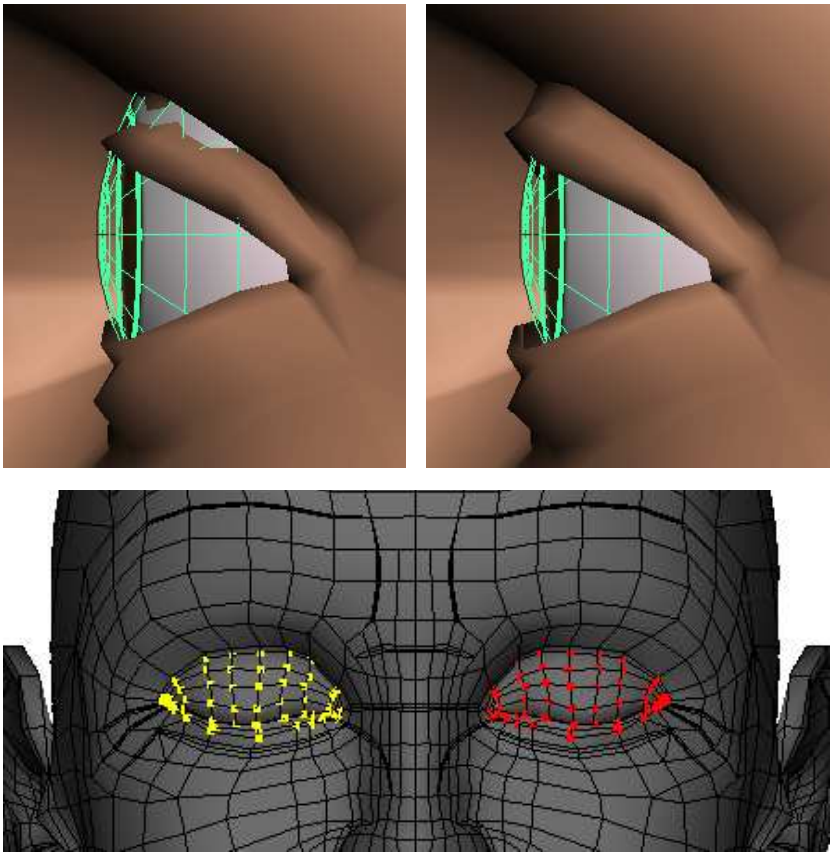


**Figure 4.35: The eyelid sculptor deformation and deformation sets. The top images show how the sculptor prevents intersections between the eye and eyelid surfaces. The lower image shows the deformation sets for these deformers.**

of the eye rig.

### Tongue

The tongue rig allows Murphy to move his tongue up and down, left to right, and in and out of the mouth. The movements are implemented with a smooth skinning deformer, using the joint skeleton shown in Figure 4.37. The figure also shows some tongue configurations achievable with this joint system.

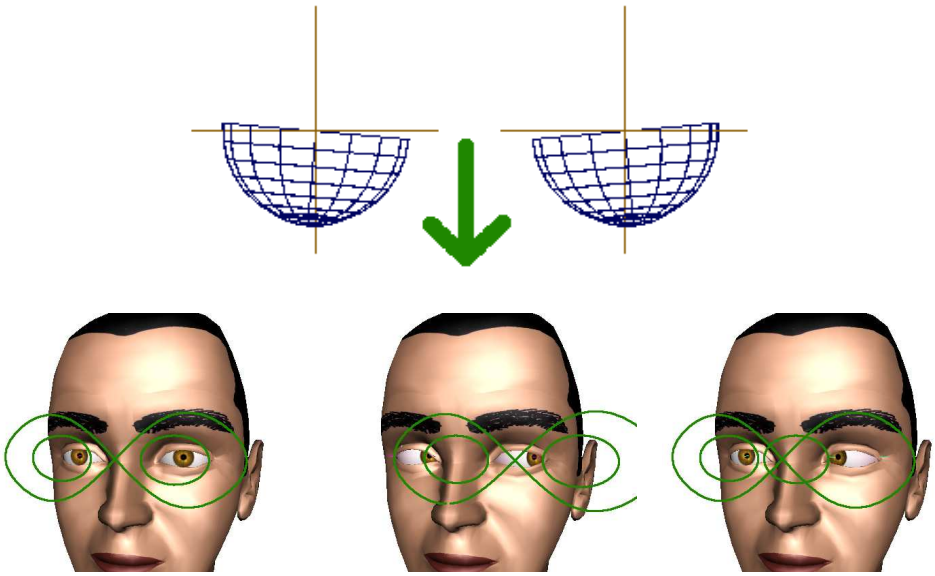


**Figure 4.36: Details of Murphy's eye rigs are shown above. The top row shows the locator shapes which are parents to the eye shapes, highlighting the diverging angle of the eyes. The parent locators are aim constrained to control shapes, located in the direction of the green arrow and shown in the bottom row. The bottom left image shows the default location of the aim controls, and the center and right images show the effects of moving the large control and the smaller controls, respectively.**

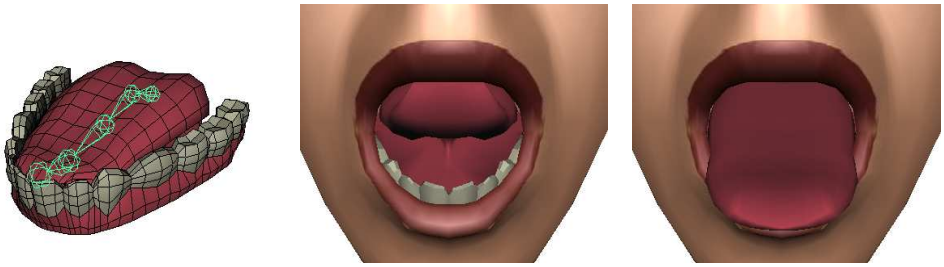


**Figure 4.37: Details of Murphy's tongue rig. On the far left the joints used for smoothly skinning the rig are highlighted in green. The center and left images show the two possible configurations of the tongue using the skinning deformer.**

## 4.4 Summary

We have presented the reference head model and rig used in our animation system. The rig uses a muscle-based parameterization to drive the deformations of the face. In order to transfer the rig easily between different head models, we have parameterized the construction of the rig on the topology of the reference surface.

An ad hoc skin/muscle model, based on wire deformers and point constraints, was developed to implement the rig. The model uses point constraints to average out the influence of various muscles on a region of skin, and the point constraint weights give the artist control over the final deformations of the skin surface, which are implemented by the wire deformers.

The head surface of the generic model has been optimized for animation. Using this surface, the skin/muscle model, and additional deformations, we have developed a realistic character: Murphy. Figure 4.38 shows Murphy posed in the six universal expressions, illustrating the flexibility of our model.

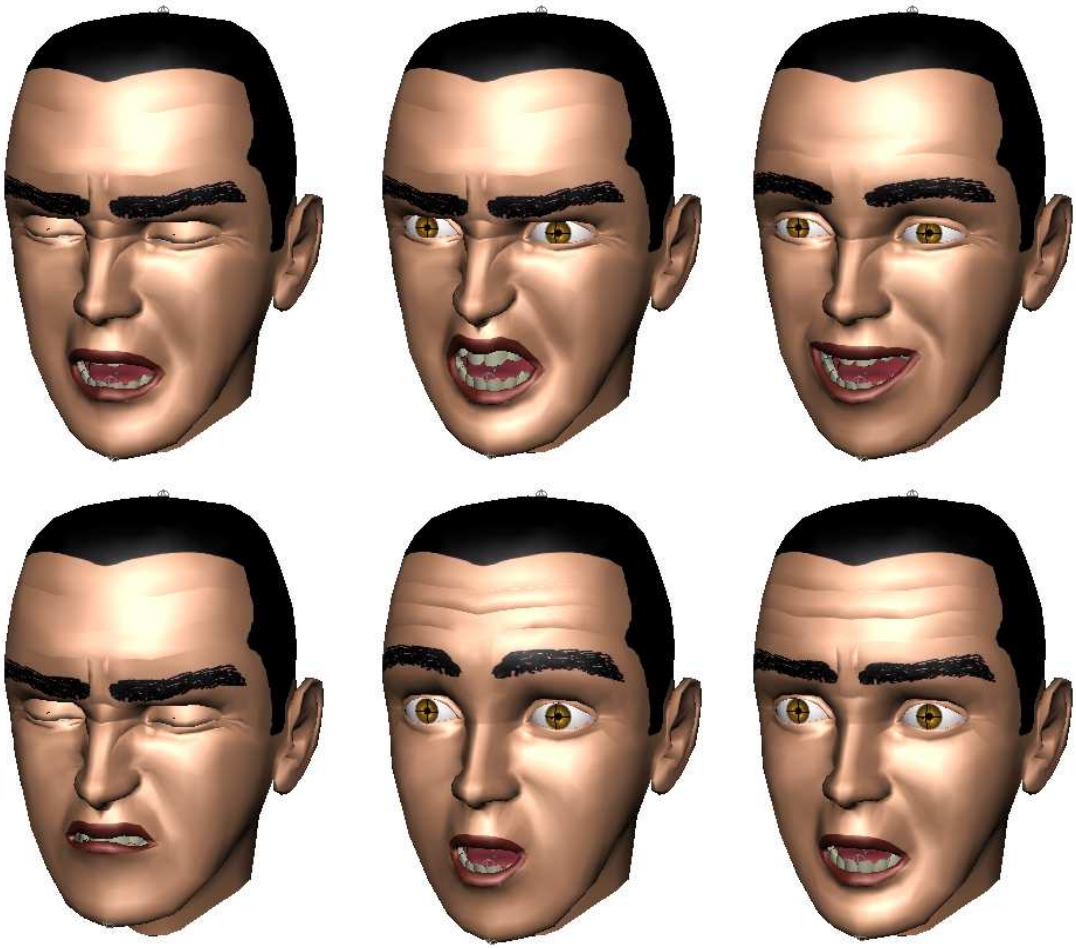


**Figure 4.38: Illustration of Murphy portraying the six universal expressions. From left to right, top to bottom, are sadness, anger, joy, disgust, surprise, and fear.**

# Chapter 5

# Fitting Facial Scan Data

Our next task is to apply the reference facial rig developed in the previous chapter to new head models. In order to do so, we deform the reference model to match the new models, maintaining the mesh topology unchanged. The reference facial rig can then be rebuilt or procedurally reattached to the new geometry, as it is parameterized on the topology only. Thus, the new head model can be quickly rigged and animated after the fitting procedure is applied. For the fitting procedure, we build on the wealth of research already conducted on the subject, which we have described in Chapter 3.

Maintaining a single topology is not a limitation, but a strength of the procedure. By transferring the reference topology, we ensure that the new surface model has a topological layout that matches the requirements of the deformations in the reference rig.

In this work, we have used digitized faces produced by a Cyberware scanner as new face models. We chose these models because they are accurate reflections of typical human faces and they are easily generated in a matter of seconds. In addition, we have based our deformation procedure on the one developed by Jeong

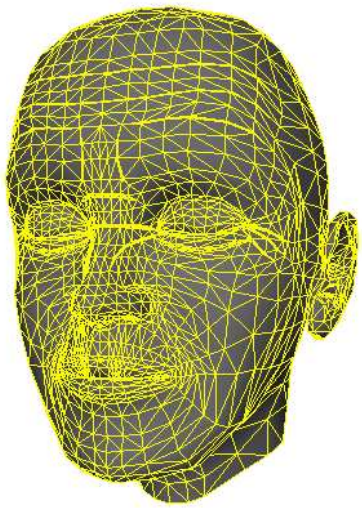


**Figure 5.1: Triangulated reference model.**

et al. [JKHS02], which requires minimal user intervention and produces fair results (see Figure 3.24). We described the details of Jeong's algorithm in Chapter 3, so this chapter will focus on the improvements of our implementation.

Jeong's algorithm is optimized for use with triangular polygonal reference meshes. Therefore, we have triangulated the original quad reference mesh to implement the fitting algorithm. The new mesh is shown in Figure 5.1. Note that this does not affect the rigging procedure described in the previous chapter. Although the edge and face topology of the reference model change with triangulation, the vertex topology, which the reference rig is based on, remains unchanged.

The remainder of this chapter proceeds as follows: first, we discuss the details of the scan acquisition procedure. Next, we describe the user interface we designed to provide an initial scan alignment. Later, we briefly review Jeong's three scan-fitting steps and then discuss our implementation. Finally, we conclude with our results. An outline of the entire fitting procedure is shown in Figure 5.2.

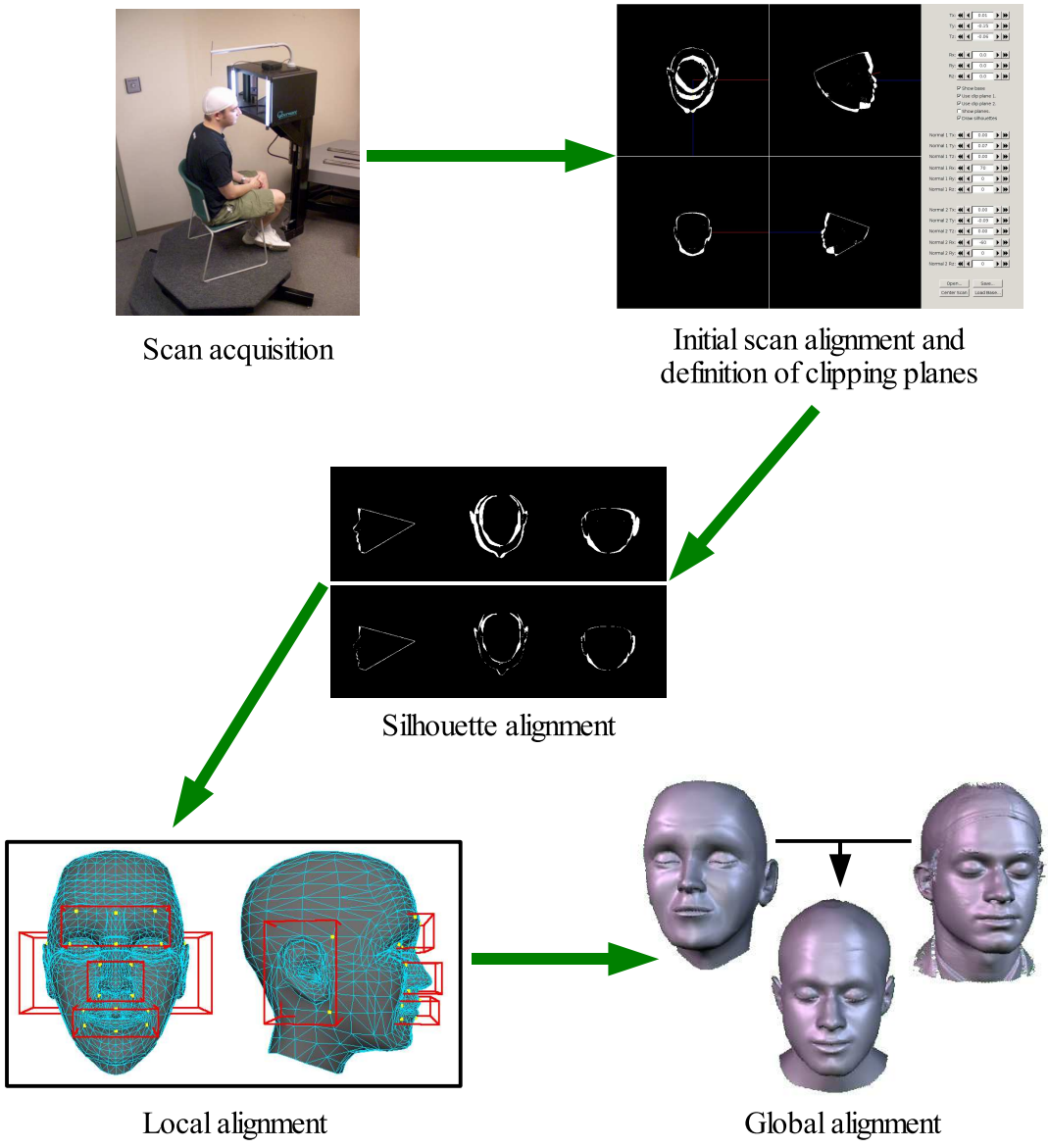



**Figure 5.2: Outline of the scan fitting procedure.**

## 5.1 Scan Acquisition

Before the scan fitting procedure is applied, we must obtain a digitized head. In this work we have used digitized facial data generated by the Cyberware 3030/RGB scanhead and PS motion system described in Chapter 3. Subjects were scanned using default settings: full circle scan with samples every 0.75 degrees. The orientation of the subjects was uncalibrated, although the *up* direction was defined by the $y$ axis. The scan output was stored in PLY format [PLY] with distances measured in meters.

The subjects were asked to close their eyes during the scan and to assume a neutral expression with a closed mouth. Since we have modeled separate eye surfaces in our rig, it is unnecessary to capture eye geometry; we capture the closed upper eyelid instead. Also, the relaxed, neutral expression matches the expression on the reference mesh.

As hinted on Figure 3.6, the subjects also wore a swim cap during the scanning procedure. The swim cap improves the scanned data by removing most of the noise and holes typical of the hair region. Additionally, our experiments have shown that, for our purposes, the fitting procedure works better when the subjects' hair is covered by the swim cap, as will be shown later in this chapter.

## 5.2 Scan Alignment

Since our scan data orientation is uncalibrated, we have developed a user interface to roughly align the scan in the reference direction. This interface is pictured in Figure 5.3. Four OpenGL viewports show orthographic projections of the three canonical axes and one perspective projection with a modifiable camera. Using the

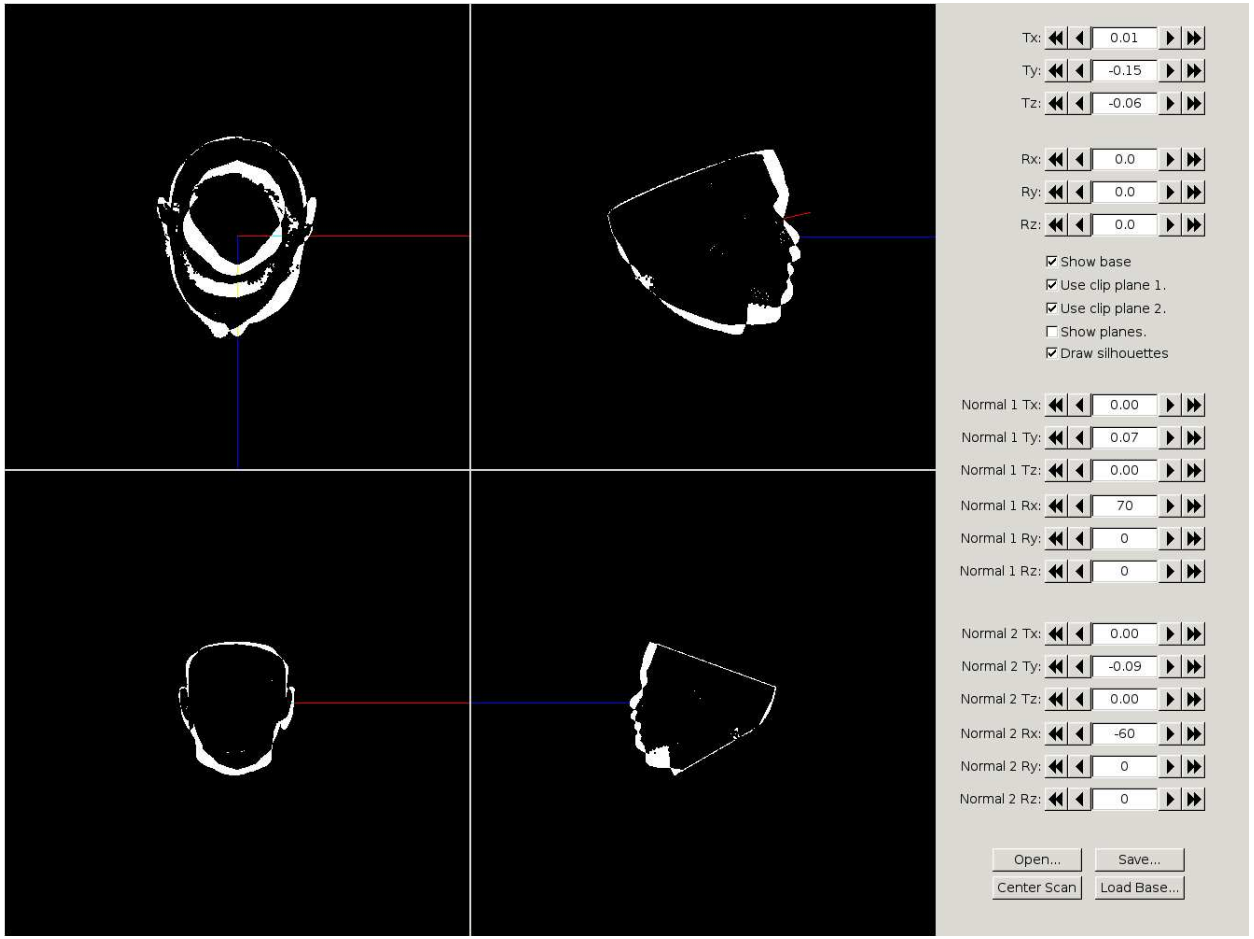



**Figure 5.3: User interface for scan alignment and definition of clip planes, showing the silhouette difference visualization with two active clipping planes. Controls and clip plane parameters are on the right-hand side of the window.**

controls on the right-hand side of the window, the user can rotate and translate the scan. A **Center Scan** button centers the bounding box of the scan at the origin, matching the location of the reference mesh. It then resizes the bounding box of the reference mesh to match the scan's bounding box. Controls allow for loading and saving the scan, for showing and hiding the reference mesh, and for visualizing the silhouette difference image.

Optional clipping planes for use in the silhouette fitting step can also be specified using this interface. The purpose of these clipping planes will be discussed in Section 5.3. A maximum of two clipping planes are supported, and they are defined by a plane normal and a point on the plane. The user is allowed to specify whether or not to use the clipping planes for rendering and whether or not to render the planes themselves.

Upon selecting **Save**, the user defined transform is applied to the scan before the file is saved. The clipping plane information is saved in a separate file, which is loaded by the next procedure.

## 5.3 Review of Jeong's Algorithm

Before we discuss our implementation of the surface fitting procedure, we briefly review Jeong's algorithm. For a comprehensive description, the reader is referred to Chapter 3 or to [JKHS02].

Jeong's fitting algorithm, which fits a triangular mesh to a set of unorganized points obtained from a facial scan, is composed of three steps: the silhouette alignment step, the local alignment step, and the global alignment step.

The purpose of the silhouette alignment step is to find an affine transform which minimizes the silhouette difference between the reference mesh and the scan points.

The silhouette difference function is defined by XOR-ing together flat-shaded renderings of the reference model and scan cloud in three canonical axes. The function value is the sum of the number of white pixels in the three XOR-ed images. Figures 3.18 and 3.19 illustrate this procedure. To avoid artificial difference pixels caused by differences in the length of the neck or by scan artifacts in the hair region, two clipping planes are used to remove these areas from the difference image. Using an iterative procedure (Powell's method), the silhouette difference function is minimized, producing an initial registration between the reference model and the scan cloud.

To eliminate differences in proportion between the facial scan and the reference model, a local alignment step is applied. This step further registers salient features in the face by applying local deformations in the areas of interest, blending the deformation into the surrounding area. The local areas are predefined for the reference model as bounding boxes around the features of interest: the ears, nose, and mouth. To find the optimal local deformations, defined as a local similarity transform, an energy minimization procedure, based again on Powell's method, is applied to the parts of the scan cloud and reference mesh within each box. For convenience, we restate the energy functionals here. For each alignment box $b \in B$ we let $V_b$ and $P_b$ represent the sets of reference vertices and scan points inside of $b$, respectively.

$$E_{local}(V_b, P_b) = E_{dist}(V_b, P_b) + E_{stretch}(V_b) \tag{5.1}$$

$$E_{dist}(V_b, P_b) = \sum_{p \in P_b} \|p - \Pi(V_b, p)\|^2 \tag{5.2}$$

$$E_{stretch}(V_b) = \sum_{e \in \mathsf{edges(V_b)}} \frac{1}{2} k \|l_e - r_e\|^2 \tag{5.3}$$

Here, $E_{dist}$ penalizes the distance from the scan points to the reference mesh, and

$E_{stretch}$ penalizes large changes in the reference mesh's scale. Diagrams explaining these terms can be found in Figures 3.15 and 3.22. The optimal transforms are then applied locally at each alignment box and blended into the surrounding region. The blending displacement for a given vertex is calculated by combining the displacements of predefined landmarks in each bounding box, weighted according to the distance from the vertex to the landmark.

Finally, the global alignment step produces the final polygonal model from the scan cloud. This step is also based on an energy minimization procedure which optimizes the positions of the vertices in the reference model. We repeat the energy function below, using $V$ and $P$ as the sets of vertex positions and scan cloud points.

$$\begin{aligned} E_{global}(V,P) &= E_{dist}(V,P) + \lambda E_{smooth}(V) \\ &\quad + \mu E_{binding}(V) + \nu E_{constraint}(V) && (5.4) \\ E_{smooth}(V) &= \sum_{v \in V} \left\| v - \frac{\sum_{w \in \mathsf{star(v)}} w}{\mathsf{valence}(v)} \right\|^2 && (5.5) \\ E_{binding}(V) &= \sum_{(u,v) \in \mathsf{binding}(V)} \|u - v\|^2 && (5.6) \\ E_{constraint}(V) &= \sum_{v \in \mathsf{constrained}(V)} \|v - \hat{v}\|^2 && (5.7) \end{aligned}$$

The $E_{dist}$ term is given by Equation 5.2. The term $E_{smooth}$ encourages the resulting polygonal surface to be locally smooth by penalizing the distance between a mesh vertex and the centroid of its neighbors, as shown in Figure 3.16. The $E_{binding}$ term prevents certain areas from being flattened by the $E_{smooth}$ term, such as the back of the ears. It is also useful to keep the boundaries at the mouth and eyes closed. Finally, the term $E_{constraint}$ is useful to maintain the vertices of the inner lips and eyes at their current locations instead of allowing them to be pulled out to the scan surface by the $E_{dist}$ term. The constants $\lambda$, $\mu$, and $\nu$ are arbitrary and control the influence of each energy component in the final result. By approximating $E_{dist}$ as

a linear term, the minimization problem reduces to a linear least squares system which is easily solved to obtain the optimal vertex positions for the reference mesh.

## 5.4 Silhouette Alignment

We have implemented Jeong et al.'s initial silhouette alignment step without any significant changes. However, upon loading the scan, we scale it uniformly by 100 to convert the units to centimeters, which are more convenient to work with in the rigging software.

For each canonical viewing direction, we create an 256x256 image, which is the resolution recommended by Jeong. Although we tried rendering these images directly into a frame buffer window, we found that other windows on the desktop affected the result of the `glHistogram()` count. Therefore, the images are rendered into a pixel buffer instead. The implementation of Powell's method was taken from [PTVF02]. On our system, which consisted of an Intel® Pentium® 4 processor [Int] running at 2.4GHz and an NVIDIA® Quadro®4 750 XGL card [NVI][1], the optimization procedure took an average of ten minutes per scan.

Figures 5.4 and 5.5 shows results for this procedure, which we have found not to be robust. In most cases, a fair alignment is obtained, but occasionally, as shown in Figure 5.5, disastrous results occur. We have not investigated methods for solving this problem. Instead, we have painstakingly tweaked the clipping planes for each scan until we achieved a decent alignment. In some cases, an alignment which produced accurate results after completing all of the steps in the fitting process

[1]Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. NVIDIA and other NVIDIA Marks are registered trademarks or trademarks of NVIDIA Corporation in the United States and other countries.

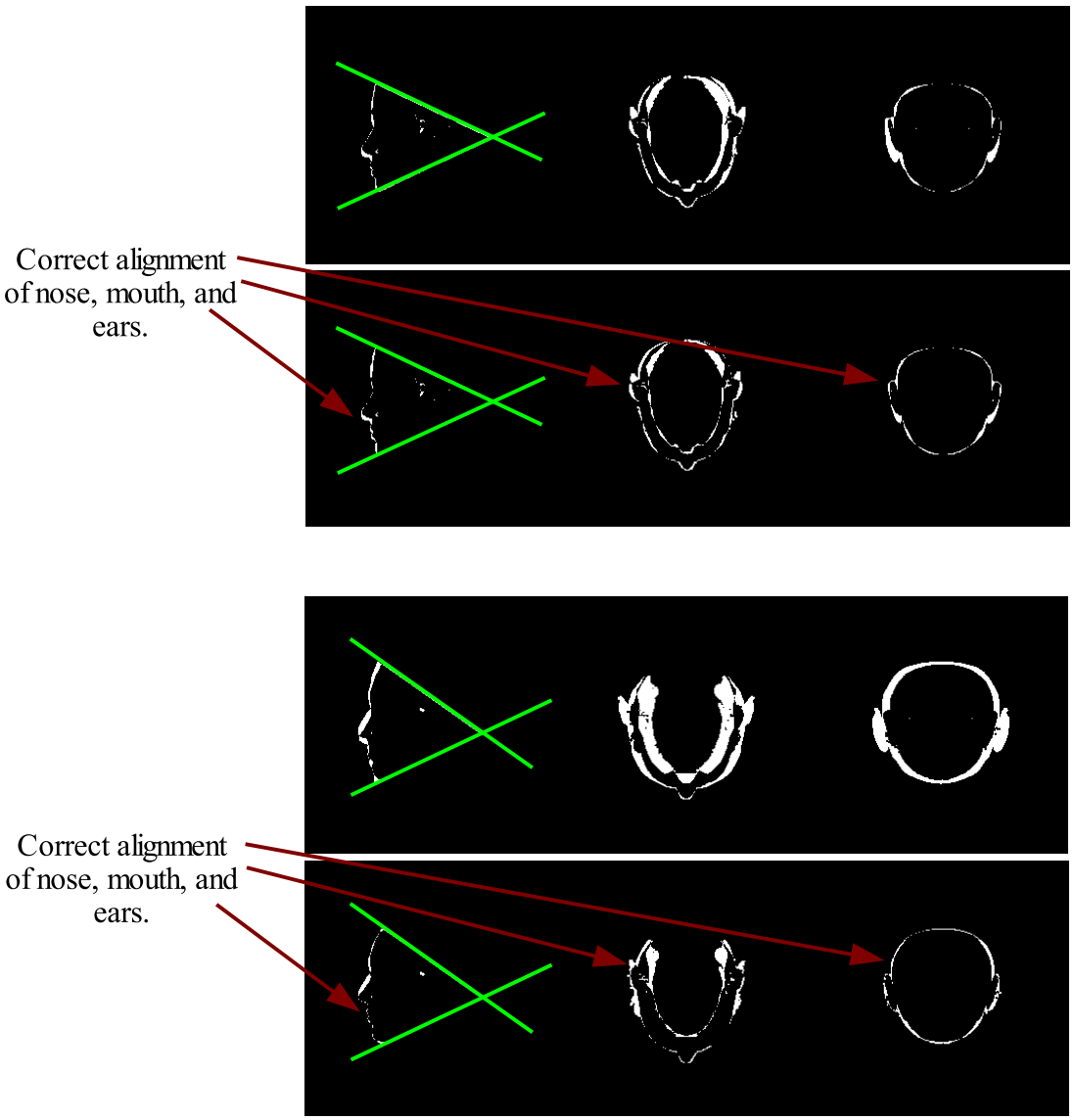



**Figure 5.4: Correct results from the silhouette alignment procedure applied to Dhruva's (top) and Marissa's (bottom) facial scans, proceeding from top to bottom. For each scan, the initial silhouette difference is presented first and the final below it. In each row, three images corresponding to the three canonical views are presented. The green lines show the locations of the clipping planes used for each scan.**

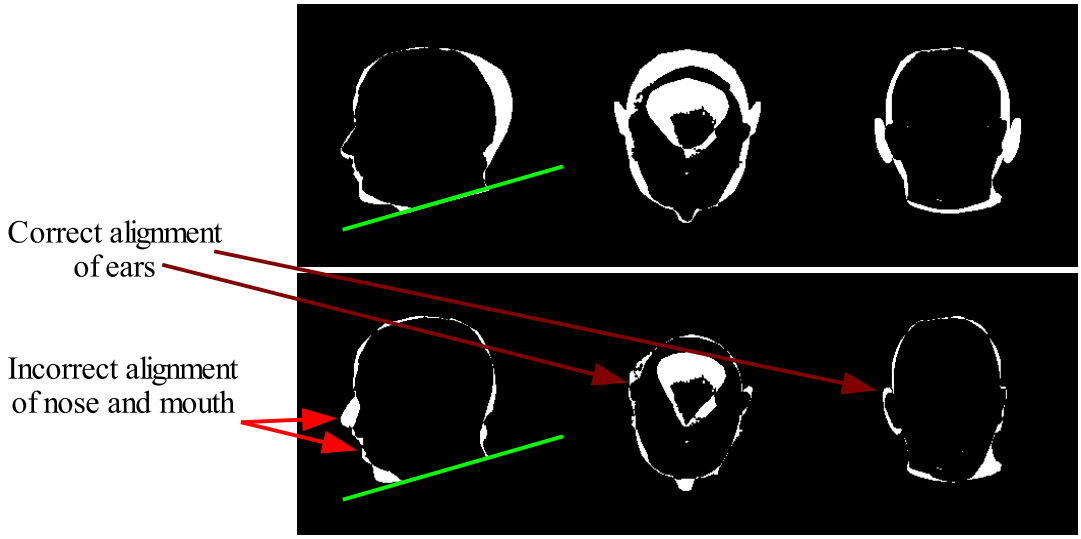



**Figure 5.5: Results from the silhouette fitting procedure applied to Sebastian's scan, producing incorrect results. Again, the initial silhouette difference is presented above the final difference. Although the number of white pixels is at a minimum in the final count, notice that the ears fit together but the nose and mouth regions have moved away from the desired orientation.**

was not found.

For those scans with good alignment, we stored the optimal transform $\tau$ and applied it to the reference model before continuing on to the next step.

## 5.5 Local Alignment

Our implementation of the local alignment step differs slightly from Jeong's implementation. Although we have used the same energy functionals described in Section 5.3, we evaluate them differently.

We have chosen to use five local alignment boxes instead of the four proposed by Jeong et al. The boxes and corresponding landmarks are shown in Figure 5.6. The new box is used to align the eyebrow regions of the scan and the reference mesh, accounting for differences in proportion in the middle third of the skull.

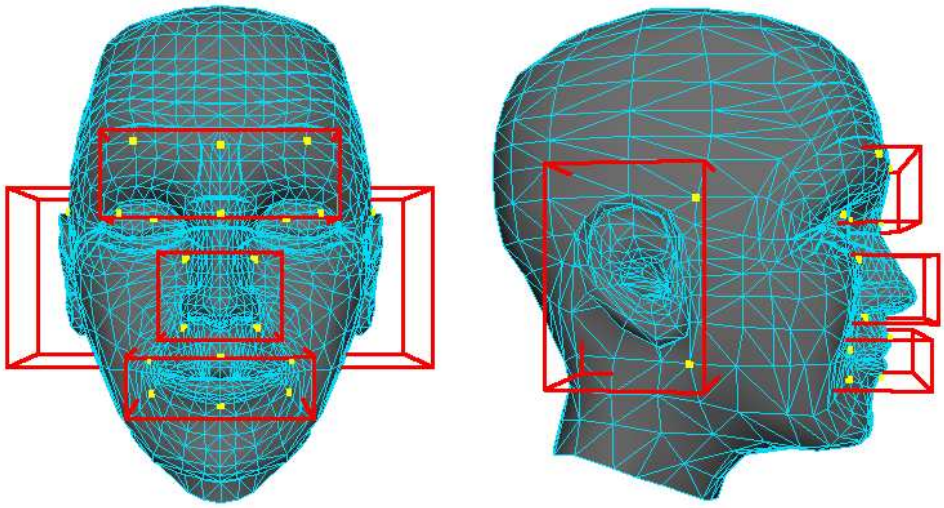


**Figure 5.6: Local alignment boxes defined on our reference model. Note that we have used five alignment boxes instead of the four suggested by Jeong (see Figure 3.21). There is one box for each ear and one box each for the nose, mouth, and eyebrows. The alignment landmarks are highlighted in yellow. Note that the third landmark for the ear region is occluded by the ear.**

Without this box, as shown in Figure 5.7, the boundary between the eyelids will sometimes creep to the middle of the eye, giving the final fitted mesh an unnatural look. We have also moved the alignment landmarks for each box to areas near the box boundary, since this is where the exponential blending function should have a stronger effect. We used three landmarks for each ear, four for the nose, six for the mouth, and eight for the eyebrows.

Recall that Jeong's formulation of the $E_{dist}$ term, unlike Marschner's, allows points in the scan to project to an area of the reference mesh whose normal points in the opposite direction (see Chapter 3). To overcome this discrepancy, we recall that the Cyberware scanner cannot capture internal or self-occluding surfaces of the face, such as the interior of the lips and the area behind the ears. Therefore, we do not allow points in the scan to project to such occluded regions in the reference model. These non-projectable regions are shown in Figure 5.8 and include the

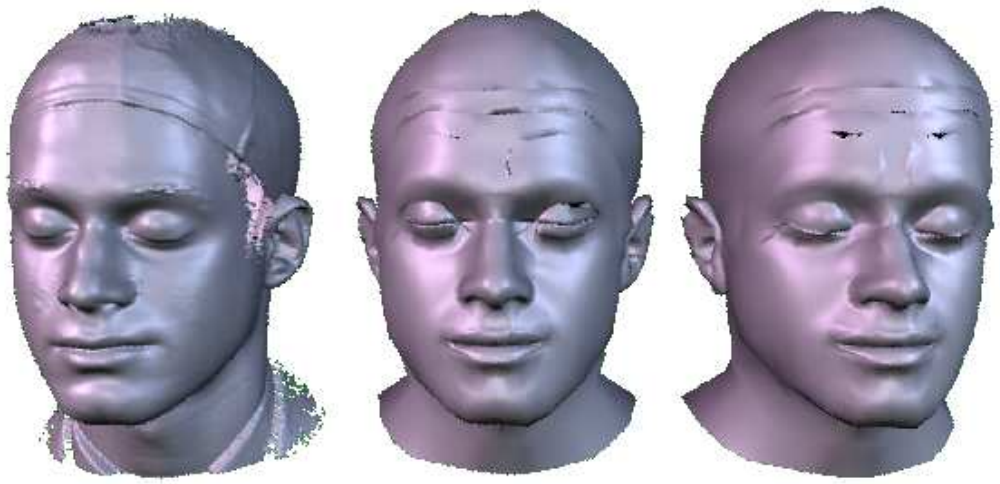


**Figure 5.7: Comparison of results of using four versus five local alignment boxes. On the left, we see the original Cyberware scan of Dhruva for comparison to the fitted surfaces. The shape in the center shows the eyelid boundary at the middle of the eye, which looks unnatural. Using a fifth alignment box, we push this boundary down, obtaining the shape on the right.**

region behind the ears and the interior parts of the eyes, nose, and mouth.

To speed up the computation of projections for the $E_{dist}$ term, we build a bounding sphere hierarchy and use the closest point algorithm developed by Maier et al. [DMM03]. The hierarchy is applied to the triangles which have at least one vertex within the alignment box and are not in the set of non-projectable faces. Like Jeong, we then apply Powell's method to find the optimal transform. However, we have found that evaluating $E_{dist}$ accurately for each evaluation of $E_{local}$ is too slow to be of practical use. Following Marschner, we have therefore approximated the term by expressing the projected points $\Pi(V, p)$ as linear combinations of the vertices in $V$. In practice, we use the barycentric coordinates determined from the triangle on which the projected points lie.

Again following Marschner, we improve upon this approximation by iterating

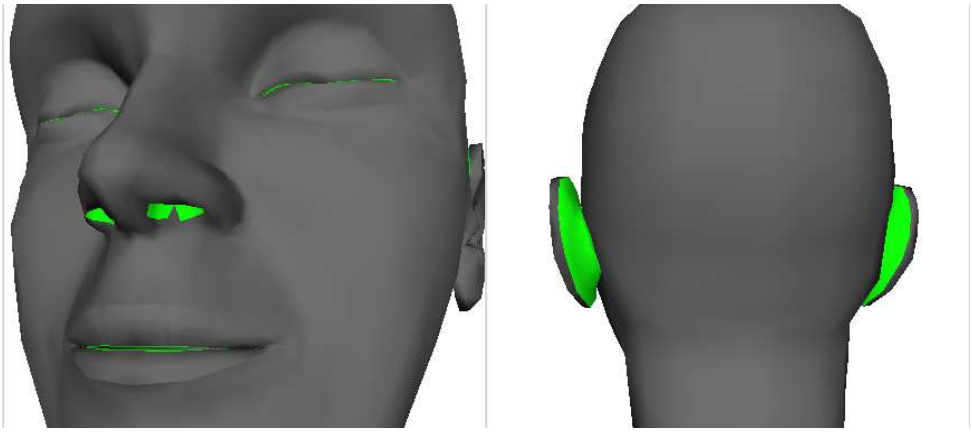


**Figure 5.8: The non-projectable regions of the reference mesh are highlighted in green in this figure.**

over calls to Powell's method, computing new approximations of $E_{dist}$ before each call. After calling Powell, we compare the final energy value $E_{local}$ to the one obtained in the previous iteration, halting when the difference between these values is less that one percent of the current value. Since this condition alone does not ensure termination, we limit the number of iterations to fifty.

Once the transforms for each alignment box have been found, we apply them to the reference mesh and use Jeong's blending formula (Equation 3.9) in the surrounding regions. A diagram of the entire procedure is shown in Figure 5.9.

## 5.6 Global Fitting

Our implementation of the global fitting step also differs slightly from Jeong et al. Recall that this step performs the final optimization of the reference vertex positions using an energy minimization procedure.

The binding edges and constrained points we have specified, shown in Figure 5.10, follow Jeong's specification. We use binding edges in the ear region to prevent the back of the ear from smoothing out and collapsing the ear ridge. In

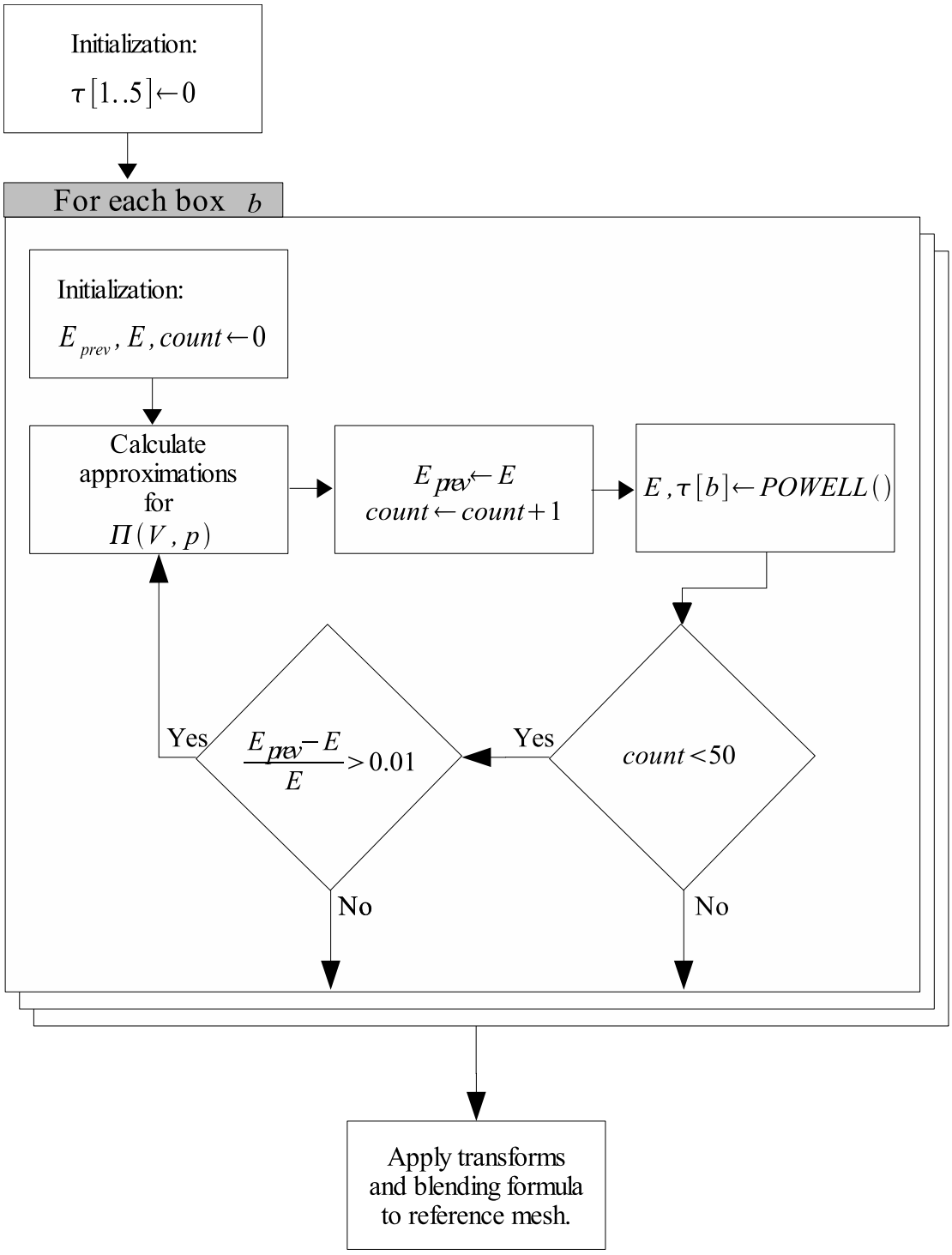



**Figure 5.9: Schematic diagram of our implementation of the local alignment step.**

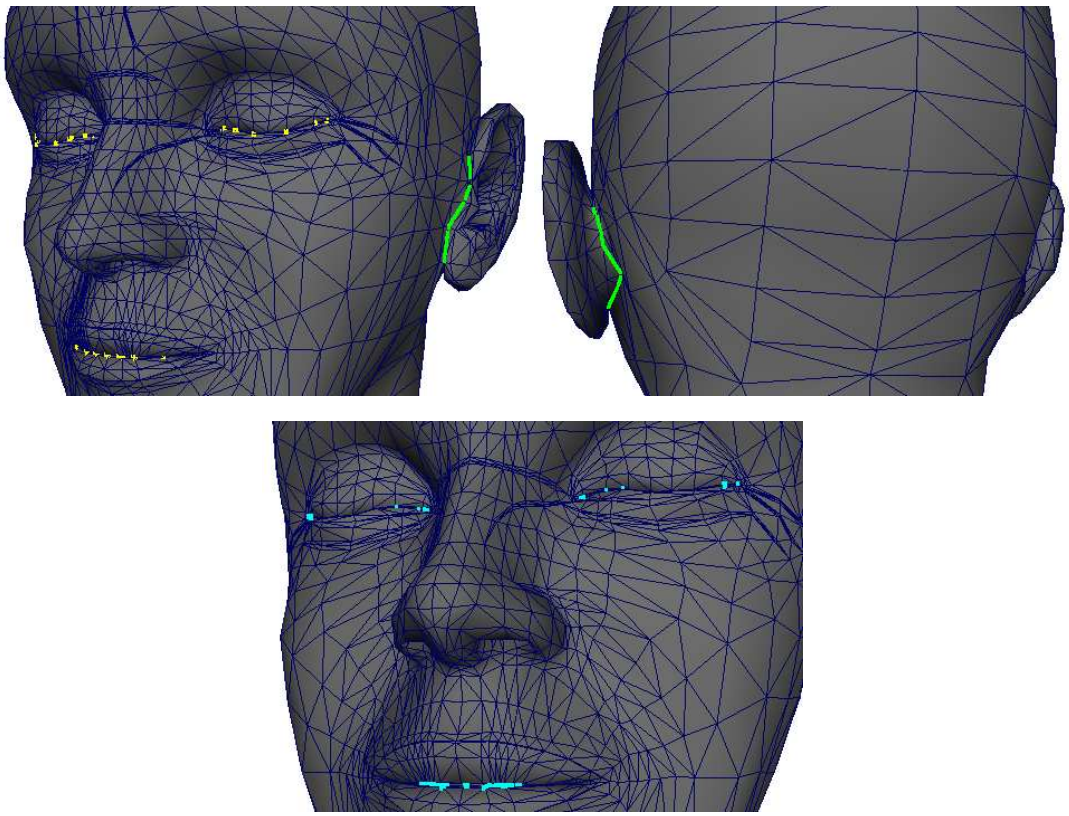


**Figure 5.10: Binding edges and constrained points used in the global fitting step. The two images above show the binding edges that are actually edges in the reference mesh highlighted in green. The endpoints of the non-existent binding edges are highlighted in yellow in the same images. The lower image shows the constrained vertices in light blue. Note that none of the constrained vertices is also an endpoint for a binding edge, since the former lie further away from the visible surface.**

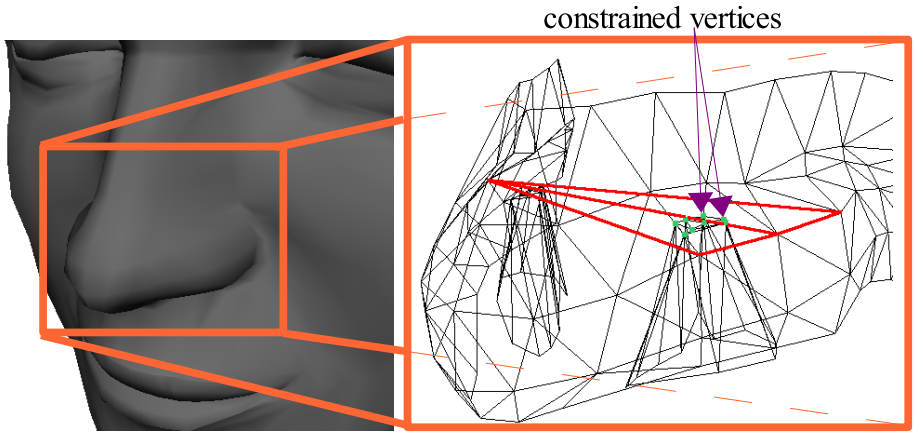



**Figure 5.11: Barycentric constraints defined in the interior nose. The wireframe image on the right is a blow up of the nose region, including some of the internal nose faces. The vertices shown in green are projected onto the red triangles, which are defined from vertices in the reference mesh. The green vertices are then constrained to the barycentric location of their projection. A symmetric setup is used for the other side of the nose.**

the boundaries of the eyes and mouth, we specify binding edges which are not part of the mesh to keep the borders closed. Constrained points are used to keep the inner points in the eye and mouth boundaries from being pushed toward the outer surfaces.

We also considered constraining the vertices in the interior of the nose, but we were afraid that the constraint would allow the vertices to poke through the exterior nose surface. However, we needed some form of constraint to prevent the smoothness term from flattening out the interior nose region. We therefore developed a *barycentric* constraint, which is shown in Figure 5.11. For each barycentrically constrained point, we chose three other vertices on the reference mesh, forming a triangle. We then constrain the vertex to the barycentric coordinates of the closest point on this triangle. This is similar to the linearization of the $E_{dist}$ term.

Mathematically, using the variables defined in Figure 5.12, we define a barycen-

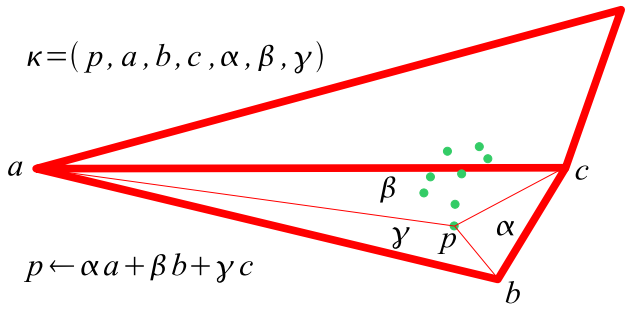



**Figure 5.12: A graphical explanation of the definition of a barycentric constraint $\kappa$. The triangles and constrained vertices are those shown in Figure 5.11. Here, $\alpha$, $\beta$, and $\gamma$ represent the areas of $\triangle pbc$, $\triangle pca$, and $\triangle pab$, respectively.**

tric constraint $\kappa$ as a 7-tuple

$$\kappa = (p_\kappa, a_\kappa, b_\kappa, c_\kappa, \alpha_\kappa, \beta_\kappa, \gamma_\kappa), \tag{5.8}$$

where $p_\kappa$ is the constrained vertex, $a_\kappa$, $b_\kappa$, and $c_\kappa$ are the reference vertices defining the constraining triangle, and $\alpha_\kappa$, $\beta_\kappa$, and $\gamma_\kappa$ are the corresponding barycentric coordinates for the projection of $p_\kappa$ unto triangle $a_\kappa b_\kappa c_\kappa$. The barycentric energy term penalizes the distance between $p_\kappa$ and its projection, and is defined as

$$E_{barycentric}(V) = \sum_{\kappa \in \text{bary}(V)} \|\alpha_\kappa a_\kappa + \beta_\kappa b_\kappa + \gamma_\kappa c_\kappa - p_\kappa\|^2. \tag{5.9}$$

In practice, we do not add this extra term to $E_{global}$. Instead, to all the vertices in $V$ we apply either a barycentric constraint or a smoothness constraint, since we are using the barycentric constraints to avoid over-smoothing the nose region. The modified smoothness term then becomes

$$E'_{smooth}(V) = E_{barycentric}(V) + E_{smooth}(V - \mathsf{bary}(V)). \tag{5.10}$$

We substitute $E'_{smooth}$ for $E_{smooth}$ in our final formulation of $E_{global}$.

We evaluate $E_{dist}$ once again using a bounding sphere hierarchy which excludes the set of non-projectable regions. To evaluate $E_{smooth}$, we implemented

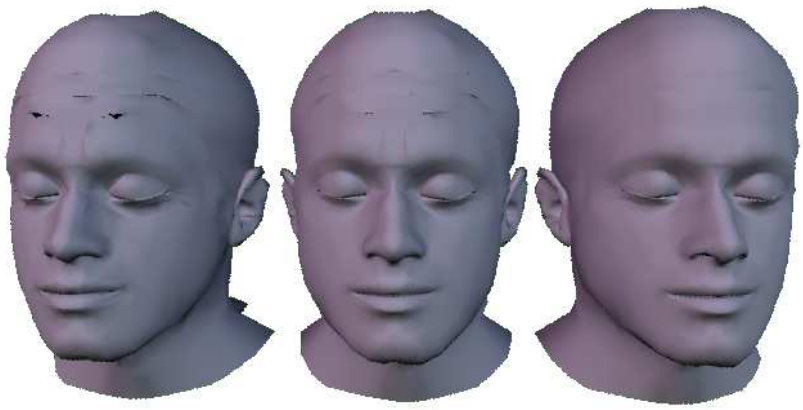


**Figure 5.13: A comparison of results from trying to smooth out the forehead during scan fitting. The shape on the left is the output from the original fitting procedure, replicated from the right of Figure 5.7. At center, the smoothness constant $\lambda$ was set to 2.0, but the forehead still shows unacceptable artifacts. After adding barycentric constraints to the forehead, the smooth shape of the right is obtained.**

the tri-edge data structure [Loo00], which stores adjacency information required to compute the function. The values of $\lambda$, $\mu$, and $\nu$ were all set to one.

Using Marschner's approximation for $E_{dist}$, the minimization of $E_{global}$ is a linear least squares problem. To find the solution, we express $E_{global}$ as a sparse linear system

$$\mathbf{A}\mathbf{v} = \mathbf{r}, \tag{5.11}$$

where $\mathbf{v}$ is a vector containing the points in $V$ (see Appendix A for a derivation). We store this system in a sparse matrix representation similar to the one suggested by Press et al. [PTVF02] and use their implementation of the conjugate gradient method to solve it[2]. Note that we actually solve three systems—one for each coordinate of the points $v \in V$. However, the matrix $\mathbf{A}$ remains the same for each system, so it is only computed once.

As shown on the far left of Figure 5.13, the results from the fitting procedure as

[2]Press et al. actually implement the biconjugate gradient method [PTVF02].

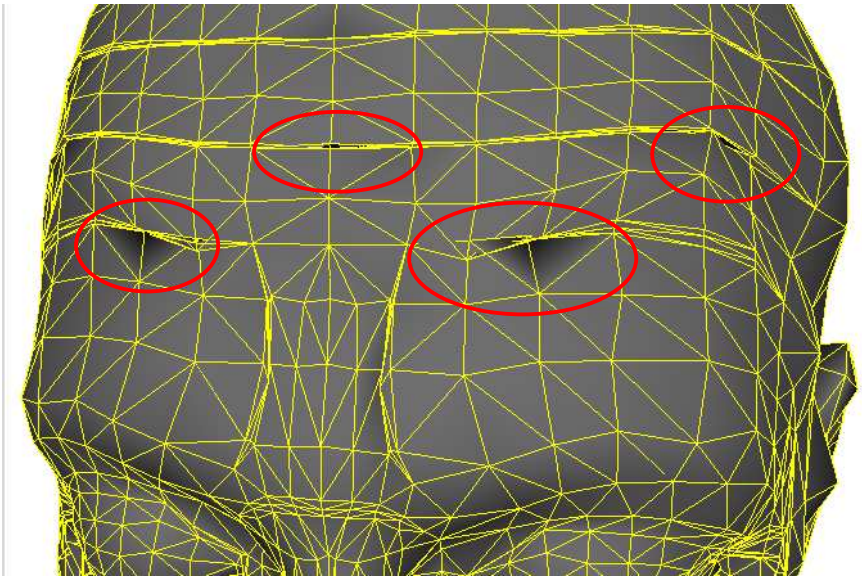


**Figure 5.14: Close-up of the artifacts in the fitted forehead shape from the far right of Figure 5.13. The areas circled in red highlight the criss crossing of vertices. Note how these artifacts are limited to the regions containing very slim triangles.**

described so far produce artifacts in the forehead region. Upon closer inspection, as shown in Figure 5.14, we see that the vertices surrounding thin triangles in the forehead are criss crossing over each other, producing odd discontinuities in shading. These vertices border the triangles which were added to the forehead to enhance skin creasing (see Figure 4.19). Increasing the smoothing constant $\lambda$ to 2.0 still does not produce optimal results, as shown in Figure 5.13.

This problem stems from the thin and long shape of the triangles in the creasing regions. Because of their shape, the number of scan points projecting onto these triangles is fairly small. However, the number of vertices which project onto the long edges of these triangles can be fairly large. This increased pull on the edges of the triangles is responsible for pulling nearby vertices in opposite directions, leading to criss crossing.

To solve this problem, we remove the thin triangles by tying the positions of the

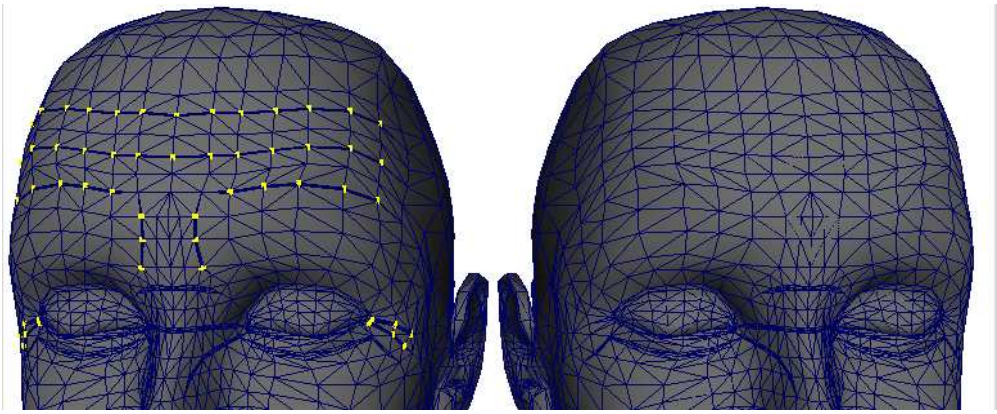


**Figure 5.15: The positions of the vertices in the forehead highlighted in yellow on the left image are tied together during the fitting procedure. The resulting mesh is shown on the right. To restore the position of the outer crease vertices, barycentric constraints are used.**

vertices shown on the left of Figure 5.15. The resulting mesh appears on the right side of the figure. We use this mesh to calculate smoothness terms for the central vertex of the crease, as shown in Figure 5.16. However, since we want to keep the thin triangles for creasing, we restore the positions of the outer crease vertices using barycentric constraints. This time we constrain the vertices to lie on the edge defined by the central crease vertex and the closest vertex on the opposite side of the crease edge. This is easily done by setting two of the vertices in a constraint $\kappa$ (say, $a_\kappa$ and $b_\kappa$) to the mentioned vertices and setting the coordinate for the third vertex ($\gamma_\kappa$, in this case) to zero.

With the addition of these new barycentric constraints, and maintaining the smoothness value $\lambda = 2.0$, we obtain the results on the right of Figure 5.13. Note how the forehead has been smoothed out with the new constraints.

Once the global alignment procedure is finished, we perform one final transformation. We take the original transform $\tau$ found during the silhouette fitting step and apply the inverse of the translation and rotation components only. This rigid transformation realigns the fitted result with the original orientation of the

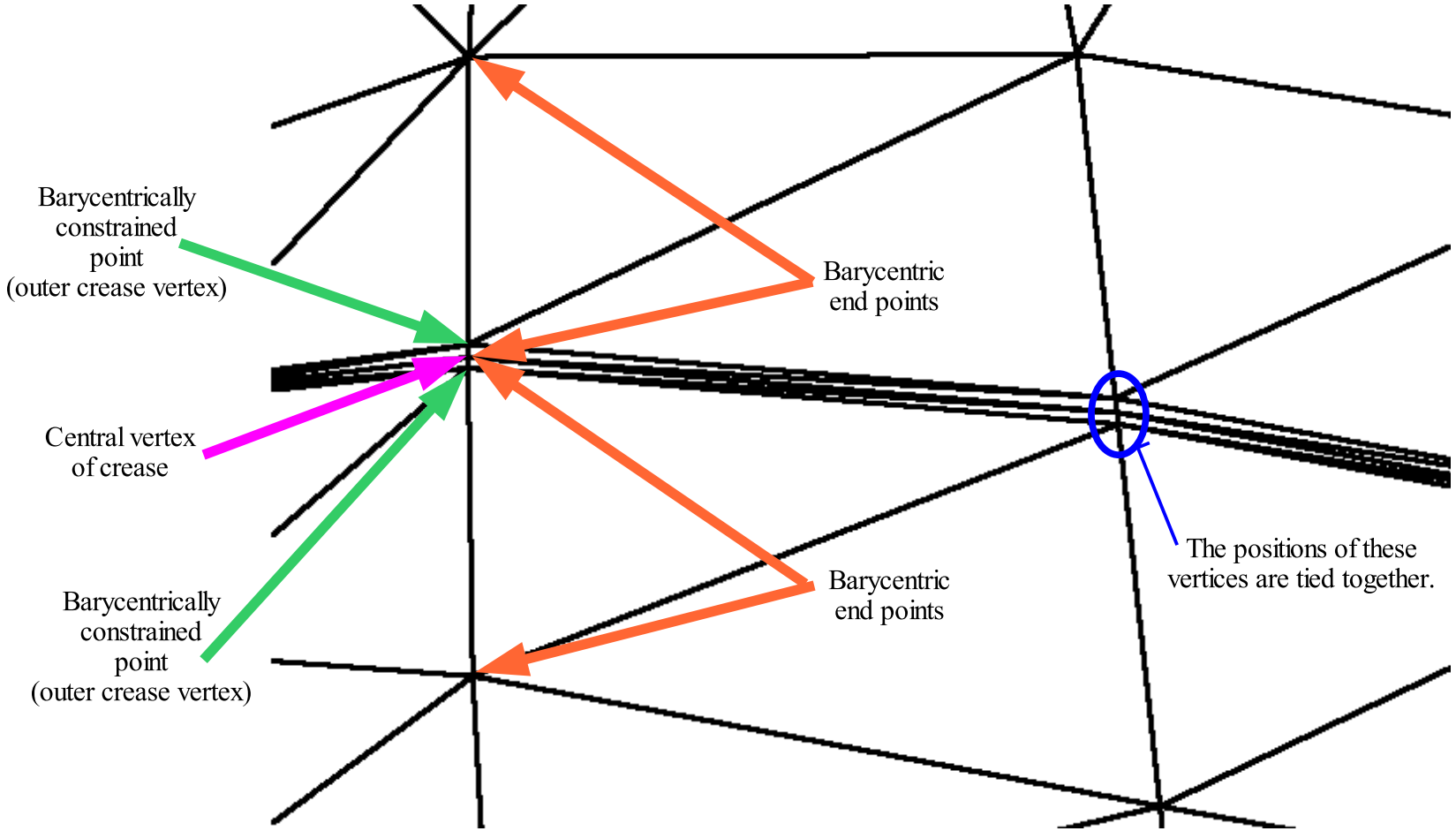



**Figure 5.16: Close-up of a forehead crease, showing which vertices are tied together, which vertices are barycentrically constrained, and which vertices are the target vertices in the constraint.**

reference mesh, allowing the rigging procedure to assume the $y$ axis points up and the $z$ axis points forward.

## 5.7 Results

The fitting procedure described above has been applied to approximately 25 scans, with mixed results. Figure 5.17 shows some of the most accurate fits. Note that the fitted surface always looks smoother than the scan, since it has fewer triangles to describe the surface.

Some issues in the fitting procedure still need to be addressed. As stated before, the silhouette fitting procedure is not very robust. This will sometimes cause problems that the local alignment cannot fix. For example, in Figure 5.18, the silhouette fit and local alignment fitted both the lower and upper lips of the reference mesh to the upper lip of the scan. The result is a mouth line placed at the wrong position.

The procedure also has trouble fitting to scans of people belonging to the Asian races. This is due to the differences in silhouettes and profiles between Asians and Caucasians. Even when the silhouette alignment produces a good fit, the local alignment box for the eyes has trouble finding an adequate alignment, as shown in Figure 5.19. This is due to the flatter region near the glabella characteristic of Asian faces.

Finally, we consider what happens to the the fitted mesh when the subject is scanned without the swim cap. As shown in Figure 5.20, the fitted mesh looks much like the scanned individual. However, upon inspection of the topology, we find that the forehead region no longer has any creases defined. The geometry added for the creases has now become a part of the hair region, making the fitted

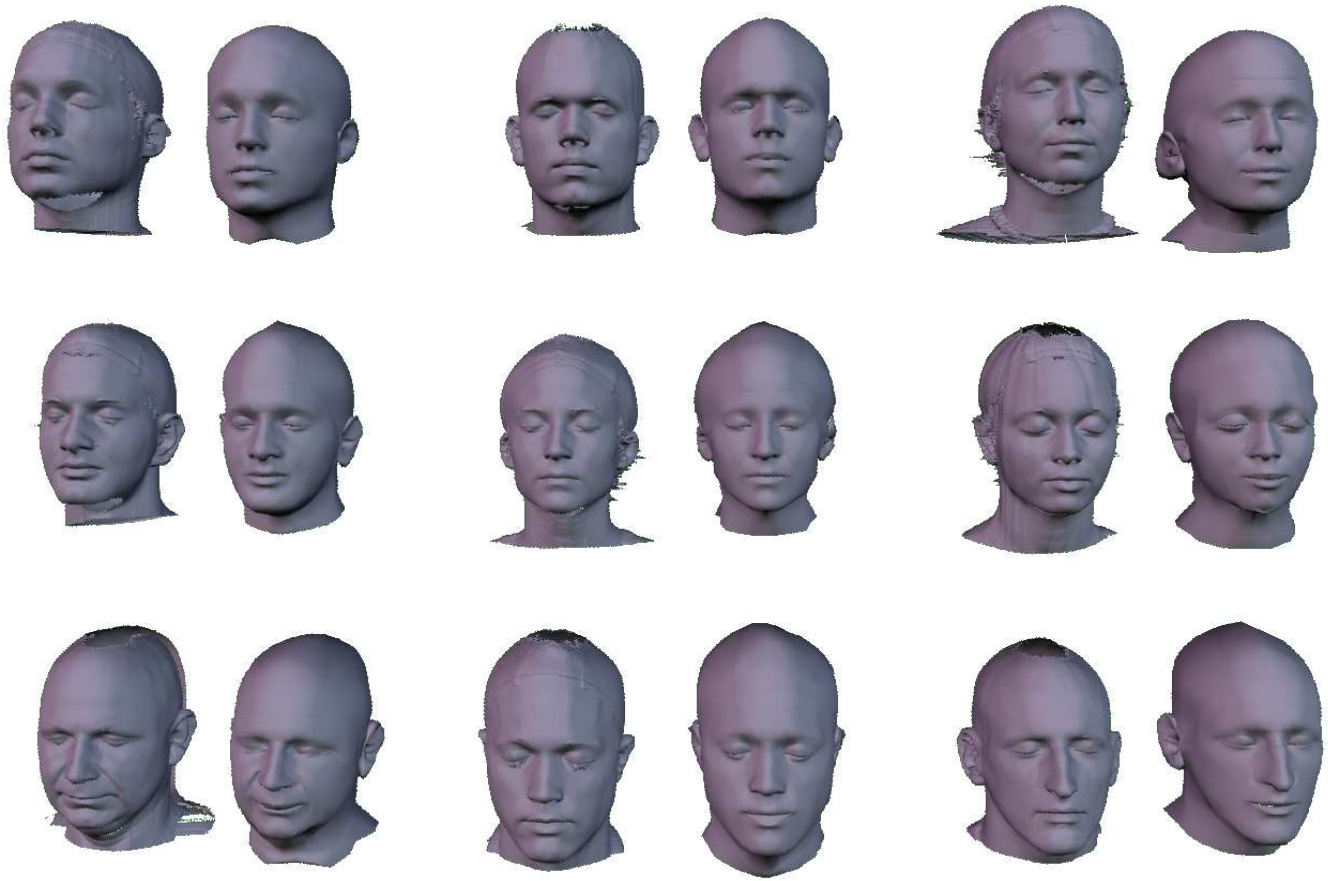


**Figure 5.17: Images of the best surface fitting results. For comparison, the original scan is shown to the left of each fitted surface.**

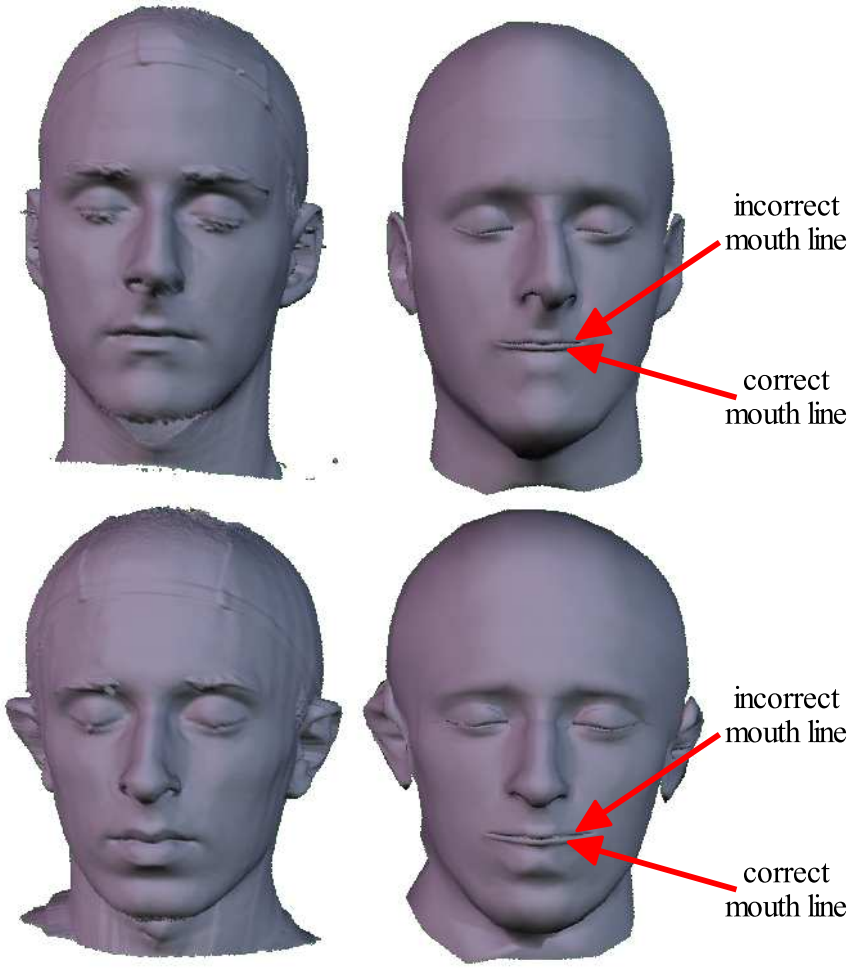



**Figure 5.18: Examples of a bad fit around the lip area. In these scans, both lips of the reference mesh are matched to the upper lip of the scan, so the line of the mouth is placed incorrectly.**

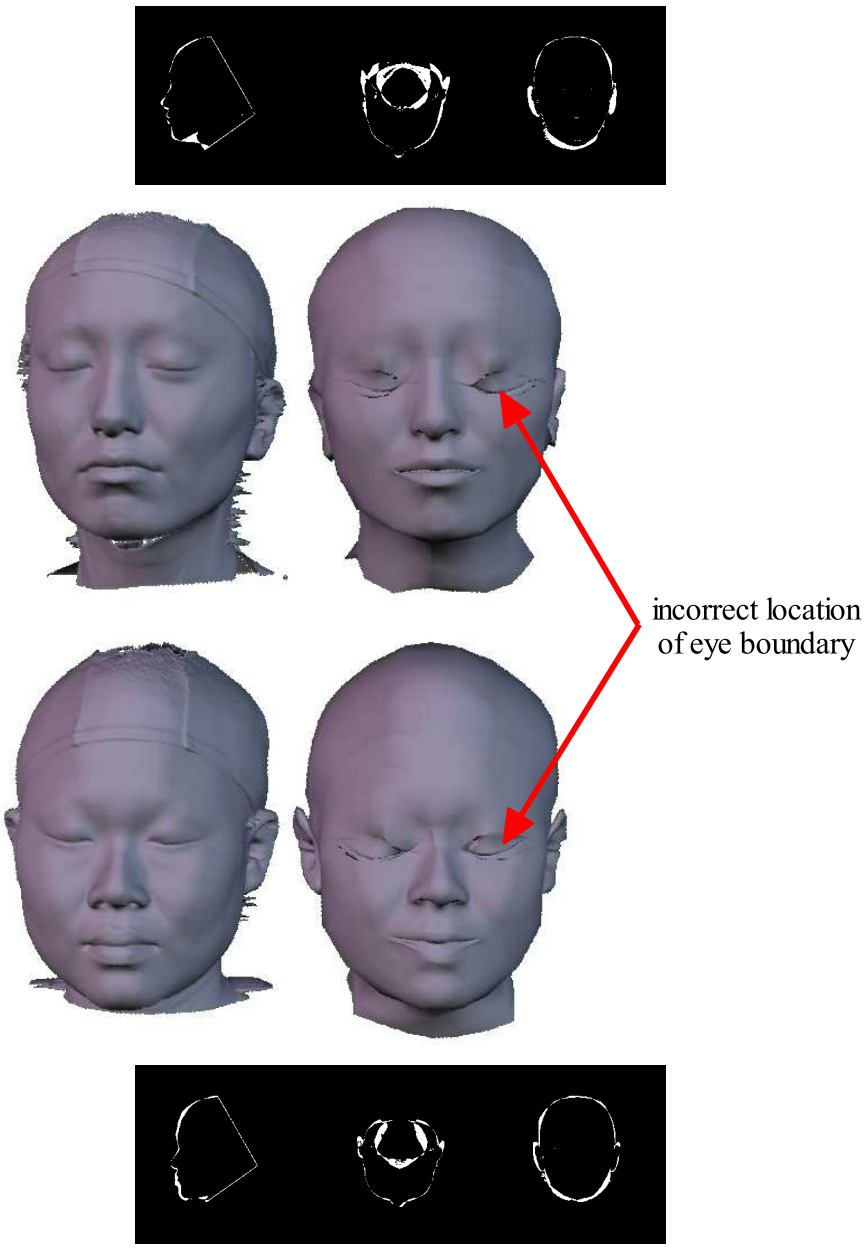



**Figure 5.19: Examples of failed fittings of Asian faces. Even though the silhouette alignment has found an adequate fit, the local and global alignment procedures fail to place the eye boundaries at the correct locations.**

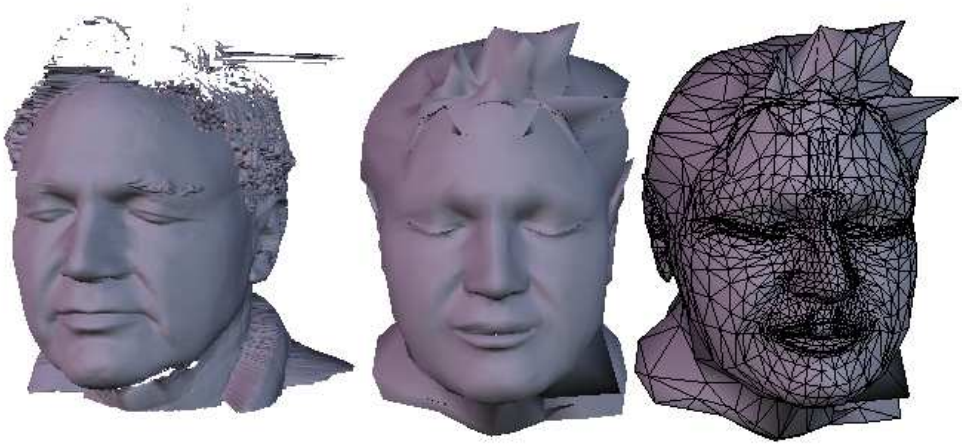


**Figure 5.20: Problems fitting scans without a swim cap. As shown at center, the fitted surface can produce accurate geometry. However, upon inspection of the mesh topology, shown at the right, we see that the creases in the forehead are no longer defined, since they have been moved up to the hair region. Therefore, this fitted surface is unsuitable for animation using the rig presented in this work.**

surface unusable for animation with the rigging system presented in this work.

In hindsight, this drifting of the topology to other regions of the mesh would disqualify it from use with a topological parameterization of the rig. By parameterizing the rig construction on the topology and deforming the reference model, we are implicitly assuming that corresponding vertices on both models will lie in the same regions of the face; for example, the tip of the nose will be the same vertex in both models, and so will the outer corner of the left eye. This property is not guaranteed by this method. A constraint-based or feature-based approach would be more useful. Additionally, better results could be obtained if the fitting procedure is limited to the front of the face, disregarding inexpressive parts of the human head (such as the ears).

Another possibility would be to allow more than one reference topology. This approach has a few drawbacks. First, the reference rig must now be parameterized

to each of the new topologies, which is a non-trivial process. Second, it is unclear if the optimal topology for a given face model can be selected before running the fitting procedure. Without such an algorithm, all topologies must be attempted, which increases the fitting time for each surface.

## 5.8 Summary

We have presented a surface fitting method suitable for deforming the reference head model to digitized human faces. The method leaves the topology of the reference model unchanged, which allows us to quickly rebuild the reference rig on the new geometry. Thus, the fitted model can be quickly rigged and animated, as will be shown in the next chapter.

# Chapter 6

# Rigging and Animating Fitted Scans

In the previous chapters we developed a reference model and rig for a generic human face. The rig has been parameterized to the topology of the reference head surface in order to rebuild or reattach it easily. We have also developed a method to deform the reference surface mesh to conform to digitized facial data produced by a Cyberware 3D scan. By combining these two systems, we are able to work on the animation of a particular person's face quickly and effectively.

In this chapter, we merge the work described previously into a functional facial animation system. The system provides interfaces which allow the user to load a fitted scan, attach the reference rig, and animate easily and effectively. These interfaces have been implemented in Alias Maya 5.0 since the reference rig was implemented using the same software.

The animation interfaces provide a convenient mechanism for manipulating the values of rig parameters. However, we have no guarantee that the rig developed for Murphy will deform fitted facial scans in the correct manner; for example, we

do not know whether setting the rig parameter values in a smiling configuration will make a fitted mesh smile. We show in this chapter that the muscle-based rig does, in general, deform fitted faces correctly. Additionally, if the artist is not entirely satisfied with the resulting expressions, he/she is free to modify them to his satisfaction. Therefore, our animation system jumpstarts the artist into the final "tweaking phase" of the modeling and rigging steps in the animation pipeline.

The remainder of this chapter is organized as follows. First, we present an interface for loading and rigging fitted scans in Section 6.1. In Section 6.2 we describe a second interface which provides a versatile and flexible animation environment. We close this chapter with a discussion of the results obtained from our system in Section 6.3.

## 6.1 Rigging the Fitted Face

Before a fitted scan can be animated, it must be loaded into the animation software and rigged. This procedure is simplified by the user interface we have developed, which provides a simple and convenient method for applying the reference rig to a fitted scan. The actual procedure consists of two steps: a loading stage and a rigging stage. The interface is likewise divided into two components, but only one of them is enabled at a particular time.

### 6.1.1 Loading the Models

The purpose of the first step of the rig setup procedure is to load all of the surface model information defining the face. The user interface for this task is pictured in Figure 6.1. Using this interface, the user selects a name for the head model and a fitted surface to load. The surface, produced from the fitting procedure described

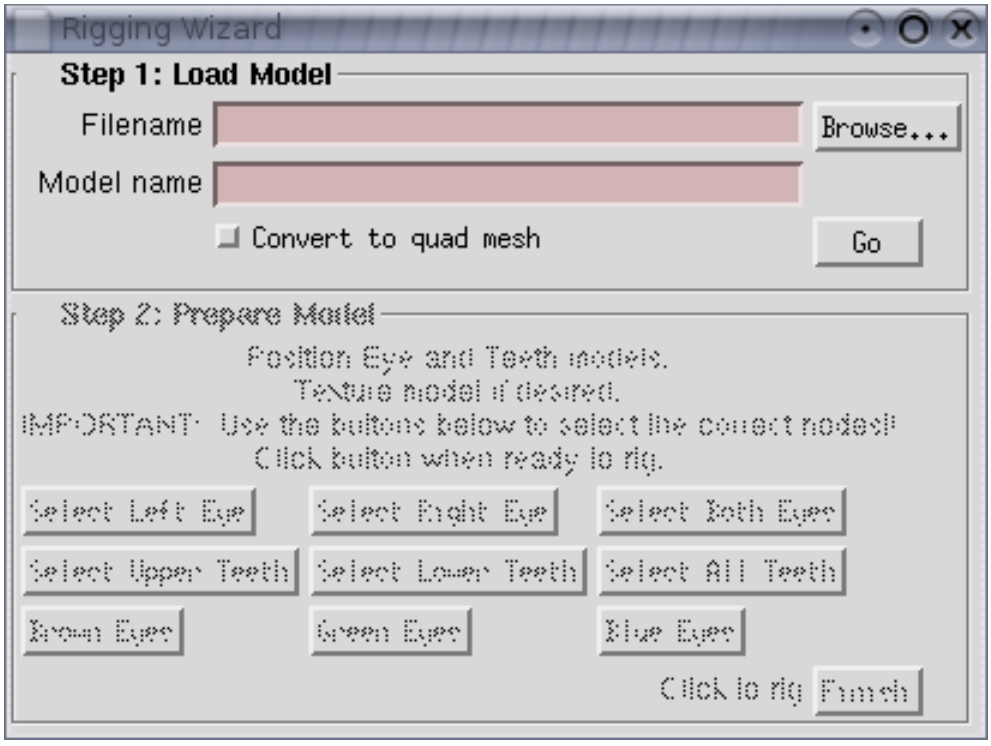



**Figure 6.1: The loading and rigging interface is pictured with the loading step enabled. At the top, the user can specify a file from which to load the fitted model. A Browse button is available to allow searching for files. Immediately below, the user specifies a name for the model. At bottom, the user can select whether to use a quadrilateral mesh or the default triangular mesh. The bottom-right Go button executes the loading step of the procedure.**

in the previous chapter, is specified as a file in PLY format. There is also a **Browse** button to allow the user to search for the desired file.

At bottom, the checkbox named **Convert to quad mesh** allows the user to select whether he/she would like a triangular mesh or a quad mesh. Recall from Chapter 4 that the reference model, Murphy, is constructed as a quad mesh, but in Chapter 5 we triangulated this model to run the scan fitting procedure. The output of the fitting procedure is therefore a triangular model. However, since the vertex topology of the two meshes is the same, we are free to apply either of the two polygonal topologies to the model, as shown in Figure 6.2.

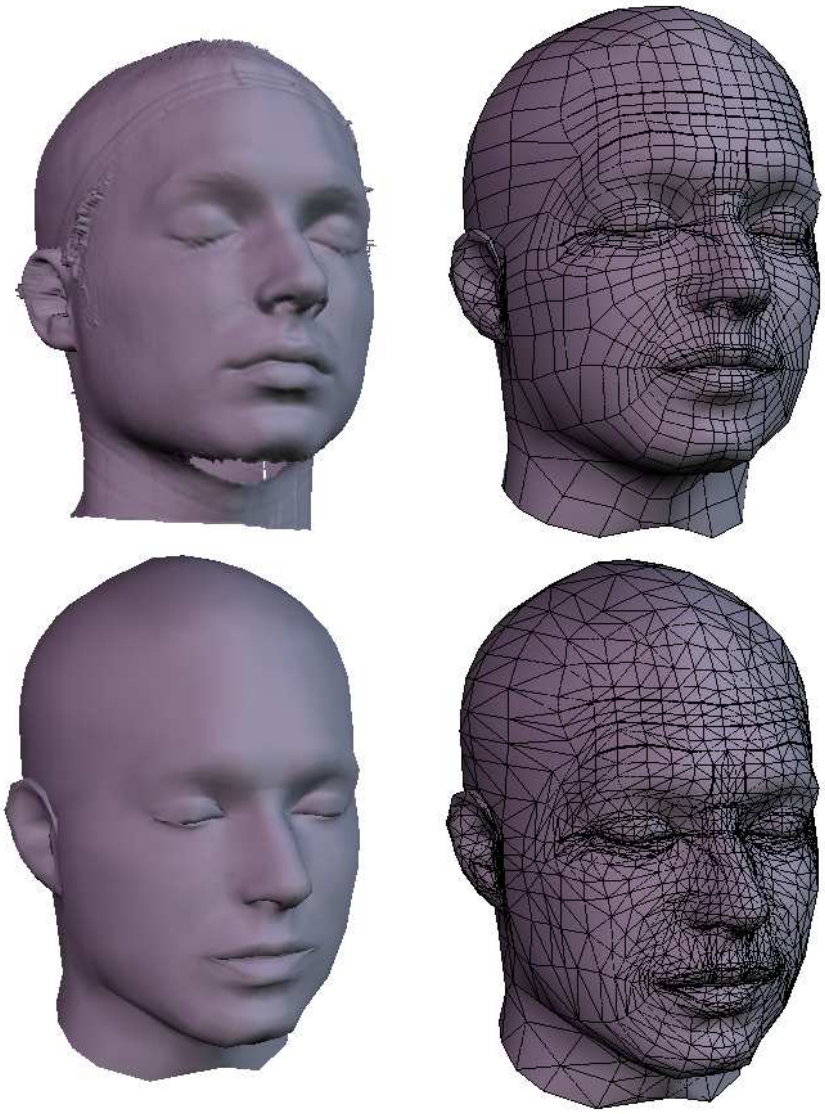


**Figure 6.2: The triangular and quad topologies applied to Adam's fitted scan. Adam's original scan appears above the fitted mesh in the left column. The right column shows both the quad mesh (above) and the triangular mesh (below).**

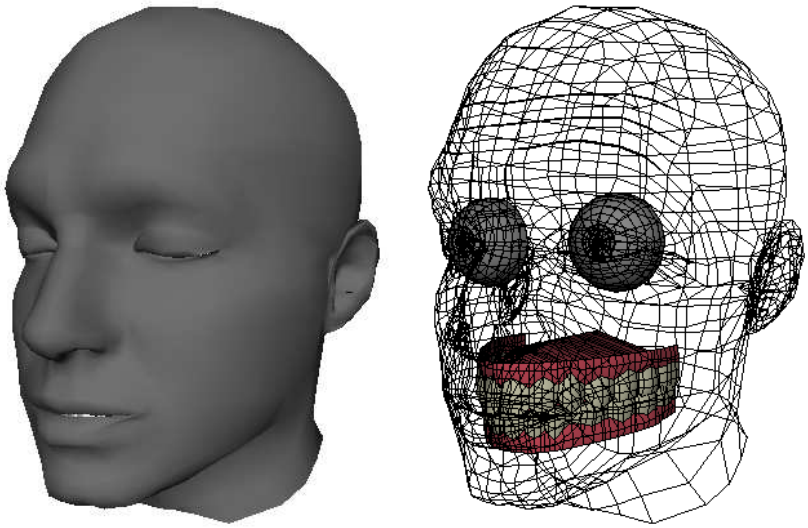


**Figure 6.3: Results from the loading procedure. The fitted model for Adam is shown on the left and in wireframe on the right, which also shows the approximate positions of the teeth and eye models.**

In practice, if the user chooses the triangular topology, the system simply loads the user-specified file. When the quad topology is chosen, the system loads both the user-specified file, which contains the default triangular topology, and a copy of the quad reference model. The vertex positions of the former model are then copied onto the latter.

Once the user has selected the desired topology, he/she can execute the loading procedure by clicking on the **Go** button at the bottom-right corner of the window. The system then proceeds as follows: first, the user specified model is loaded and the appropriate topology is applied. Next, the surfaces for the eyes and teeth are loaded along with their respective materials. These surfaces are placed in approximate locations within the head model based on the vertex topology of the fitted mesh. Finally, the user interface for the second step of the setup procedure is enabled. The resulting loaded surfaces are shown in Figure 6.3.

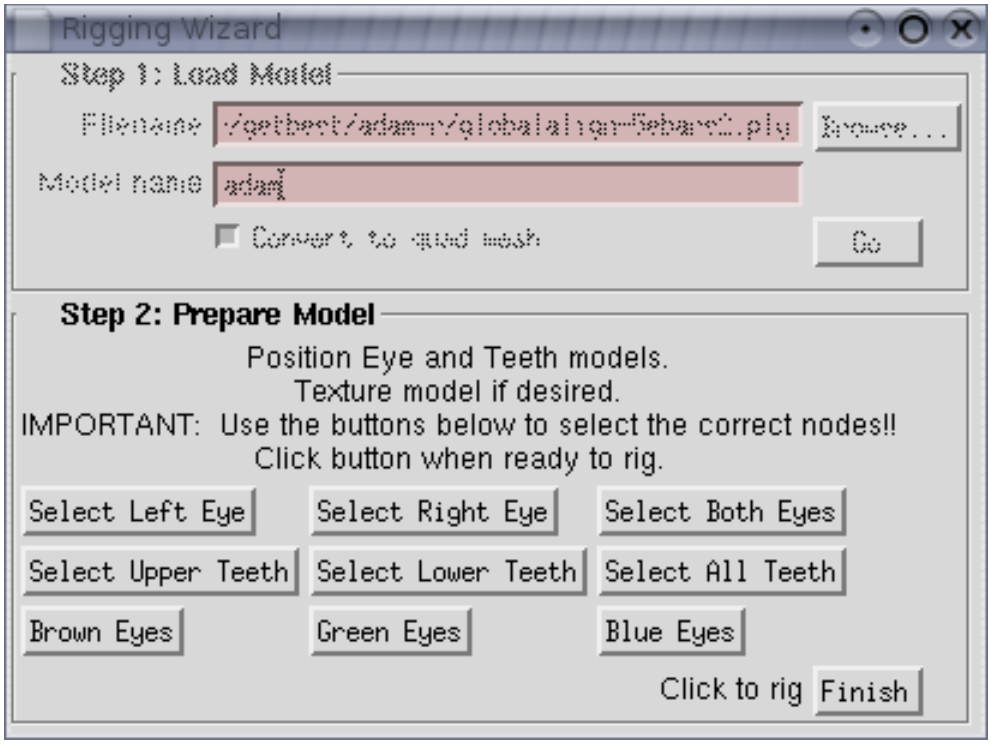



**Figure 6.4: The loading and rigging interface is shown with the second step enabled. Six buttons allow the user to easily select the eyes and teeth for accurate positioning. Another three buttons control the color of the iris in the eye shapes. The Finish button at the lower-right corner executes the automated rigging procedure.**

### 6.1.2 Rigging the Face

Although the system could continue and attach the reference rig to the fitted model at this point, we pause here for various reasons. As explained previously, the location of the helping geometry (the eyes and the teeth) is only an estimate of the optimal position. Therefore, we allow the user to position the surfaces interactively. To aid the user in this task, the interface to the second step of the rigging procedure provides controls to select these surfaces, as shown in Figure 6.4.

The user now has a chance to tweak the geometry of the model, fixing any problems he/she encounters. This is also a convenient time to shade and texture the head model—an operation which is normally done before deformations are

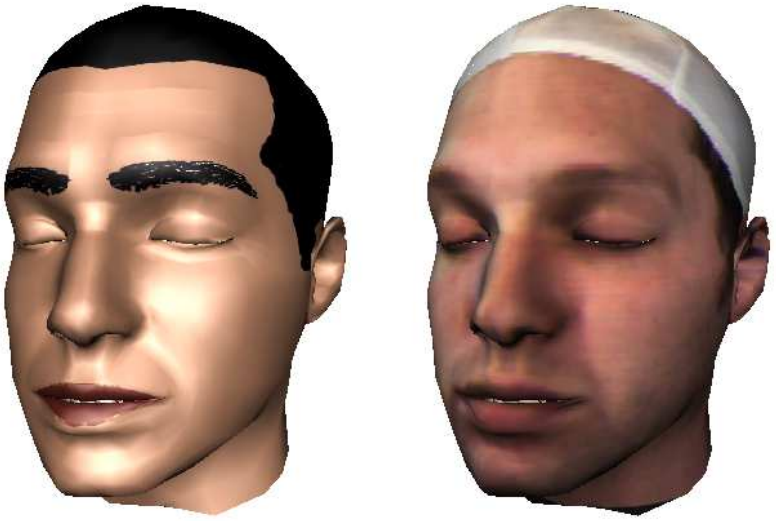


**Figure 6.5: Two example textures applied to Adam's fitted model. The texture on the left is the same texture developed for Murphy in Chapter 4, while the texture on the right is the original texture produced by the Cyberware scanner.**

applied. Figure 6.5 shows some examples of textured face models. To aid in the shading task, the user interface provides three buttons which change the color of the eyes. The options provided are brown, green, and blue.

Once the user is satisfied with the shape and shading of the model and the positions of the extra geometry, he/she can click the **Finish** button on the lower-right corner of the window shown in Figure 6.4 to execute the automated rigging process. The button runs a script that procedurally builds the reference rig on the fitted model. The resulting model, along with some control curves, is shown in Figure 6.6. The function of the control curves will be explained in Section 6.2.1.

## 6.2 Animation

To aid the animation process, we have developed a second user interface which provides a complete animation environment optimized for the reference rig. The

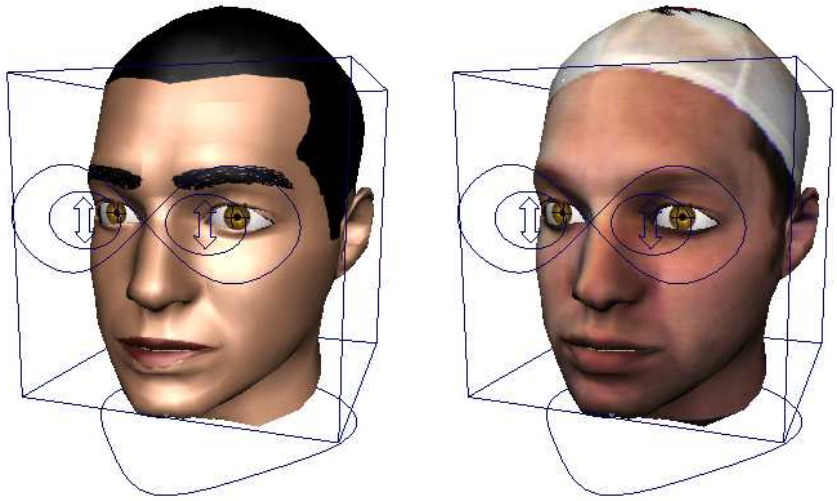


**Figure 6.6: The models shown in Figure 6.5 appear in their fully rigged state here, after completion of the rigging step. The control icons for placement, head rotation, eye position, and eyelid position are also shown.**

interface is flexible since it provides more than one interface for the rig controls and it allows the user to modify the limits of the control values.

The animation interface is composed of four panels, as shown in Figure 6.7. The bottom panel contains a time control which allows for interactive selection of the current frame and sound scrubbing, useful in lip synching. At the right is a perspective viewport showing the animatable model and the control icons. On the left there are two panels: the top panel, known as the *configuration* panel, contains controls which are useful during animation but do not necessarily affect the rig parameters. The lower panel is the character animation panel, which contains most of the animation controls related directly to rig parameters. The details of the interface are presented below.

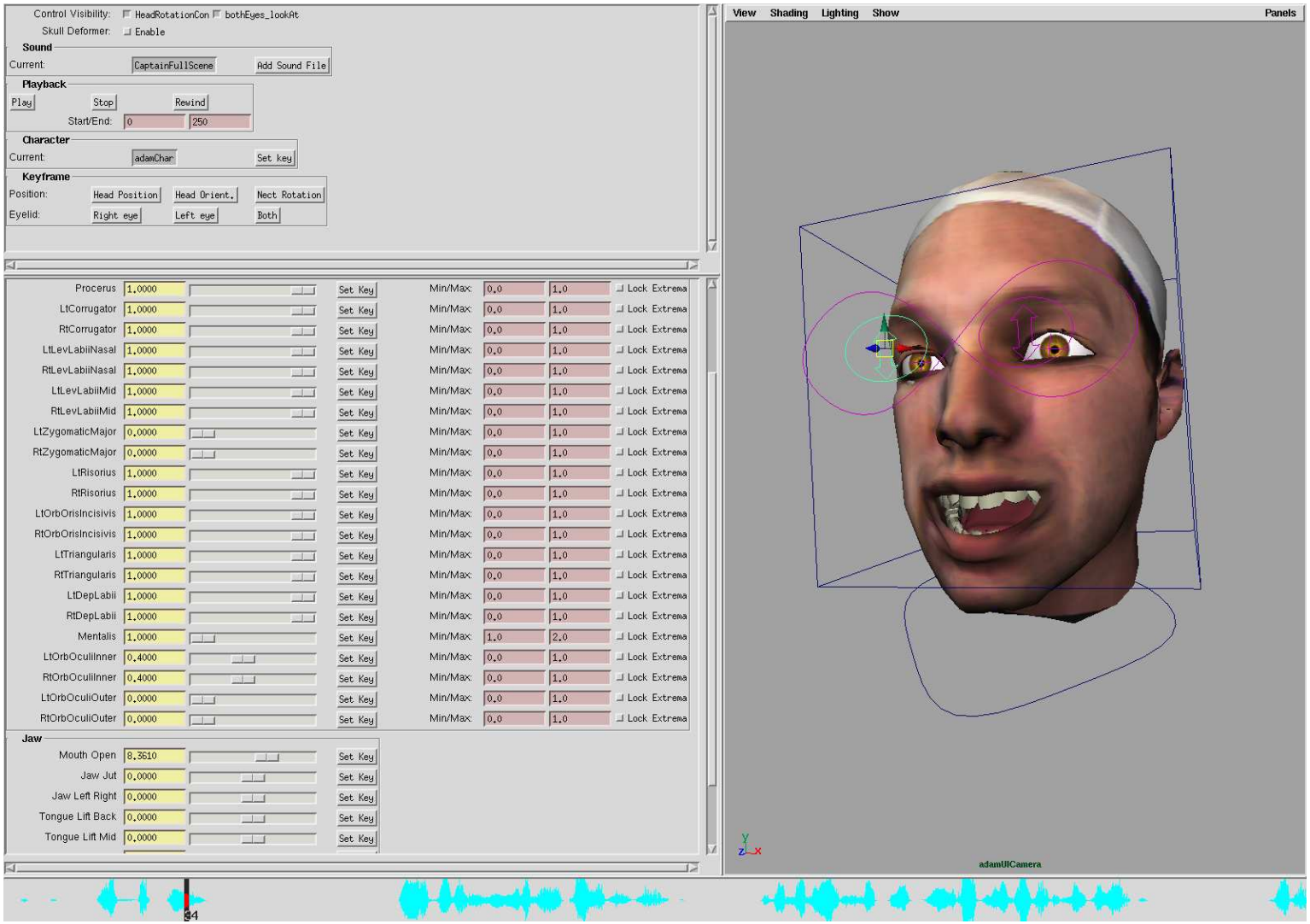



**Figure 6.7: The animation user interface.**

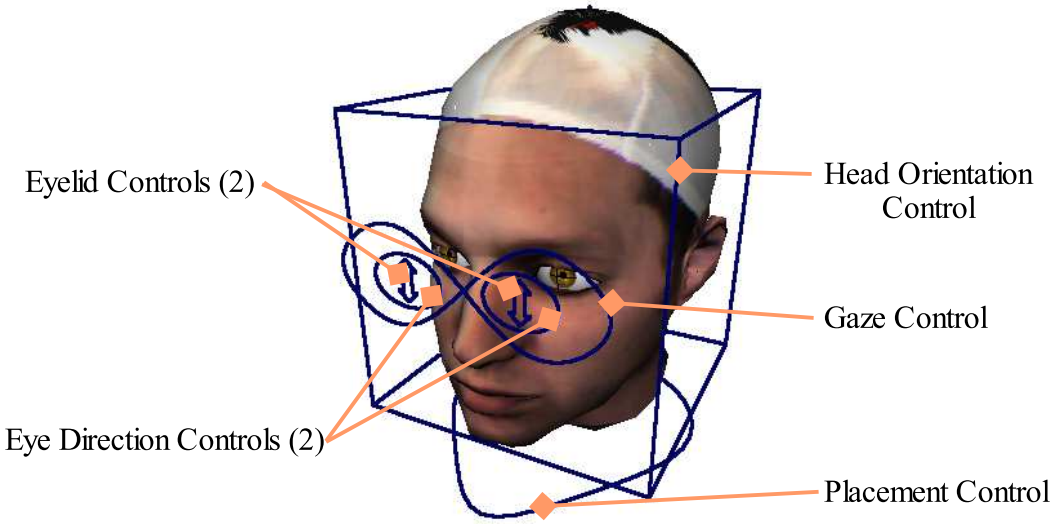



**Figure 6.8: Control icons used in the animation interface.**

### 6.2.1 Control Icons

The animation interface provides a total of seven control icons, shown in Figure 6.8. The icons control parameters which are more easily manipulated directly in the viewport window. The control icons are:

**Placement Control:** Located below the neck, this icon controls the overall position and orientation of the face model. It is shaped as a rounded arrowhead, showing the animator which direction points to the front. The animator can use this control to find the center position once the face has been turned to either side.

**Head Rotation Control:** Shaped as a cube, the icon controls the rotation of the head over the neck, simulating movement from the neck vertebrae. The animator can use this control to nod, shake the head, or let the head drop to the side.

**Gaze Control:** This icon was presented in Chapter 4. It is shaped in a figure '8'

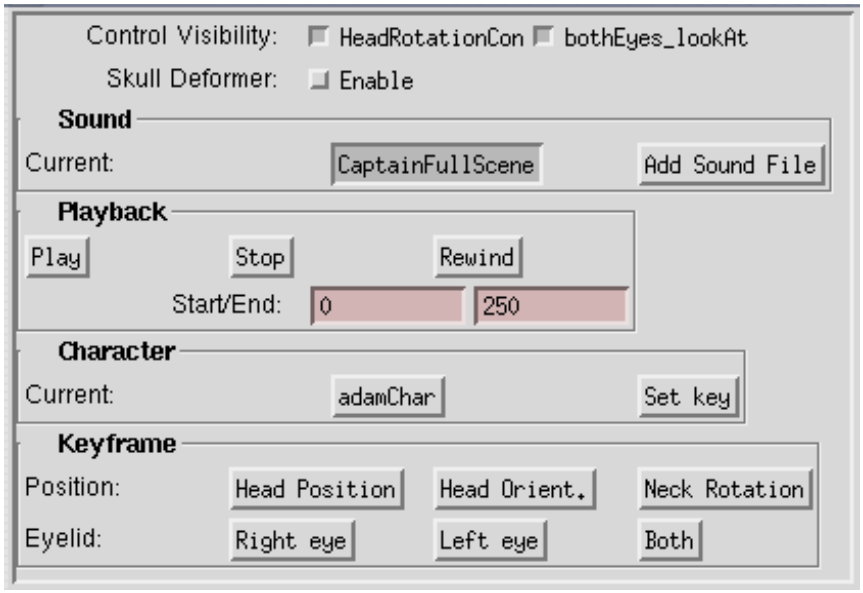



**Figure 6.9: The configuration panel of the animation interface.**

and controls the direction in which the eyes focus.

**Eye Direction Controls:** There are two of these icons, one for each eye. These were also presented in Chapter 4 and allow the animator to fine-tune the direction in which each eye is gazing individually.

**Eyelid Controls:** There are two of these icons, shaped as vertical double arrows. When moved up and down, they control the opening and closing of the eyelids.

### 6.2.2 Configuration Panel

The configuration panel contains controls which affect the animation environment, although it also contains some keyframing controls. The name "configuration panel" is therefore a misnomer, but we continue to use it for historical reasons.

A closeup of the configuration panel is shown in Figure 6.9. Two checkboxes

preceded by the label **Control Visibility** show and hide the control icons relating to head rotation and gaze. Just below, another checkbox enables or disables the skull deformer in the rig. Recall from Chapter 4 that the skull deformer is the most computationally expensive deformer. In practice, leaving this deformer active slows down the animation system to less than interactive rates. However, by turning it off, the animation system can respond at interactive rates and provides the animator with approximate feedback on the final look of the face. Therefore, the animator can use this button to disable the skull deformer for interactive work and turn it back on for final renderings.

Below the skull deformer control is the **Sound** group, which provides controls to select the current sound track that is displayed in the sound scrubber control at the bottom of the animation window. Another button allows the user to insert new sound bites into the scene.

Following the **Sound** group is the **Playback** group. This group provides three buttons to control playback of the animation. The number of frames displayed in the time control at the bottom of the window is changed using the text boxes found in this group.

The **Character** group follows. *Characters* are sets of attributes which can be assigned key frames as a group. They can also be grouped hierarchically, such that setting a key frame on a parent character will set key frames on all attributes of all its children. A hierarchical character set has been defined for the reference rig, and it is accessible through the controls in this group. Using a pop-up list control, the user can select a character and key it with the button immediately to the right. Table 6.1 shows the character breakdown of the parameters in the rig.

Finally, at the bottom of the panel, we find the **Keyframe** group, which pro-

**Table 6.1: Hierarchical character breakdown of the parameters in the facial rig. Parent characters are located to the right of each character or attribute. The number of attributes contained in the lower level characters are shown in parentheses. Note how the parent character's name is derived from the name given to the face model—in this case, Adam.**

| | | | |
|---|---|---|---|
| adamChar | PositioningChar | WorldSpaceChar | placement control translation and rotation (6) |
| | | NeckRotationChar | rotation of the neck (3) |
| | EyeMotionChar | EyelidChar | position of the eyelids (2) |
| | | EyeOrientationChar | position of three eye control targets (9) |
| | MouthMotionChar | JawChar | jaw rotations and jutting (3) |
| | | TongueChar | tongue motions (6) |
| | MusclesChar | EyeMusclesChar | eye muscles, except for orbicularis oculi (7) |
| | | MouthMusclesChar | mouth muscles (15) |
| | | CheekMusclesChar | orbicularis oculi muscles (4) |

vides a quick means to keyframe some of the parameters managed by the control icons of the rig.

### 6.2.3 Animation Panel

The animation panel is actually composed of three tabs which provide three different interfaces to the parameters of the rig. They are the **Low-level** tab, the **Descriptive** tab, and the **Expression** tab.

#### Low-level controls

The **Low-level** tab provides direct access to the rig parameters which are not managed by control icons. These include all of the muscle joint contractions and the movements of the jaw and tongue.

Figure 6.10 shows the complete tab, with enlargements shown in Figures 6.11, 6.12, and 6.13. The controls in the tab are separated into a **Muscles** group and a **Jaw** group. In the **Muscles** group there is an entry for each of the muscle contraction parameters defined in the rig. Proceeding left to right, each entry contains a label with the muscle name followed by a textbox which shows the current value of the muscle contraction. This box can also be used to type in the desired contraction value. The slider which follows allows the animator to change the contraction value interactively, and keys can be added with the **Set Key** button on the right (see Figure 6.11). Further to the right are two text boxes, shown in Figure 6.12, which control the extremum values of the slider for this muscle. The **Lock Extrema** checkbox locks these values so that they are not accidentally modified. The values for the extrema are saved between animation sessions, so the animator need only set them once.

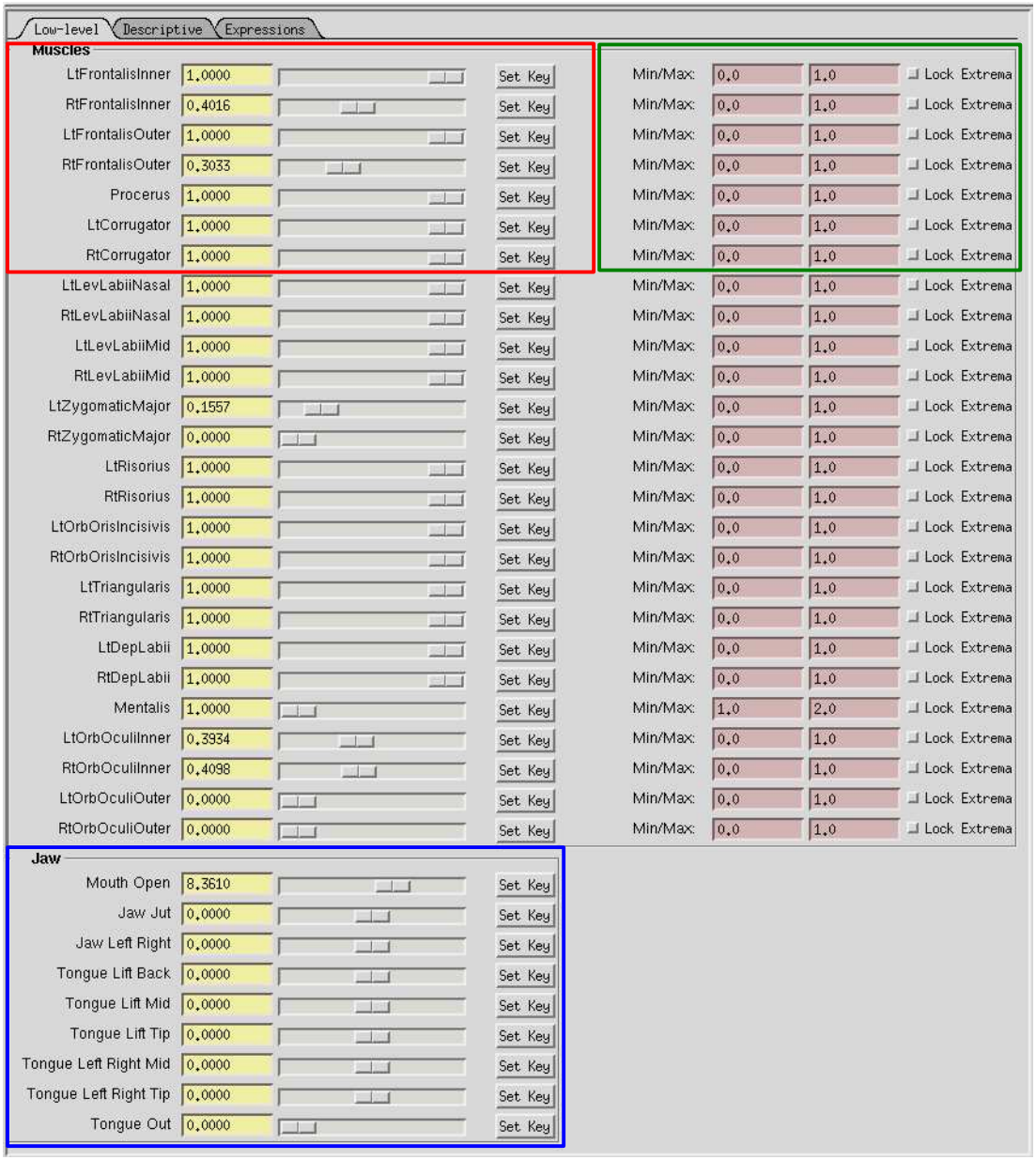



**Figure 6.10: The Low-level controls tab in the animation user interface. Magnifications of the red, green, and blue boxes are shown in Figures 6.11, 6.12, and 6.13, respectively.**

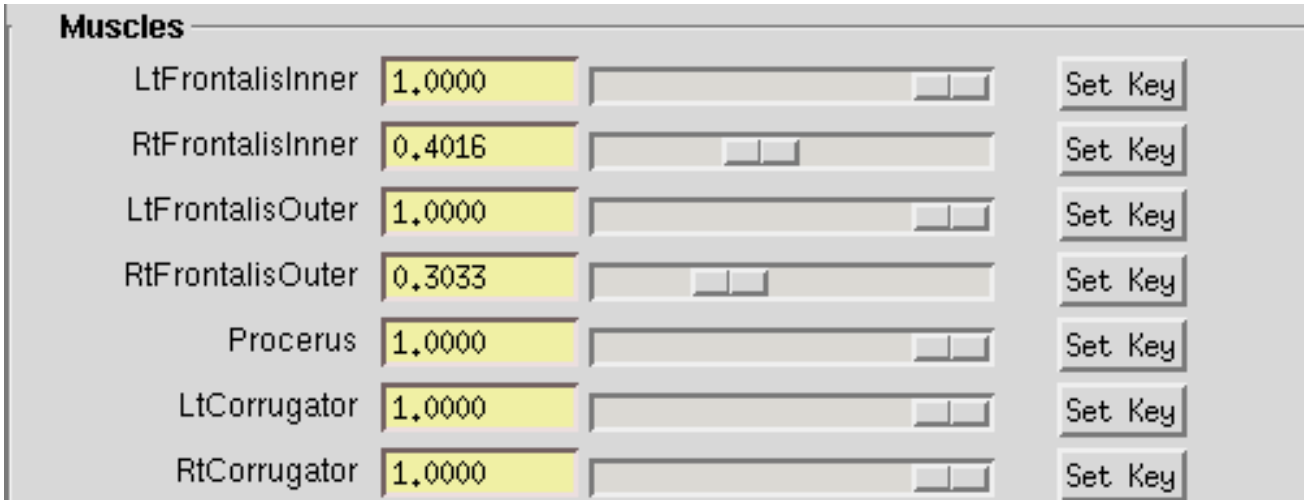



**Figure 6.11: An enlargement of the left side of the Muscles group in the Low-level tab. The area shown corresponds to the red box in Figure 6.10.**

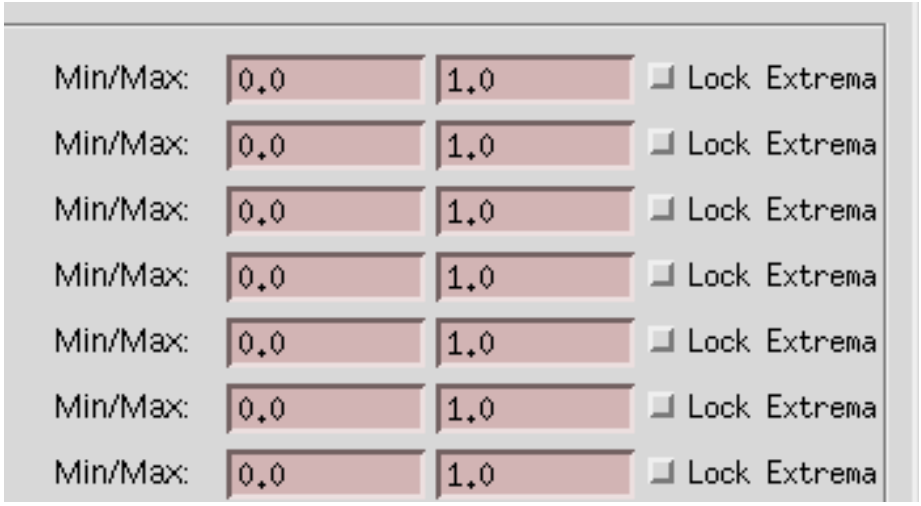



**Figure 6.12: An enlargement of the right side of the Muscles group in the low-level tab. The area shown corresponds to the green box in Figure 6.10.**

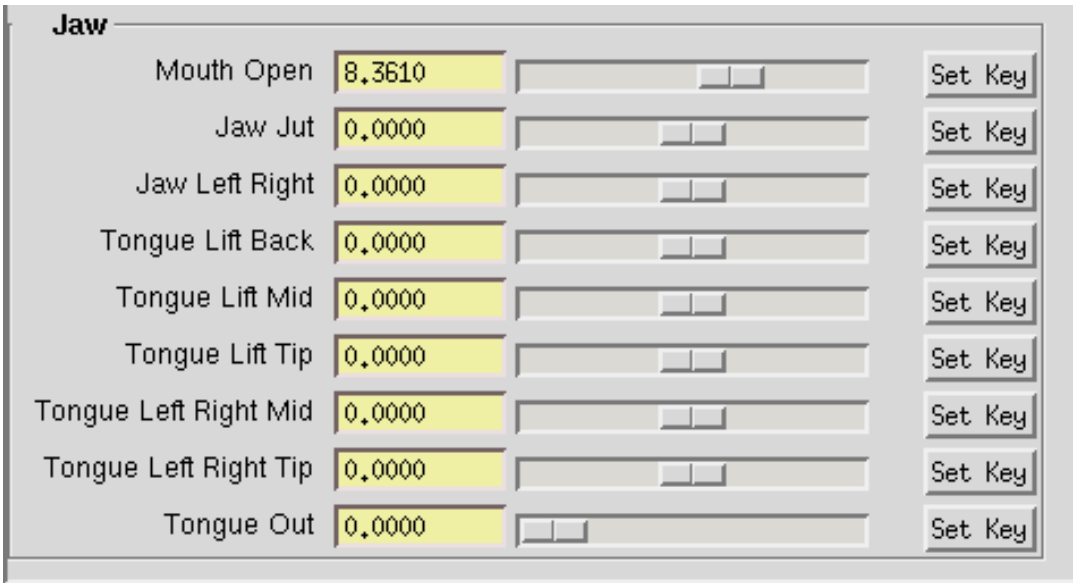



**Figure 6.13: Enlargement of the Jaw group in the Low-level tab. The area shown corresponds to the blue box in Figure 6.10.**

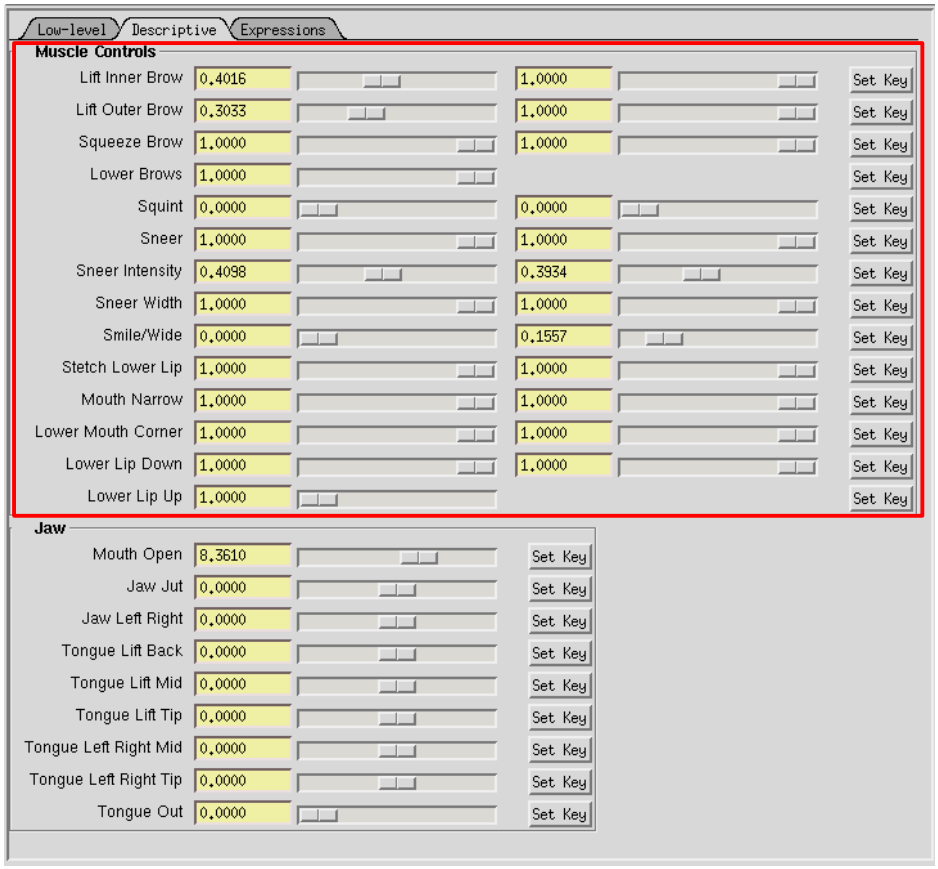



**Figure 6.14: The Descriptive control tab for the animation interface. A closeup of the Muscle Controls group, framed by the red box, appears in Figure 6.15.**

The **Jaw** group is similar in layout to the **Muscles** group, lacking only the extrema controls. This group contains controls for all of the parameters involving the movements of the jaw and the tongue, and it is pictured in Figure 6.13.

### Descriptive controls

This tab, like the **Low-level** tab, is also divided into two groups: a **Muscle Controls** group and a **Jaw** group, as shown in Figure 6.14. The **Jaw** group in this tab is an exact copy of the **Jaw** group from the **Low-level** tab.

An enlargement of the **Muscle Controls** group is shown in Figure 6.15. This

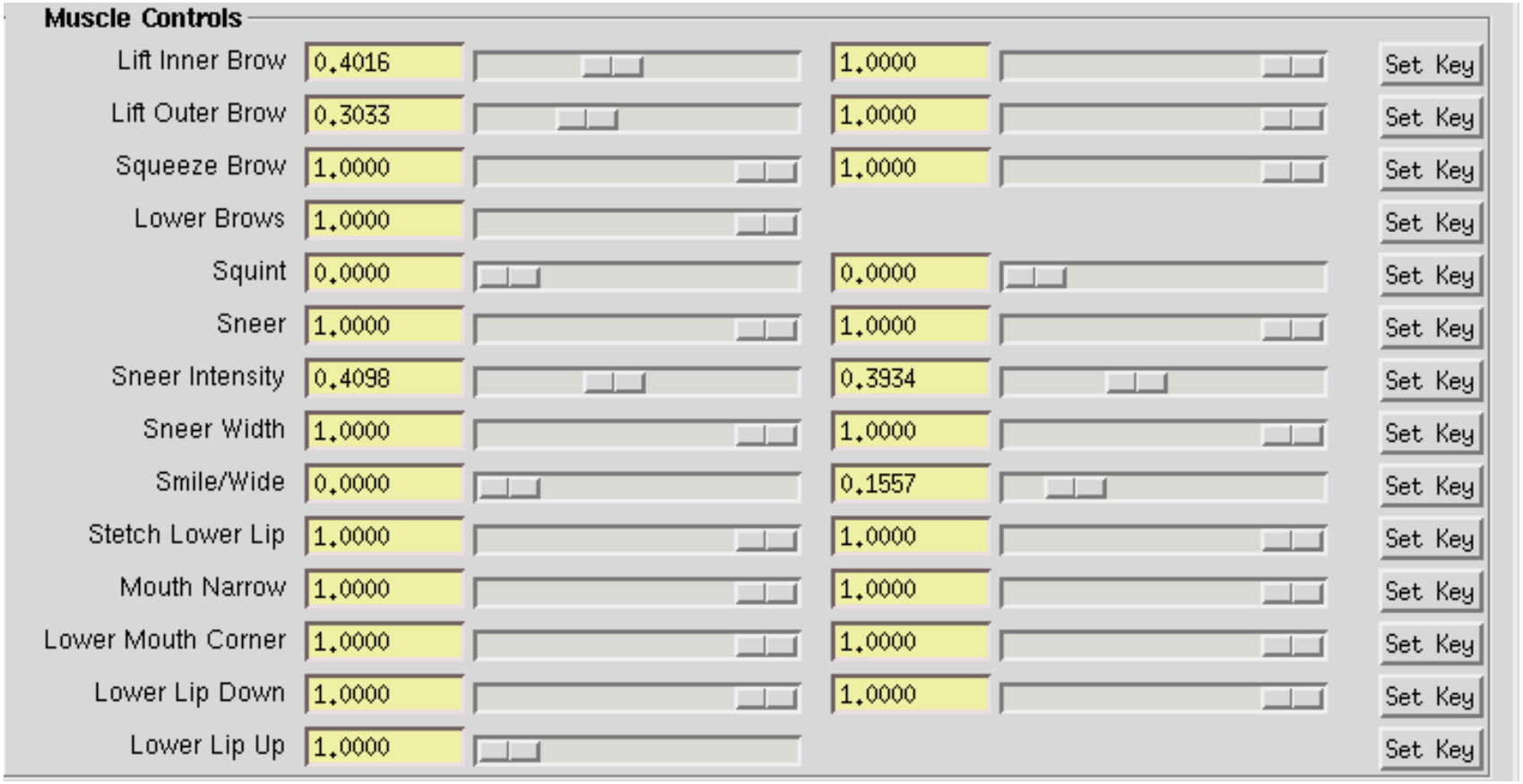



**Figure 6.15: An enlargement of the Muscle Controls group of the Descriptive tab, shown in Figure 6.14 inside the red box.**

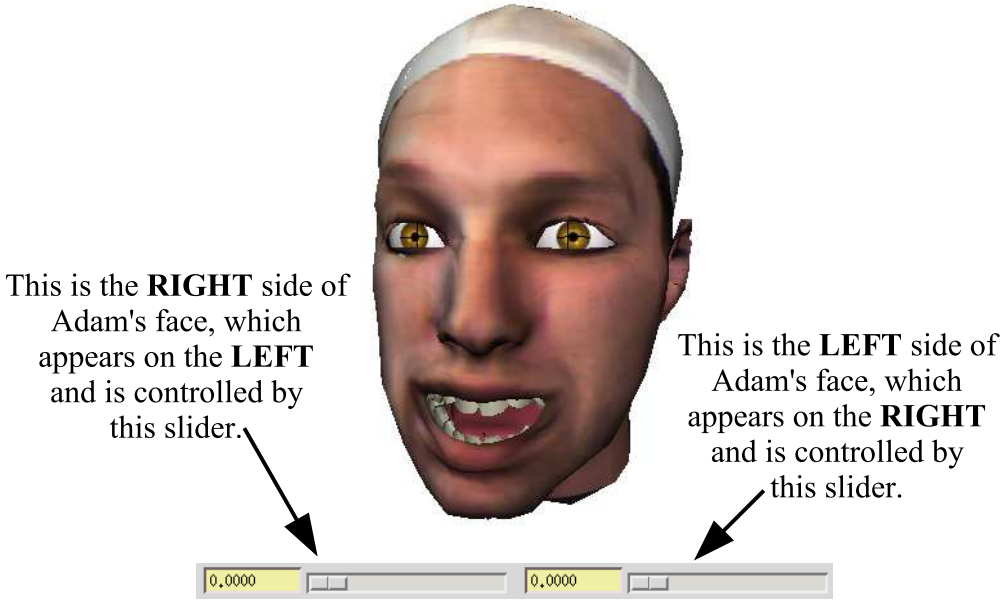



**Figure 6.16: Change of bearings in the control sliders.**

group provides controls to the same parameters as the **Muscles** group in the **Low-level** tab, but the labels for the controls are much more descriptive and the layout is more intuitive. For example, instead of using the label "Zygomatic Major," we use the label "Smile/Wide." Table 6.2 shows the mapping from low-level controls to descriptive controls.

Also, the sliders for both of these muscles are found on a single row of the interface. The slider on the left controls the right side of the face and vice versa. Although this sounds counterintuitive, consider that when you face a person, her right side appears on your left and vice versa, as shown in Figure 6.16. Thus, the slider on the left controls the region of the face which appears on the left.

The **Set Key** button which appears on the far right of each row sets key values for all of the parameter values in each row.

**Table 6.2: Mapping from Low-level control names to Descriptive names.**

| **Low-Level Name** | **Descriptive Name** |
|---|---|
| Frontalis Inner | Lift Inner Brow |
| Frontalis Outer | Lift Outer Brow |
| Procerus | Lower Brows |
| Corrugator | Squeeze Brow |
| Orbicularis Oris Inner | Sneer Intensity |
| Orbicularis Oris Outer | Squint |
| Levator Labii Nasal | Sneer |
| Levator Labii Middle | Sneer Width |
| Zygomatic Major | Smile/Wide |
| Risorius | Stretch Lower Lip |
| Orbicularis Oris Incisivis | Mouth Narrow |
| Triangularis | Lower Mouth Corner |
| Depressor Labii | Lower Lip Down |
| Mentalis | Lower Lip Up |


**Figure 6.17: The Expressions tab in the animation interface. For each expression, the icon and text provide a button which sets the parameters of the rig to the appropriate values for the expression.**

### Expression library

The **Expressions** tab, shown in Figure 6.17, provides a library of facial expression which the animator can quickly draw upon. We define expressions simply as a collection of rig parameter values. Fifteen expressions are available, described in the interface by a name and an icon showing Murphy emoting the particular expression. The icons are actually buttons which, when pressed, run a script setting the rig parameters to the corresponding expression values. The animator, therefore, can quickly select an expression to pose the facial model. Using the controls described in the previous tabs, the animator can change this initial pose to suit his needs.

## 6.3 Results

As stated in the beginning of this chapter, there is no guarantee that the reference rig developed for Murphy will produce the appropriate deformations on other meshes. However, we have found that the muscle based rig generally produces the correct expression for appropriate configurations of parameter values in the rig.

To illustrate our conclusion, we refer to the face models shown in Figure 6.18. Murphy is located at the top left in this set for convenient access to the reference results. Note that we have included Adam's scan twice: once with the texture generated from the scanner and once using Murphy's reference texture. This will allow us to examine the effects of texture on the perceived expression. Finally, note that Chris' scan (located at bottom right) did not require the use of a swim cap given Chris's short hair.

Figures 6.19 through 6.24 show the head models from Figure 6.18 portraying


**Figure 6.18: The head models used to illustrate the correctness of the generic rig deformations applied to fitted head scans. On the top left corner we show Murphy, the reference model. The other meshes on the top row correspond to Adam, first textured with Murphy's reference texture and then textured with the scan texture. On the bottom row, from left to right, we find head models for Ellie, Dhruva, and Chris.**

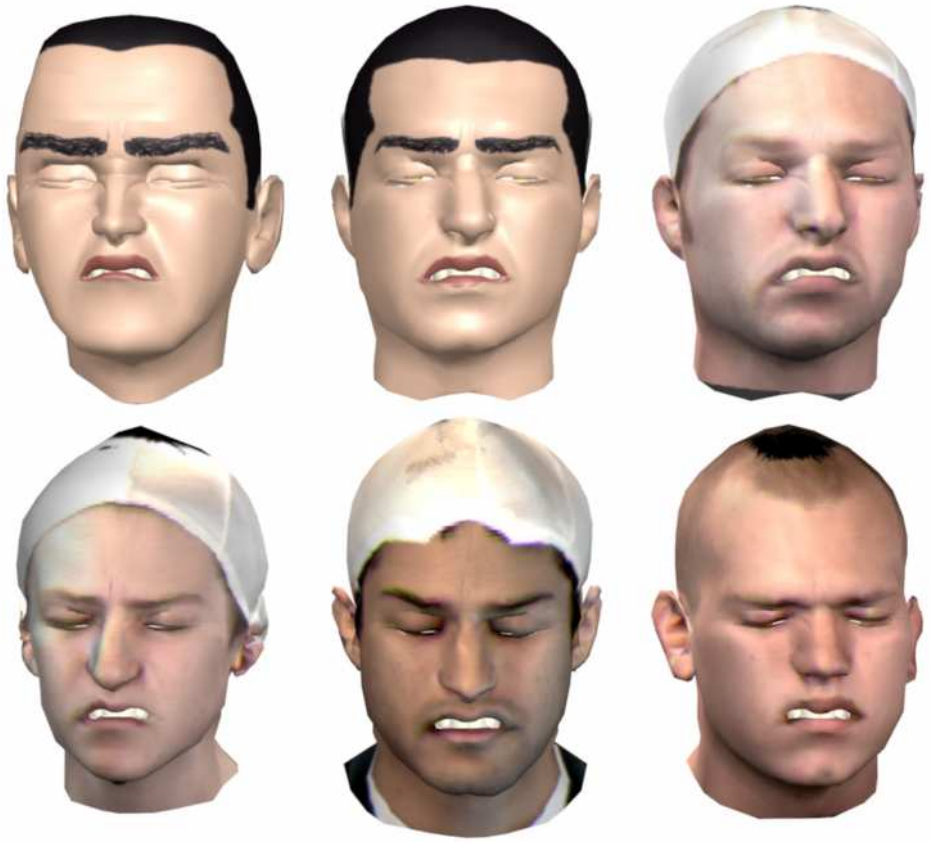

**Figure 6.19: The sample head models showing extreme sadness (weeping).**


**Figure 6.20: The sample head models portraying extreme anger (rage).**

Left to right:
Murphy
Adam 1
Adam 2

Left to right:
Ellie
Dhruva
Chris

**Figure 6.21: The sample head models portraying joy.**



Left to right:
Murphy
Adam 1
Adam 2

Left to right:
Ellie
Dhruva
Chris

**Figure 6.22: The sample head models showing disgust.**


**Figure 6.23: The sample head models showing surprise.**

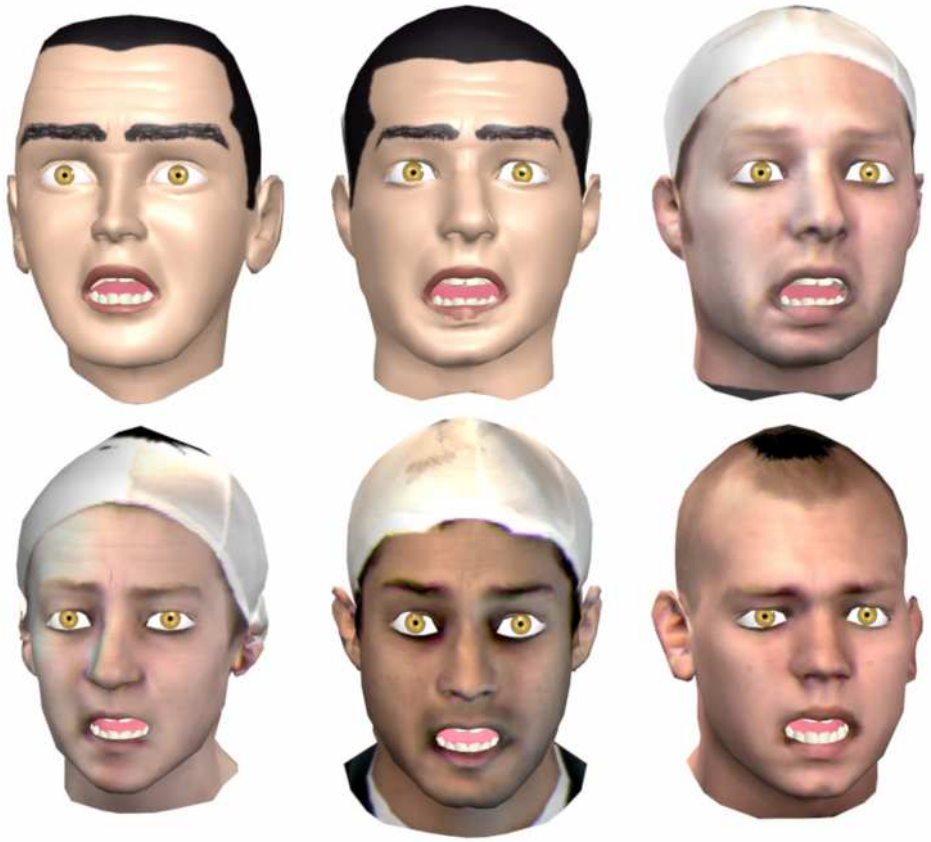



**Figure 6.24: The sample head models portraying fear.**

each of the six universal expressions. For the most part, the expression portrayed is unmistakable: parameters which produce an angry face in Murphy also produce angry faces in the fitted scans, and likewise for the other expressions. There are, however, some details which could be improved.

Of the six universal expressions, the least convincing is weeping. There are a few causes for this. First, the upper teeth are showing in all of the models. As explained in Chapter 2, this is very uncommon in the weeping expression. Second, the corners of the mouth are not pulled down far enough to reflect the strong action of the risorius muscle. The action of the mentalis muscle is also masked somewhat since we have not implemented a simulation of pockmarks in the chin. Note that these problems can all be corrected by the animator by extending the limits of the risorius muscle contraction or by implementing his own deformation for the pockmarks.

Aside from the particular concerns for the weeping expression, there are more general concerns as well. For example, there are a number of issues involving the eyes. In Figures 6.23 and 6.24, the expressions of surprise and fear are not convincing in Ellie's and Chris's scans (bottom, corners) because their eyes do not appear to open wide as they should. Figure 6.25 compares the default and wide eye configurations for Dhruva's and Ellie's eyes. Although Dhruva's eyes open wide, Ellie's eyes do not open past the default position. In fact, the eyelid has moved behind the skin of the upper eye (note the evidence of self-intersections in the eyelid model). The relative position of the skin over the iris, therefore, does not appear to rise, thus eliminating the most important visual queue for determining how wide the eye is. There are two explanations for these problems. First, the fitting procedure can pull the vertices of the upper eye too close to the eye opening.

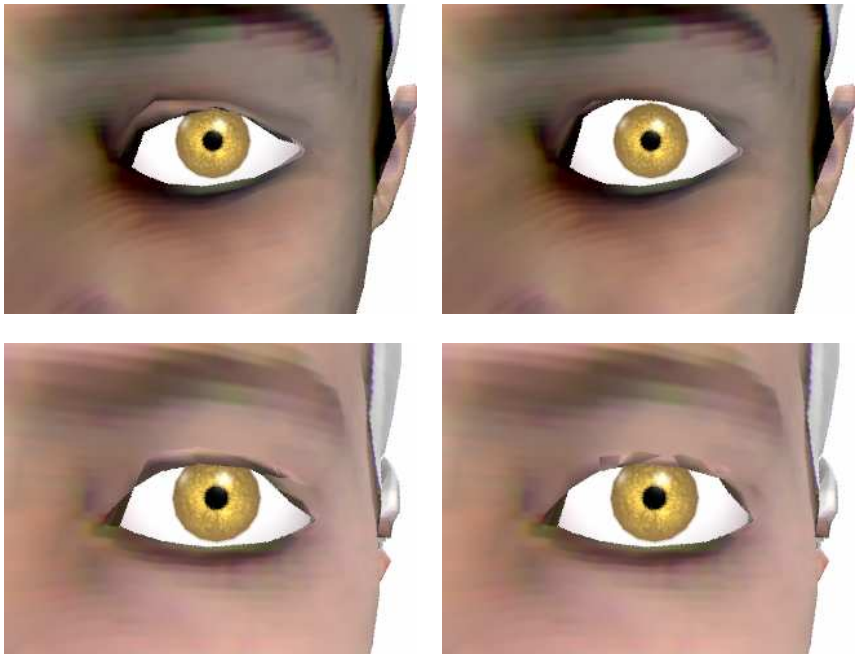



**Figure 6.25: Comparison between the default and open eye configurations for Dhruva's and Ellie's fitted scans (top and bottom rows, respectively). Note how Dhruva's eyes are able to open wide. Ellie's eyelids, however, are hidden behind the skin of the upper eye and therefore the eyelid line does not rise over the iris.**

This sets an upper limit on the line between the skin and the iris which cannot be changed by opening and closing the eye. Second, recall from Chapter 4 that the eyelid motion is simulated as a rotational transform about the center of the eye. Real eyelids do not move in this fashion, but are lifted instead by the levator palpebrae muscle. It is possible that using a more structurally-based deformation about the eye might overcome the problems due to the fitting procedure.

Other issues which affect the correct portrayal of expressions involve shading. In Figures 6.18 through 6.24, the models textured with Murphy's texture use a Blinn/Phong material while the models with scan textures use a Lambertian material, which better reflects the properties of skin. The Lambertian material,

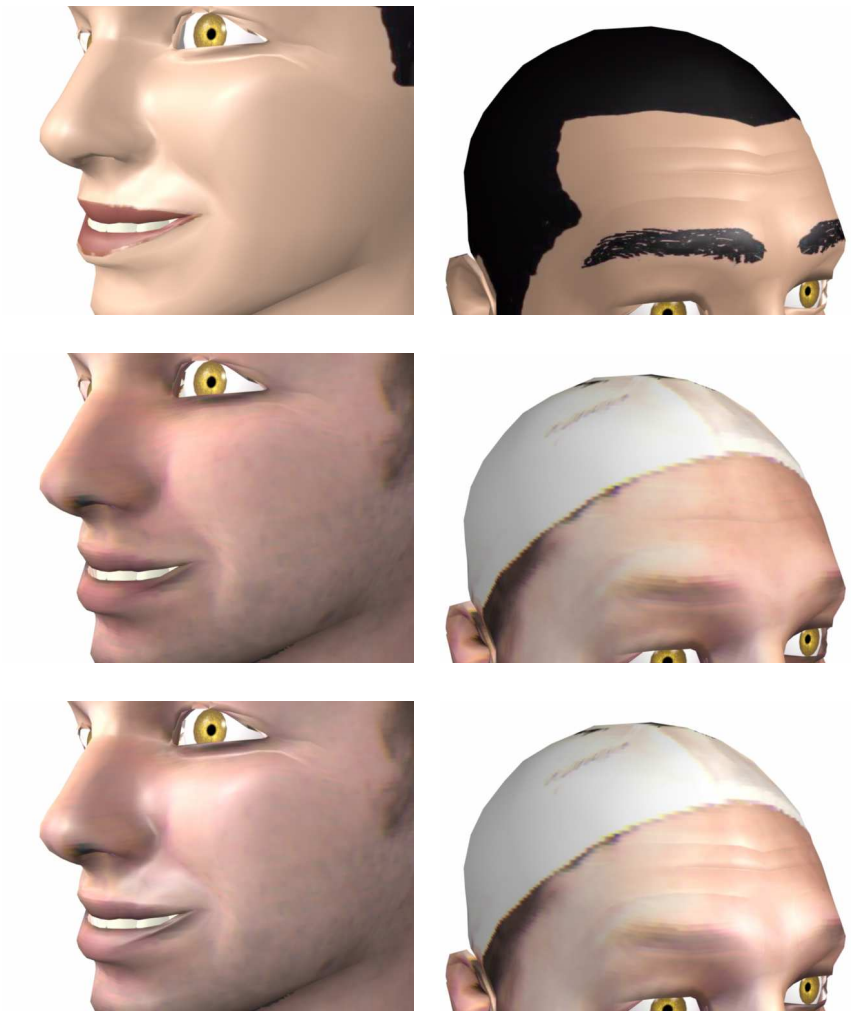


**Figure 6.26: The crease masking effects of shading are evident from these images. The top row shows Adam's scan textured with Murphy's original texture and shaded with a Blinn/Phong material, which makes the creases in the eyes and forehead evident. At center, the texture has been switched to the scan texture, and the model is shaded with a Lambertian material. This material masks the creases, which reappear once the Blinn/Phong material is applied (bottom row).**

unfortunately, has a masking effect on the creases around the eyes and the forehead; that is, the material hides the creases from view, as seen in Figure 6.26. The absence of the creases reduces the intensity of the emoted expression, as can be seen in Figures 6.23 and 6.24. However, if the Blinn/Phong material is applied to these models (bottom of Figure 6.26), the creases reappear, although the surface acquires a plastic sheen not typical of human skin. Since neither of these shading models accurately reflects the properties of skin, a more complex shader is highly recommended. However, in terms of facial expression, it is important that the shading algorithm highlight the creases in the model surface.

Finally, note that Dhruva's swim cap grows two points whenever the emoted expression requires lowering the eyebrows, as in Figures 6.19, 6.20, and 6.22. Since the swim cap was not placed exactly at the hair line during Dhruva's scan, some of the white color is spilling into the areas of the forehead that actually move. Thus, careful placement of the swim cap is important during the initial scanning procedure.

Aside from these defects, the faces appear to move and emote realistically, even when the emoted expression is asymmetric, not universal, or a combination of expressions, as shown in Figures 6.27–6.30. Asymmetries in the face are maintained during the deformations as well—see, for example, Adam's chin.

## 6.4 Summary

We have combined the work developed in the previous chapters into a comprehensive animation system. This system allows the artist to quickly load and rig a model of a human head. Additionally, a flexible animation environment is available to jumpstart the animation process.


**Figure 6.27: The sample head models portraying a joyous surprise, a combined expression.**

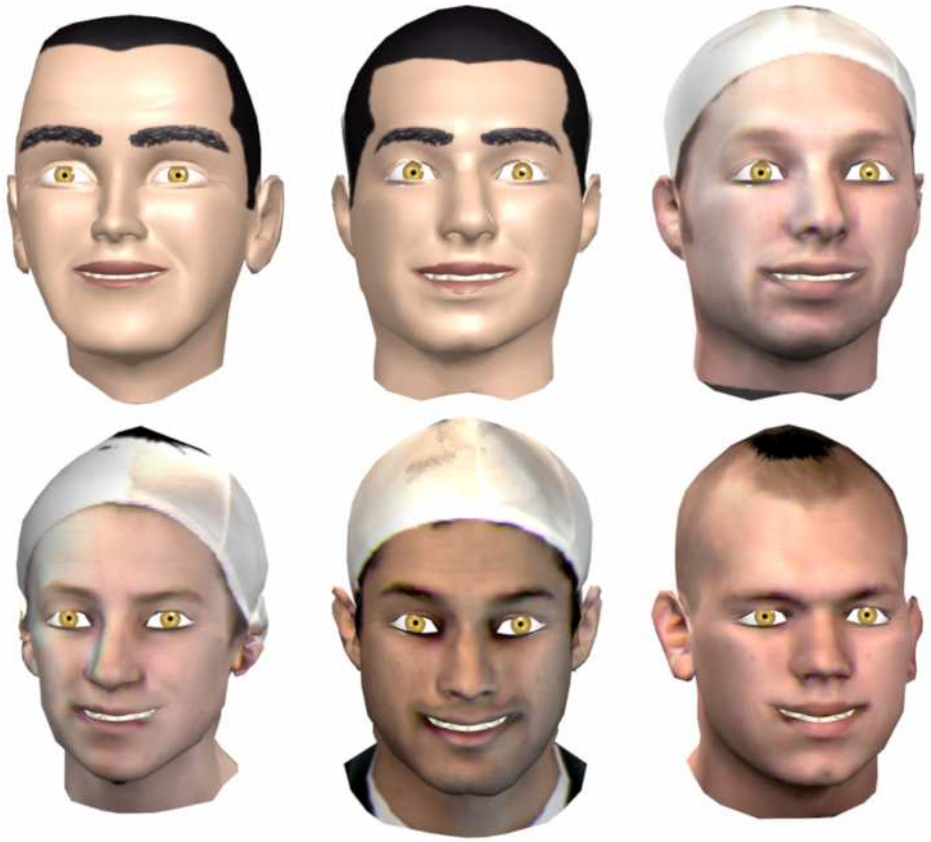



**Figure 6.28: The sample head models smiling. Compare to the fake smile shown in Figure 6.29.**


**Figure 6.29: The sample head models portraying a fake smile. Compare to the genuine smile shown in Figure 6.28.**

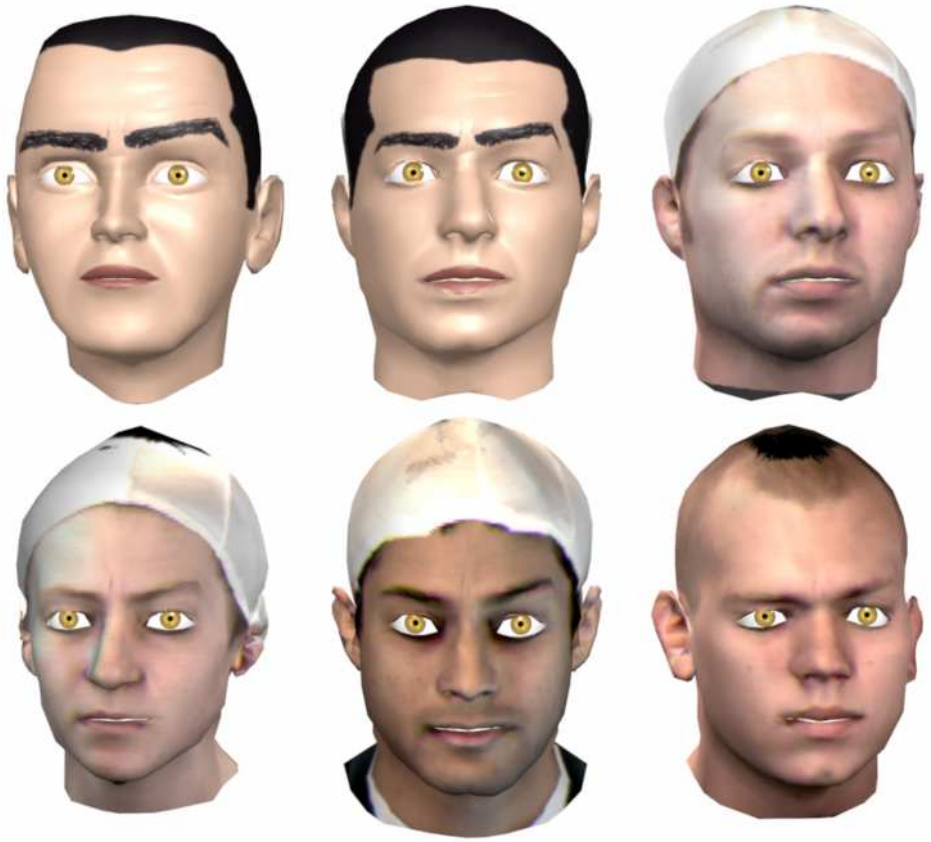



**Figure 6.30: The sample head models showing a quizzical expression, which is asymmetric on the face.**

Although the reference rig is constructed on new geometry, the deformations applied produce realistic expressions. However, if the animator is not satisfied with a particular expression, he or she is free to tweak both the face model and the rig deformations to produce better expressions. It should be pointed out that building and rigging a model from scratch, even for a professional animator, requires an estimated 100 man-hours of work[1], while a full run of our system (scanning, fitting, and rigging) can be accomplished in under one hour. Thus, the time savings are enormous, even if final tweaking is necessary. Therefore, this approach is efficient and effective, and ultimately systems of this type will reduce the time required to generate realistic facial animations.

[1]This figure is extrapolated from values presented in [Osi03].

# Chapter 7

# Conclusions and Future Work

We have presented a system for the automated rigging of human face models. Our system takes advantage of previous work which deforms the surface of a reference face model to match digitized face models generated by a Cyberware facial scanner. To automate the rigging procedure, we have parameterized the construction of the reference rig on the vertex topology of the reference model. The rig can then be procedurally reconstructed onto deformed copies of the reference model. The animation parameters in our rig are structural in nature, based on the expressive muscles of the face. A novel skin/muscle model, based on wire deformers and point constraints is used to implement this parameterization. The model supports any number of muscle contractions about a particular skin patch (for example, the corner of the mouth) while allowing the artist to control the shape of the deformations. Finally, we presented a user interface for quickly loading and rigging new models and a flexible environment for animation.

The models and resulting facial deformations produced by our system are realistic and expressive, although we have not provided any objective measure of the "realism" of our generated models. However, our main goal has been to reduce

the amount of time required to rig new face models. Thus, our system provides a good starting point for the animator to tweak the shapes or deformations which are not to his or her liking. Considering the amount of time the same artist would need to create and rig a head model from scratch, the time savings provided by our system are significant. In the future, systems of this type will greatly reduce the amount of time required for generating realistic facial animations of different characters.

Our system offers other advantages over popular modeling and rigging methods such as parameterized models and morphing systems. For instance, the number of characters a parameterized face model can represent is limited by the range of values of its conformal parameters. In our system, the number of characters is limited only by the number of configurations achievable by displacing the vertices of the reference model, providing a much wider range.

Also, it is well known that morphing systems can be used to animate any character's face. However, the only achievable expressions in these systems are combinations of target shapes which have been previously modeled or captured. This extra modeling work is not necessary in our system—only the default, neutral expression is required. This can be modeled or captured using the reference model surface as a template, and the reference rig will be capable of generating a wide range of expressions. Our approach, therefore, saves time and storage space by reducing the number of surface models needed to create an animatable face. Additionally, if a morphing system is required due to design constraints, our system can be used to generate the target shapes or shape templates.

Although effective at generating believable facial animation, the current implementation of our system can be improved. As mentioned in Chapter 5, the scan

fitting procedure implemented in this work provides no guarantee that the vertices in the reference and fitted models will be located in corresponding regions of the face. Since our system implicitly assumes that this correspondence exists, we recommend that future implementations use either feature-based or constraint-based fitting algorithms. Another option is to limit the fitted region to the facial mask only, since the rest of the head is not involved in facial motion.

Enhancements should be made to the reference rig as well. A significant deficiency is the lack of sphincter muscles, which should be used in the eye and mouth regions. Such muscles could be implemented by a scaling transformation about the center of the muscle, and control over the final shape of this deformation should be given to the animator. Complications can also arise due to the interaction of skin with the underlying facial structure. This issue must be addressed to avoid self-intersections and intersections with the teeth and the eyes. Although sculptor and push-to-surface deformers can help, their computational expense might prohibit interactive responses. How to prevent these intersections procedurally is an area of future research.

Our system currently makes no attempt at modeling or rendering hair or skin accurately, relying on basic texturing routines. Future developments should incorporate more advance procedures for added realism.

Although we have formulated our rig for animation by a trained artist, current industry trends have focused on driving animation using motion-captured data. Additionally, MPEG-4 Facial Animation FAP streams also provide prerecorded motion data for driving face models in real time. Whether topologically parameterized rigs can be driven by these technologies should be explored.

Finally, although we have focused entirely on the human face in this work,

we believe the concepts presented are flexible enough to be used on non-human characters.

# Appendix A

# Expressing $E_{global}$ as a Sparse Linear System

We derive here the mathematical formulation which expresses $E_{global}$ as a sparse linear system solvable using the conjugate gradient method. Let the set of vertices $V$ of the reference mesh be represented by a column vector $\mathbf{v}$. Close inspection of Equations 5.4–5.7, 5.9, and 5.10, using the linearized version of $E_{dist}$, shows us that each term in $E_{global}$ is a linear combination of the vertices in $V$, which we can express as

$$E_{global}(V) \rightarrow E_{global}(\mathbf{v}) = \sum_i \|\mathbf{c}_i \cdot \mathbf{v} + b_i\|^2. \tag{A.1}$$

Here, $\mathbf{c}_i$ is the vector of coefficients of the linear combination and $k_i$ is the constant term. Expressing the dot product as $\mathbf{c}_i{}^{\mathbf{T}}\mathbf{v}$ and expanding the square, we obtain

$$E_{global}(\mathbf{v}) = \sum_i (\mathbf{c}_i{}^{\mathbf{T}}\mathbf{v})^2 + 2b_i\mathbf{c}_i{}^{\mathbf{T}}\mathbf{v} + b_i^2. \tag{A.2}$$

Considering each term independently, we immediately observe that $\sum_i b_i^2$ is simply a constant, which we denote as $K$. Now, since $\mathbf{c}_i$ is a column vector, we denote its elements ans $c_{ji}$, where $j$ represents the row index. This allows us to

express the first term of Equation A.2 as

$$\begin{aligned}
\sum_i (\mathbf{c}_i{}^{\mathbf{T}}\mathbf{v}) &= \sum_i (\sum_j c_{ji}{}^{\mathbf{T}} v_j)^2 \\
&= \sum_i (\sum_j c_{ji}{}^{\mathbf{T}} v_j)(\sum_k c_{ki}{}^{\mathbf{T}} v_k) \\
&= \sum_i \sum_j \sum_k c_{ji}{}^{\mathbf{T}} v_j c_{ki}{}^{\mathbf{T}} v_k \\
&= \sum_i \sum_j \sum_k v_j c_{ji}{}^{\mathbf{T}} c_{ik} v_k \\
&= \mathbf{v}^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}\mathbf{C}\mathbf{v}, \qquad \text{(A.3)}
\end{aligned}$$

where we have expressed the sparse matrix $[c_{ij}]$ as $\mathbf{C}$. Note that the product of the matrices $\mathbf{C}^{\mathbf{T}}\mathbf{C}$ is a square, symmetric, positive definite matrix.

Finally, using the same constructions as above, we consider the term

$$\begin{aligned}
\sum_i 2 b_i \mathbf{c}_i{}^{\mathbf{T}}\mathbf{v} &= 2 \sum_i \sum_j b_i c_{ji}{}^{\mathbf{T}} v_j \\
&= 2 \sum_i \sum_j b_i c_{ij} v j \\
&= 2\mathbf{b}^{\mathbf{T}}\mathbf{C}\mathbf{v}. \qquad \text{(A.4)}
\end{aligned}$$

Combining Equations A.3 and A.4 together with $K$, we obtain

$$E_{global}(\mathbf{v}) = \mathbf{v}^{\mathbf{T}}\mathbf{C}^{\mathbf{T}}\mathbf{C}\mathbf{v} + 2\mathbf{b}^{\mathbf{T}}\mathbf{C}\mathbf{v} + K. \qquad \text{(A.5)}$$

Equation A.5 is a standard quadratic form which can be minimized by solving the linear system

$$\mathbf{C}^{\mathbf{T}}\mathbf{C}\mathbf{v} - \mathbf{b}^{\mathbf{T}}\mathbf{C} = 0. \qquad \text{(A.6)}$$

Using $\mathbf{A} = \mathbf{C}^{\mathbf{T}}\mathbf{C}$ and $\mathbf{r} = \mathbf{b}^{\mathbf{T}}\mathbf{C}$, Equation A.6 reduces to Equation 5.11.

# References


[Ali] Maya 5.0. Alias Corp. Inc. Toronto, Canada.

[BV99] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[Cyb] Head and Face Color 3D Scanner model 3030RGB/PS. Cyberware Laboratory Inc. Monterey, CA, USA.

[Dar65] Charles Darwin. *The Expression of the Emotions in Man and Animals*. The University of Chicago Press, 1965. Edited by Konrad Lorenz.

[dB90] G.-B. Duchenne de Boulogne. *The Mechanism Of Human Facial Expression*. Cambridge University Press, 1990. Edited and translated by R. Andrew Cuthbertson.

[Dis] 3ds max 6.0. Discreet, a division of Autodesk, Inc. Montreal, Canada.

[DMM03] Jürgen Hesser Dennis Maier and Reinhard Männer. Fast and accurate closest point search on triangulated surfaces and its application to head motion estimation. In *3rd WSEAS International Conference on SIGNAL, SPEECH and IMAGE PROCESSING (WSEAS ICOSSIP 2003)*, Rethymna, Crete Island, Greece, October 2003.

[DP63] Peter B. Denes and Elliot N. Pinson. *The Speech Chain: The physics and biology of spoken language*. Bell Telephone Laboratories, Incorporated, 1963. Printed by Waverly Press, Inc.

[EF76] Paul Ekman and Wallace V. Friesen. Measuring facial movement. *Journal of Environmental Psychology and Nonverbal Behavior*, 1:56–75, 1976.

[EF78] Paul Ekman and Wallace V. Friesen. *Facial Action Coding System (FACS): A technique for the measurement of facial action.* Consulting Psychologists Press, Palo Alto, CA, 1978.

[EF82] Paul Ekman and Wallace V. Friesen. Measuring facial movement with the facial action coding system. In Paul Ekman, editor, *Emotion in the Human Face*, chapter 9, pages 178–211. Cambridge University Press, second edition, 1982.

[EFE82] Paul Ekman, Wallace V. Friesen, and Phoebe Ellsworth. What are the similarities and differences in facial behavior across cultures? In Paul Ekman, editor, *Emotion in the Human Face*, chapter 7, pages 128–143. Cambridge University Press, second edition, 1982.

[EO82] Paul Ekman and Harriet Oster. Review of research, 1970–1980. In Paul Ekman, editor, *Emotion in the Human Face*, chapter 8, pages 147–177. Cambridge University Press, second edition, 1982.

[ERC] Exhibitor Relations Co., Inc. Encino, CA, USA.

[Fai90] Gary Faigin. *The Artist's Complete Guide to Facial Expression.* Watson-Guptill Publications, 1990.

[GGW+98] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Fredric Pighin. Making faces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 55–66. ACM Press, 1998.

[Gou71] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–629, June 1971.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[Int] Intel Corporation. Colorado Springs, CO, USA.

[JKHS02] Won-Ki Jeong, Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Automatic generation of subdivision surface head models from point cloud data. In *Proc. Graphics Interface*, pages 181–188, May 2002.

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

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

[KHYS02] Kolja Kähler, Jörg 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.

[KMMTT92] Prem Kalra, Angelo Mangili, Nadia Magnenat-Thalmann, and Daniel Thalmann. Simulation of facial muscle actions based on rational free form deformations. In A. Kilgour and L. Kjelldahl, editors, *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*, volume 11 of *3*, pages 59–69, 1992.

[Kun99] Andrew Kunz. Face vectors: An abstraction for data-driven 3-D facial animation. Master's thesis, Cornell University, 1999.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000.

[LMH00] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94. ACM Press/Addison-Wesley Publishing Co., 2000.

[Loo00] Charles Loop. Managing adjacency in triangular meshes, 2000. Microsoft Research Technical Report, MSR-TR-2000-24.

[LTW93] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Constructing physics-based facial models of individuals. In *Proceedings of Graphics Interface '93*, pages 1–8, Toronto, Ontario, Canada, 1993.

[LTW95] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62. ACM Press, 1995.

[Mar04] Chris Maraffi. *Maya Character Creation, Modeling and Animation Controls*. New Riders Publishing, 2004.

[MGR00] Stephen R. Marschner, Brian K. Guenter, and Sashi Raghupathy. Modeling and rendering for realistic facial animation. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 231–242. Springer-Verlag, 2000.

[MTPT88] N. Magnenat-Thalmann, N. E. Primeau, and D. Thalmann. Abstract muscle actions procedures for human face animation. *Visual Computer*, 3(5):290–297, 1988.

[MTT87] N. Magnenat-Thalmann and D. Thalmann. The direction of synthetic actors in the film *rendez-vous à montréal*. *IEEE Computer Graphics and Applications*, 7(12):9–19, December 1987.

[Mur04] Bruce Murray. English phonemes, spellings, example words, and meaningful names. Website, 2004. http://www.auburn.edu/~murraba/spellings.html.

[NN01] Jun-yong Noh and Ulrich Neumann. Expression cloning. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 277–288. ACM Press, 2001.

[NVI] NVIDIA Corporation. Santa Clara, CA, USA.

[O'R95] Michael O'Rourke. *Principles of Three-Dimensional Computer Animation*. W. W. Norton & Company, Inc., first edition, 1995.

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

[Par72a] Frederic I. Parke. Computer generated animation of faces. In *Proceedings of the ACM National Conference*, volume 1, pages 451–457, 1972.

[Par72b] Frederic I. Parke. Computer generated animation of faces. Master's thesis, University of Utah, 1972. UTEC-CSc-72-120.

[Par74] Frederic I. Parke. *A Parametric Model for Human Faces*. PhD thesis, University of Utah, 1974. UTEC-CSc-75-047.

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

[Pel02] Catherine Pelachaud. Visual text-to-speech. In Igor S. Pandzic and Robert Forchheimer, editors, *MPEG-4 Facial Animation: The Standard, Implementation and Applications*, chapter 8, pages 125–140. John Wiley & Sons, Ltd., 2002.

[PF02] Igor S. Pandzic and Robert Forchheimer, editors. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Ltd., 2002.

[PHL+98] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 75–84. ACM Press, 1998.

[Pir95] Mark Piretti. Realistic facial animation. Internal Report, Program of Computer Graphics, Cornell University, May 1995.

[PLY] Ply file format. Website. http://astronomy.swin.edu.au/~pbourke/-geomformats/ply/.

[PP01] S. Pasquariello and C. Pelachaud. Greta: A simple facial animation engine. In *Proc. of the 6th Online World Conference on Soft Computing in Industrial Applications, Session on Soft Computing for Intelligent 3D Agents*, September 2001.

[PT97] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer-Verlag, second edition, 1997.

[PTVF02] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, second edition, 2002.

[PW96] Frederic I. Parke and Keith Waters. *Computer Facial Animation*. A K Peters, Ltd., 1996.

[SB94] Robert Sekuler and Randolph Blake. *Perception*. McGraw-Hill, Inc., third edition, 1994.

[SF98] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414. ACM Press, 1998.

[TMPS03] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, 2003.

[TW90] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation. *Visualization and Computer Animation*, 1:73–80, 1990.

[Vid] FaceStation 2, Vidiator Technology (US) Inc. Website. http://www.vidiator.com/products_facestation.html.

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

[WT91] K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *Visualization and Computer Animation*, 2:123–128, 1991.

[ZSCS04] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *ACM Annual Conference on Computer Graphics*, page to appear, August 2004.

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