Geometry-based Muscle Modeling for Facial Animation

Kolja Kähler Jörg Haber Hans-Peter Seidel

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
{kaehler,haberj,hpseidel}@mpi-sb.mpg.de

Abstract

We present a muscle model and methods for muscle construction that allow to easily create animatable facial models from given face geometry. Using our editing tool, one can interactively specify coarse outlines of the muscles, which are then automatically created to fit the face geometry.

Our muscle model incorporates different types of muscles and the effects of bulging and intertwining muscle fibers. The influence of muscle contraction onto the skin is simulated using a mass-spring system that connects the skull, muscle, and skin layers of our model.

Key words: physics-based facial animation, muscle / skin model, muscle editor, mass-spring system

1 Introduction

Recently, the development of more and more accurate simulation of human characters based on their anatomy has led to *anatomically based modeling* as the bottom-up approach for building characters from bones, muscles, and skin.

For human faces, however, this approach is unsuitable if the target geometry is already given. Since the muscles of the face lie closely underneath the skin and have a great influence on the shape and appearance of the surface, it is difficult to model the skin from skull and muscles in such a way that the result bears close resemblance with the target face. On the other hand, surface geometry can easily be acquired using for instance a range scanner. Thus, our approach is to adapt the muscle geometry to the prescribed facial geometry.

To facilitate this task, we have developed an interactive muscle editor, which is depicted in Figure 14. The user can roughly sketch a layout of facial muscles, which are then automatically fitted to the given face mesh.

2 Previous Work

Techniques for animating human beings and human faces in particular have been an active area of research since the early 1980's [18, 15]. Apart from some recent image-based techniques [17, 1] and methods that apply previously captured facial expressions to a face model [8], the methods developed so far can be divided into two categories: parametric and physics-based models [16].

Parametric models control the shape of the skin by directly manipulating the geometry of the surface [15, 4]. WATERS [24] presented a muscle model which uses muscle vectors and radial functions derived from linear and sphincter muscles to deform a skin mesh. CHADWICK *et al.* [2] use free-form deformations to shape the skin in a multi-layer construction containing bones, muscles, fat tissue, and skin. A B-Spline surface model has been used by NAHAS *et al.* [14] to generate synthetic visual speech, while a variational approach is presented by DECARLO *et al.* [6] to generate novel synthetic face models using anthropometric statistics. The MPEG-4 standard [9] specifies a set of 68 facial animation parameters (FAPs) which can be applied to any suitable head model. GOTO *et al.* [7] use these FAPs to control their facial animation system. Though parametric models can be applied at relatively low computational costs, realistic blending between facial expressions is problematic [25]. Also, the range of possible skin deformations is limited.

Physics-based models typically use mass-spring or finite element networks to model the (visco-)elastic properties of skin [18, 12, 11]. WATERS and FRISBIE [25] proposed a two-dimensional mass-spring model of the mouth with the muscles represented as bands. A three-dimensional model of the human face has been developed by TERZOPOULOS and WATERS [21]. Their model consists of three layers (cutaneous tissue, subcutaneous fatty tissue, and muscles) that are embedded in a mass-spring system. Due to additional volume preservation constraints, this approach produces realistic results such as wrinkling at interactive frame rates. A framework for facial animation based on a simplified version of this model was presented by LEE *et al.* [13]. Their tissue model consists of two layers (dermal-fatty and muscles) and is connected by springs to a skull structure that is estimated from the surface data. The mass-spring system used in this approach also considers volume preservation and skull penetration constraints. The face model designed by WU *et al.* focuses on the viscoelastic properties of skin: muscles are represented by surfaces of revolution [28] or B-spline patches [27], which can be specified

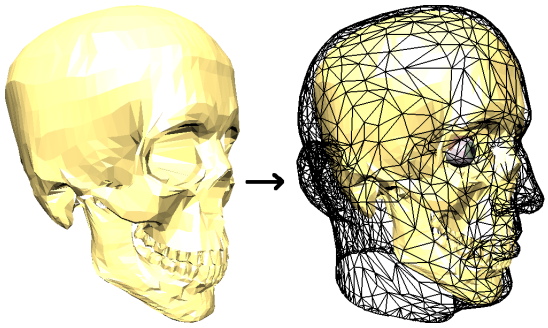

Figure 1: Generic skull model fitted to head using affine transformation for estimating assignment of skin regions to skull and jaw.

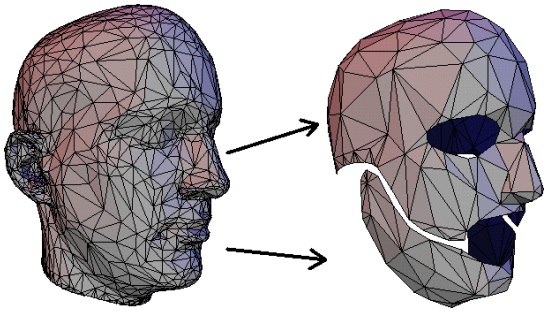

Figure 2: Head model prepared from scan data (left), cut up and simplified to represent the skull (right).

interactively. Their model is able to generate expressive wrinkles and skin aging effects. CHEN and ZELTZER [3] developed a finite element muscle model to simulate the deformation of individual muscles without an overlying skin tissue. Recently, SCHEEPERS *et al.* [19] and WILHELMS and VAN GELDER [26] presented anatomy-based muscle models for animating humans and animals. Their models incorporate skeletal bones and joints as well as muscle geometry. However, the skin tissue is represented only by an implicit surface with zero thickness [26].

3 Our Approach

Our model for muscle-based facial animation uses three conceptual layers:

- a skin / tissue layer representing the epidermis and subcutaneous fatty tissue;
- a layer of muscles attached to the skull and inserting into the skin;
- the underlying bone structure, composed of immovable skull and rotating jaw.

Our input data consists of an arbitrary triangle mesh representing the skin geometry, which is typically obtained from a range scanner. The skull geometry and the layout of the facial muscles are created semi-automatically, based on the face mesh. Animation of the face is achieved by physics-based simulation of a mass-spring system that connects the three layers of our model.

3.1 Skull and Jaw

Since we operate on models acquired from range data, we don't have access to the actual skull geometry. Instead, we use approximations of skull and jaw to which skin surface nodes and muscles are attached. Other than by computing a single offset surface [13], we distinguish between the fixed part of the skull and the movable jaw. We use the skull and jaw meshes to determine whether a part of the skin and muscle layers lies over the skull or over the jaw. For the latter, that part will follow the rotation of the jaw.

If a skull model is available, it can be aligned to the geometry by affine transformations. While it is generally not possible to match a generic skull to different human heads in this way, the approximation is good enough for assigning skin regions to skull or jaw, see Figure 1. Alternatively, an approximated skull model is obtained by cutting up the original input mesh, roughly separating the jaw from the rest of the head (cf. Figure 2). A standard mesh simplification algorithm [10] is applied, since we found that a coarse approximation of the bone structure is sufficient. Finally, the simplified geometry is scaled down (by a small offset determined by the skin thickness) and placed inside the head model. This approach is necessary for synthetic heads which have no real anatomical counterpart, see for instance Figure 13. The same skull model can be used without further work for multiple variations of the original head geometry, such as low and high resolution versions, or minor changes in facial details.

The skull and jaw meshes are used only while interactively building muscles in the editor and during the startup phase of the animation system. They are not used during the runtime of an animation, since skull penetration constraints are handled internally to the mass-spring mesh, cf. Section 3.3.

3.2 Muscles

Our muscle model is based on a piecewise linear representation similar to the one developed by LEE *et al.* [13], where isotonic contraction is expressed by shortening the linear segments. A muscle can either contract towards the end attached to the skull (*linear muscle*) or towards a point (*circular muscle*). In our model, each of the segments is additionally assigned an ellipsoidal shape.

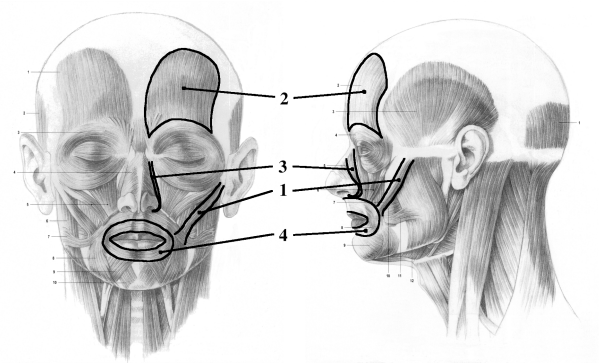

Figure 3: Different types of muscles supported by our model: linear (1), sheet (2), curved (3), and sphincter (4) muscles (original image from [20]).

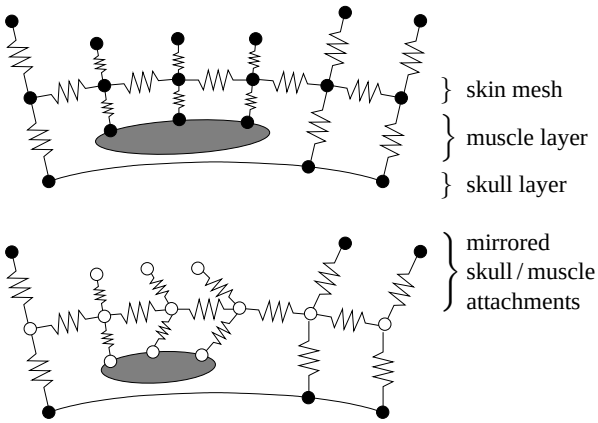

Figure 4: Mass-spring system in our model. Top: Relaxed muscle, outer springs mirroring skull and muscle attachments. Bottom: Contracted muscle, with mass points moving due to the contraction marked by ∘.

The piecewise linear muscle "fibers" can be combined into groups to form *sheet muscles*. Implicit surfaces for specifying the shape of muscles have been used before: SCHEEPERS *et al.* [19] also arrange ellipsoids to form more complex muscles. Their "general muscle model" uses bicubic patches, whereas our structure is composed of quadric segments. For construction of the muscles we need to perform operations on the segments that are readily available in the quadric representation: ray intersection tests, normal computation, and inside / outside tests [5]. For computation of deformed muscle shapes during animation only affine invariance is needed, so other segment shapes could be used efficiently as well.

Using this model, we can lay out muscles in the various configurations that appear in the human face: long and thin strands (*zygomatic major*) as well as broad sheets (*frontalis*), curved muscles (*levator labii sup. alaeque nasii*), and sphincters (*orbicularis oris*, though this muscle is in fact built from segments but usually approximated as a sphincter), see Figure 3.

Muscles are often layered, sliding freely across each other. As WATERS and FRISBIE point out [25], muscles may also intertwine and merge so that their actions are coupled, a fact that can be observed especially in the region around the mouth. In our model, muscles can merge in this way and move other muscles. To make for instance the lower part of *orbicularis oris* follow the rotation of the jaw when the mouth is opened, muscles can be attached to either the immovable skull or the rotatable jaw . We also follow WATERS and FRISBIE in that the muscles drive the animation and are not in turn moved by the skin. In reality, the *orbicularis oris* is pulled downwards by the skin when the jaw opens.

3.3 Skin and Tissue Simulation

The top layer of our model represents the skin. Currently we model elastic properties of the dermis and epidermis and the fatty layer underneath. The skin layer connects to muscles and bones, see Figure 4.

The nodes and edges of the input triangle mesh comprise the initial spring mesh. These springs are biphasic, i.e. they become stiffer under high strain, to roughly mimic the non-linear elastic properties of skin. The initial stiffness constants are computed according to VAN GELDER [22].

Each surface node is connected to either the bone layer or to an underlying muscle by a spring with low stiffness, simulating the fatty subcutaneous layer that allows skin to slide freely.

When the skin mesh is modeled as a simple membrane, it may penetrate the muscle and bone layers when stretched. Also, the mesh can easily fold over. Methods for local volume preservation and skull penetration constraints have already been proposed in [13]. We combine both requirements into one and attach another spring to each mesh node that pulls the node *outwards*, mirroring the spring that attaches it to the bone layer (cf. Figure 4). This can be interpreted as a model of the outward-directed force resulting from the internal pressure of a skin cell. Similarly, springs are added to mirror muscle attachments. However, these mirrored spring nodes move along with their counterparts when the muscle contracts. Thus a surface node preferably moves in a direction tangential to the skull and muscle surface. Thereby intersections are avoided in practice though not completely ruled out – violent distortion can still cause intersections. A nice property of this mechanism is the seamless integra-

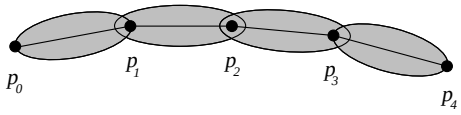

Figure 5: Muscle fiber with control polygon $P = \{p_i\}$ and per-segment ellipsoids.

tion into the spring mesh system: no special treatment of additional constraints is needed.

The equations of motion for the mass-spring system are numerically integrated through time using an explicit forward integration scheme (Verlet leapfrog [23]). To maximize stability, we measure the computation time for one time step of the simulation. The step size is dynamically adjusted to run as many small steps as possible in the time slot between two rendered frames, as determined by the given frame rate. In addition, we use an over-relaxation scheme to speed up the convergence of our simulation. This is accomplished by displacing the surface nodes to their estimated final position before invoking the solver. The estimation is based on muscle contraction and jaw rotation.

4 Muscle Model Details

Muscles are built from individual fibers that are in turn composed of piecewise linear segments. A quadric shape (ellipsoid) is aligned to each of these segments and scaled to the length of the muscle segment. The width and height of each ellipsoid correspond to the extent of the muscle parallel and orthogonal to the skin surface, respectively.

The initial description of a muscle fiber consists of n control points $p_i \in \mathbb{R}^3$ $(i = 0, \ldots, n-1)$ forming a control polygon P as shown in Figure 5.

4.1 Contraction

Given a contraction value $c \in [0, 1]$, where $c = 0$ means no contraction and $c = 1$ full contraction, a new control polygon $Q = \{q_i\}_{i=0}^{n-1}$ is computed (cf. Figure 6).

Each control point $p_i \in P$ is assigned a parameter $t_i \in [0, 1]$:

$$t_i := \begin{cases} 0 & \text{, if } \ i = 0, \\ \frac{\sum_{j=1}^{i} \|p_j - p_{j-1}\|}{\sum_{j=1}^{n-1} \|p_j - p_{j-1}\|} & \text{, else.} \end{cases}$$

The parameters t_i are scaled by the contraction factor $1-c$ and clamped to $[0.01, 1]$ to avoid shrinking a segment too much:

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

Next, we map each parameter $\tilde{t}_i$ to the index $k_i \in \{0, \ldots, n-2\}$ of the starting point of the segment that contains $\tilde{t}_i$:

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

Finally, we compute the new control points q_i by linear interpolation:

$$q_i \ := \ p_{k_i} + (p_{k_i+1} - p_{k_i}) \frac{\tilde{t}_i - t_{k_i}}{t_{k_i+1} - t_{k_i}}.$$

For sphincter muscles, segments are simply contracted towards a center point $p^* \in \mathbb{R}^3$:

$$q_i \ := \ p^* + (1 - c)(p_i - p^*).$$

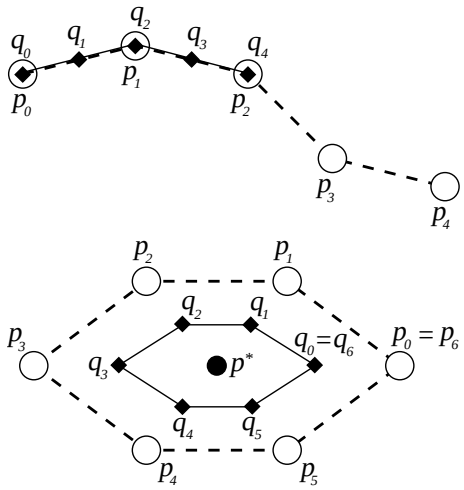

Figure 6: Contraction $(c = \frac{1}{2})$ of a linear (top) and a sphincter (bottom) muscle fiber. The control points $\{p_i\}$ and $\{q_i\}$ represent the relaxed and contracted muscle.

4.2 Bulge

Real muscles get thicker on contraction and thinner on elongation. Simulating this behavior enhances visual realism: when the face smiles, the lips retract slightly as they stretch. On the other hand, the lips get a little thicker, when the mouth forms an "o" or a kiss (cf. Figure 15).

For linear muscles, we want the center of the muscles to exhibit the highest bulge, corresponding to the belly of real muscles. Sphincters bulge evenly, see Figure 7.

In our model, bulging is achieved by scaling the height of each muscle segment $\overline{p_i\, p_{i+1}}$ by $(1 + 2s_i)$. Here, $s_i \in [0, 1]$ denotes the scaling factor computed from the length $l_i^r = \|p_{i+1} - p_i\|$ of the relaxed muscle and its current length $l_i^c = \|q_{i+1} - q_i\|$: $s_i := 1 - l_i^c / l_i^r$. This results in a center segment of triple height at maximum contraction.

For each linear muscle with at least three segments, we additionally multiply s_i by a simple quadratic function

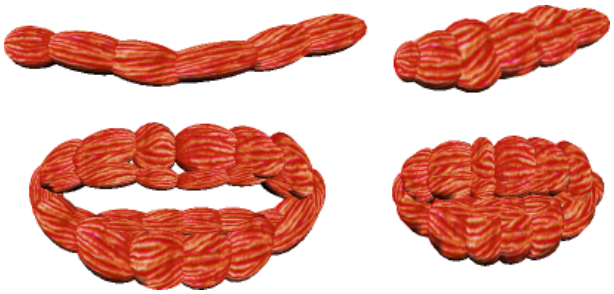

Figure 7: Relaxed (left) and contracted (right) muscles: a single fiber (top) and a sphincter (bottom) modeling the orbicularis oris

that vanishes over the first and last segment and has a maximum value of 1.0 over the central segment. In this case, the scaling factor s_i is computed as

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

Other, more accurate shape changes could be applied as well. For skeletal muscles, SCHEEPERS *et al.* developed a formulation that preserves volume as well as the ratio of width to height of the muscle belly [19].

4.3 Quadric Shapes

The transition of an original line segment $\overline{p_i\, p_{i+1}}$ into the transformed segment $\overline{q_i\, q_{i+1}}$ can be described by an affine transformation. This transformation is applied to the quadric associated with the segment. To keep the nodes of the spring mesh that attach to a muscle on the muscle surface, we simply apply the transformation to the attachment points as well. The mirrored muscle attachments (see Section 3.3) are transformed in the same way to keep the skin nodes above the muscle.

4.4 Intertwined Muscles

The end of a linear muscle can merge into another muscle, which is detected automatically by our system. This is achieved by testing whether the end point p_{n-1} of at least one fiber lies within the extent of another muscle. We only test the end points, because we still want muscles to cross without interacting. The muscle segments connected in this way are stored in *constraint groups*. After muscle contractions have been set, a constraint resolution phase moves the control points of the segments in each group such that the original distances between the control points is maintained. Muscle shape is computed only after resolving constraints (see Section 4.2), so that a muscle that is elongated by this mechanism will get thinner accordingly.

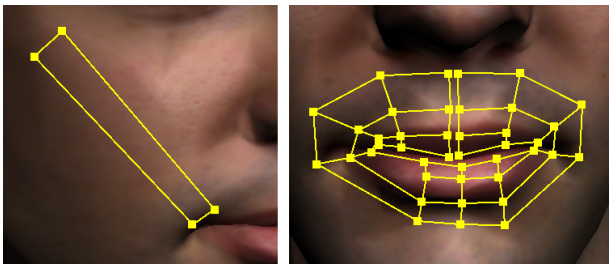

Figure 8: A simple grid (left, zygomatic major*) and a non-uniform complex grid (right,* orbicularis oris*).*

5 Building Muscles from Geometry

5.1 Overview

In our system, muscles are created automatically from coarse outlines that are interactively sketched onto the face model. From the user's point of view, the procedure follows these steps:

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

Besides this basic method for muscle construction, muscle grids can be re-edited by moving their control points around. The resulting muscle shape is immediately shown together with information about the influenced mesh vertices and connections to other muscles, see Figure 14.

The initial shape of the muscle outline can be of arbitrary detail, from the minimum of a quadrilateral up to highly complex grids, see Figure 8.

5.2 Optimizing Muscle Shape

The surface of a muscle must lie within a prescribed distance range below the skin surface. Additionally, we want to create muscles that are well-adapted to the resolution of the skin mesh: too highly refined muscles just add to the computational load during animation without enhancing the visual appearance, while too coarse muscles may not be following the surface closely enough to result in realistic deformations.

Given the skin mesh and a muscle grid, our optimization step determines the following parameters that are needed to create the muscles:

- the number of muscle fibers;
- the number of segments per fiber;
- width, height, and length of each segment;
- position of the muscle fiber control points;
- alignment of the quadrics' coordinate systems.

In addition to the regular grid, the skin thickness τ^s, which is assumed to be constant over the whole face, and the minimum and maximum muscle layer thickness τ^m_{min} and τ^m_{max} are input parameters of the optimization step. These parameters can be adjusted by the user based on the input geometry.

A muscle is created from its grid by a four-step procedure:

1. **Initializing the grid.** The initial outline is converted into a regular grid, i.e. all rows are assigned the same number of grid points. The grid points are then projected onto the face mesh and placed slightly underneath the skin surface.
2. **Refining the grid.** The grid is adaptively refined until a decent approximation has been found.
3. **Creating the muscle.** Muscle fibers are created and aligned to the refined grid.
4. **Attaching the muscle.** The muscle is attached to the spring mesh, and the control points of the muscle segments are attached to either the skull or the jaw.

Details of these steps are explained in the following sections.

5.3 Initializing the Grid

To obtain a regular grid, we first determine the maximum number n_{max} of grid points per row. Then, additional grid points are inserted by linear interpolation into every row that contains less than n_{max} points.

We now estimate normals at the grid points. For the various grid layouts we obtained best results by first computing the normal of the balancing plane through the four corner points of each grid cell and than averaging the normals of all adjacent cells at each grid point.

Having computed the grid point normals, we find the triangles of the face mesh that intersect the projection of the grid onto the skin surface and cache them for fast lookup during the iterative refinement procedure. The initial grid points are now displaced along their normal direction to lie below the skin in an initial distance of $\tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/4$, representing the middle of a muscle of average thickness running through the cell.

5.4 Refining the Grid

The fitting algorithm proceeds by sampling the distances from each cell to the skin surface. Each cell is examined and subdivided if necessary. The grid points are then again displaced to lie within the prescribed distance range below the surface. Simultaneously, the cell thickness is adjusted within the bounds τ^m_{min} and τ^m_{max}. This process is repeated until no more subdivisions are necessary or can be applied.

The main loop of this iteration is organized as follows:

```
repeat
  for each grid cell c
    (d_min, d_max, p_near, p_far) =
        minMaxDistancesToMesh(c);
    (e_near, e_far) =
        minMaxError(d_min, d_max, τ^s, τ^m_min, τ^m_max);
    if (e_near == 0 and e_far == 0)
      c.thickness = 2(d_min - τ^s)
    else if (e_far > e_near)
      trySubdivisionAtPoint(c, p_far);
    else
      trySubdivisionAtPoint(c, p_near);
  moveNewGridPoints();
until no more changes to grid.
```

The procedure `minMaxDistancesToMesh()` returns two points p_{near},p_{far} that are nearest to and farthest away from the cell in the following sense: we adaptively subsample the grid cell and shoot a ray from each sample position in the direction of the associated bilinearly interpolated grid normal vector. The base points of the rays with the nearest and farthest intersection points with the cached surface area are returned as p_{near} and p_{far} along with their signed distance values d_{min} and d_{max}. Both points can be positioned below (positive distance value) or above the skin surface (negative value), see Figure 9. The sampling density over the grid cell adjusts to the size of the cell and the number of cached triangles to ensure a minimum number of samples per triangle.

The error values e_{near} and e_{far} that are computed by `minMaxError()` represent the unsigned distances p_{near} and p_{far} would have to move along their grid normal to be in the allowed range of distances $[\tau^s + \tau^m_{\text{min}}, \tau^s + \tau^m_{\text{max}}]$ below the skin surface (see Figure 10). Also, if $|d_{\text{max}} - d_{\text{min}}| > \tau^m_{\text{min}}$, the distance from the cell to the skin varies widely over the cell area, so that there is enough space for insertion of a thin muscle. In this case, $(e_{\text{near}}, e_{\text{far}})$ are set to $(|d_{\text{min}}|, |d_{\text{max}}|)$, causing a subdivision of the cell in the next step.

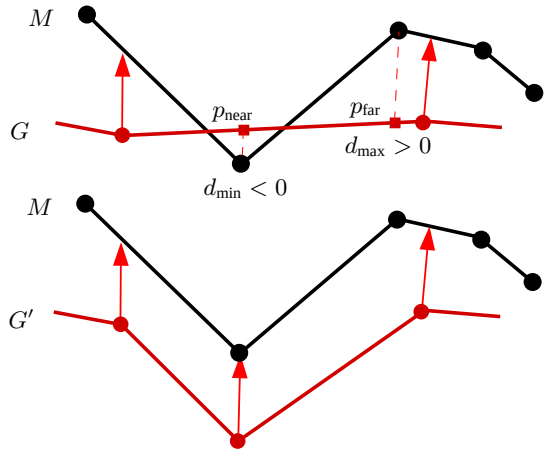

Figure 9: Refinement step for a single grid cell (simplified two-dimensional view). Top: Points of grid G have been placed below the skin mesh M along their associated normals. The closest point of the grid cell lies above, the farthest point lies below the skin mesh. Bottom: G has been subdivided at the point of larger error e_{near} (see also Figure 10).

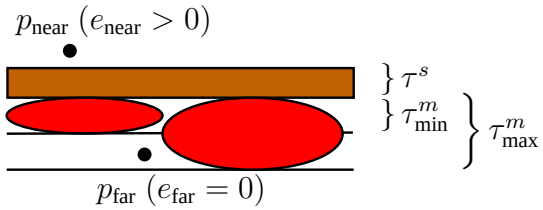

Figure 10: The range of thickness for muscle shapes: below the skin layer of constant thickness τ^s muscles can be inserted with a thickness in the range $[\tau^m_{min}, \tau^m_{max}]$. Error values for two exemplary points are shown: p_{near} is outside the allowed range for muscle segments and should be moved upwards ($e_{near} > 0$). p_{far} is within range and need not be moved ($e_{far} = 0$).

The procedure `trySubdivisionAtPoint()` is called with the sample position corresponding to the point with the larger error. A new row and/or column through that position is inserted into the grid. Before subdividing a cell along one of its dimensions, we compare the sizes of the resulting sub-cells with the average extent of the cached triangles in that direction. If the sub-cells would get too small, the insertion point is adjusted to make both parts big enough. If the cell is already too small to allow for adjustment, no subdivision along this direction is performed.

Finally, in `moveNewGridPoints()` the grid points inserted by subdivision are projected onto the surface mesh and displaced by $\tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/4$ underneath the skin.

5.5 Creating the Muscle

After a grid has been refined sufficiently, we build a sheet of muscle fibers. One muscle fiber is inserted longitudinally into each stripe of grid cells, creating one muscle segment per cell. The size of each ellipsoid is scaled to fill the surrounding cell, whereas width and length of interior ellipsoids are slightly enlarged to provide some overlap across cell boundaries. Figure 11 shows the creation of a sheet muscle from a simple grid.

5.6 Attaching the Muscle to Skin

Muscles have to be connected to the spring mesh, so that contraction will influence nearby skin vertices. We consider the vertices within a specified radius of influence from the muscle fibers as candidates for muscle attachment: for each of these skin nodes, we compute the closest point on the surface of all quadrics comprising the muscle sheet and insert a spring connecting the skin node with that point. An additional spring is created as described in Section 3.3 by mirroring the attachment point.

There are special cases where the distance-based computation of attachment points is not sufficient. For instance, when the face mesh has a closed mouth, vertices along the cut separating the upper and lower lip will have almost – if not exactly – the same coordinates. These vertices may thus be attached to the muscles around the upper and lower lips in a nondeterministic way. This will likely cause the upper lip to move along with the lower *orbicularis oris* and vice versa. To solve this problem, we weight the distance value of each skin node with the dot product $N_s N_d$, where N_s is the surface normal at the skin vertex and N_d is the normalized vector pointing from the potential attachment point to that vertex. Thereby muscle segments that lie directly below the skin vertex are favored.

5.7 Attaching the Muscle to Skull and Jaw

To find out whether a muscle control point should move along with the jaw or remain fixed, we shoot rays from the grid points along their normals through the skin mesh and examine the attachment of the closest skin vertex in the hit triangle. If the majority of points in a grid row is closest to skull-attached skin vertices, the corresponding muscle attachments will also be fixed. Otherwise, muscles will be attached to the jaw, if the closest skin vertices are mostly assigned to the jaw.

Not all regions of the face have bones underneath, e.g. the lips and the cheeks. Skin vertices in these regions are thus not attached to the bone structure. To decide about the muscle attachment in these cases, we iteratively grow

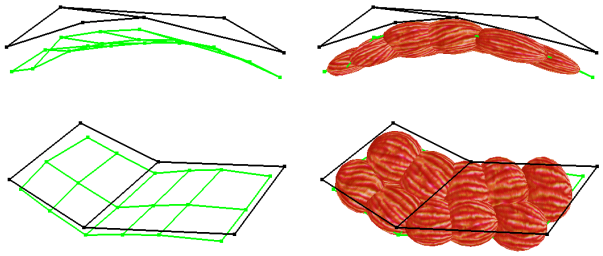

Figure 11: Side and top view of a coarse (black) and refined (green) grid and the muscle created from it.

the topological neighborhood of the skin mesh around the intersection point until a bone-attached skin node is found. Using this technique, the upper part of the *orbicularis oris* is properly assigned to the immovable skull, while the lower part is attached to the jaw.

6 Results

We have tested our editing tool on human head models obtained from range scans (see Figures 12,15) and synthetic data (see Figure 13). After the geometry had been prepared, creating a set of facial muscles varying widely in shape as described in Section 3.2 took only a few minutes. Some tweaking and experimentation was usually necessary, though, to achieve good results in animation. Especially with coarse triangle meshes, a small change in muscle layout may determine whether a large nearby triangle is influenced by a contraction of that muscle or not. Here we found it to be important to have good re-editing facilities and visual feedback about how muscles attach to the skin.

One advantage of our muscle grid fitting approach is, that once a rough layout for a muscle has been specified, the muscle can be automatically rebuilt from this data for many different head models. Muscles adjusted to different mesh resolutions of one head model, e.g. for real-time rendering or high quality animation, can easily be created.

The refinement loop (Section 5.4) typically takes only a few iterations to create a well-adapted grid that reflects the curvature and resolution of the skin mesh, allowing interactive re-editing of the muscle layout with immediate feedback of the resulting muscle shape. Furthermore, the initial muscle layout is preserved by our fitting algorithm: design decisions made by the user shouldn't be overridden.

The detail views in Figure 15 show some important features of our muscle model. In the top right image, the muscles are relaxed and the jaw is slightly rotated to open the mouth. The lower part of *orbicularis oris* has moved along with the jaw. Segments from other muscles that have been automatically attached to the *orbicularis oris* have followed the movement. In the lower right image, the mouth additionally forms an "o" by contracting the *orbicularis oris*. The muscle bulges accordingly, protruding the lips slightly while the attached muscles are thinning due to the elongation.

In our current experimental implementation of muscle model and spring-mesh simulator, we achieve interactive frame rates (5 fps on an `sgi` O2, 16 fps on a fast PC) for models with a low polygon count of about 3000 triangles. Figure 12 shows some snapshots taken from an animation. For movies and other current project information, please visit our web site: `http://www.mpi-sb.mpg.de/resources/FAM/`.

7 Conclusion and Future Work

We have developed a muscle model and an accompanying muscle editing tool that allows for fast and easy generation of physics-based animatable face models from real world geometry. Once the geometry has been prepared, the model can be brought to life within minutes. New degrees of freedom for animation are easily introduced into a model by adding new muscles.

Preparing the scanned head geometry was the most time-consuming part of the process: several hours went into fixing the scans, clearly making this the bottleneck in the creation of the animated model. Since manual preparation of skull models also is a time-consuming task, fitting a generic model of a real human skull to the skin mesh is desirable. Our experiments have shown that a more sophisticated approach is necessary for a precise fit. More accurate fitting would also allow us to use non-constant skin thickness in the simulation, and have muscle thickness change locally according to the available space. In the same vein, a more precise deformation increases the precision when automatically aligning generic sets of muscle to other heads.

A promising avenue for research is provided by the ability to automatically create muscles adapted to face meshes of different resolution: employing level-of-detail techniques, multi-resolution facial animation can be achieved with little effort.

In general, the fitting algorithm delivers good results. However, it showed that especially in non-uniform regions of the facial mesh, the termination criterion of the refinement loop is not always adequate. In those cases we had to manually adjust the parameters of the fitting procedure, making the process not fully automated. Also the current method doesn't take into account the actual segment shape during refinement. A more sophisticated

approach would probably lead to better approximations of the skin curvature with fewer muscle segments.

The muscle model itself performs well with low computational overhead. We think it would be worthwhile to add elastic behavior to the muscles themselves, thus allowing them to straighten under tension (contraction and elongation) and producing more realistic deformations of merged muscles.

Acknowledgements

The authors are grateful to their "head model" Mario Botsch and to Christian Rössl for operating the range scanner.

References

[1] V. Blanz and T. Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 187–194, August 1999.

[2] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered Construction for Deformable Animated Characters. In *Computer Graphics (SIGGRAPH '89 Conf. Proc.)*, pages 243–252, July 1989.

[3] D. T. Chen and D. Zeltzer. Pump it up: Computer Animation of a Biomechanically Based Model of Muscle using the Finite Element Method. In *Computer Graphics (SIGGRAPH '92 Conf. Proc.)*, pages 89–98, July 1992.

[4] M. M. Cohen and D. W. Massaro. Modeling Coarticulation in Synthetic Visual Speech. In *Models and Techniques in Computer Animation*, pages 139–156. Springer–Verlag, 1993.

[5] J. M. Cychosz and W. N. Waggenspeck, Jr. Intersecting a Ray with a Quadric Surface. In *Graphics Gems III*, pages 275–283. Academic Press, London, 1992.

[6] D. DeCarlo, D. Metaxas, and M. Stone. An Anthropometric Face Model using Variational Techniques. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 67–74, July 1998.

[7] T. Goto, M. Escher, C. Zanardi, and N. Magnenat-Thalmann. MPEG-4 based Animation with Face Feature Tracking. In *Proc. Eurographics Workshop on Computer Animation and Simulation '99*, pages 89–98, 1999.

[8] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making Faces. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 55–66, July 1998.

[9] ISO/IEC. Overview of the MPEG-4 Standard. `http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm`, July 2000.

[10] L. Kobbelt, S. Campagna, and H.-P. Seidel. A General Framework for Mesh Decimation. In *Proc. Graphics Interface '98*, pages 43–50, June 1998.

[11] R. M. Koch, M. H. Groß, and A. A. Bosshard. Emotion Editing using Finite Elements. In *Computer Graphics Forum (Proc. Eurographics '98)*, volume 17, pages C295–C302, September 1998.

[12] Y. Lee, D. Terzopoulos, and K. Waters. Constructing Physics-based Facial Models of Individuals. In *Proc. Graphics Interface '93*, pages 1–8, May 1993.

[13] Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animations. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 55–62, August 1995.

[14] M. Nahas, H. Huitric, and M. Saintourens. Animation of a B-Spline Figure. *The Visual Computer*, 3(5):272–276, March 1988.

[15] F. I. Parke. Parameterized Models for Facial Animation. *IEEE Computer Graphics and Applications*, 2(9):61–68, November 1982.

[16] F. I. Parke and K. Waters, editors. *Computer Facial Animation*. A K Peters, Wellesley, MA, 1996.

[17] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 75–84, July 1998.

[18] S. M. Platt and N. I. Badler. Animating Facial Expressions. In *Computer Graphics (SIGGRAPH '81 Conf. Proc.)*, pages 245–252, August 1981.

[19] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy-Based Modeling of the Human Musculature. In *Computer Graphics (SIGGRAPH '97 Conf. Proc.)*, pages 163–172, August 1997.

[20] A. Szunyoghy and G. Fehér. *Menschliche Anatomie für Künstler*. Könemann, Köln, 2000.

[21] D. Terzopoulos and K. Waters. Physically-based Facial Modelling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1(2):73–80, December 1990.

[22] A. Van Gelder. Approximate Simulation of Elastic Membranes by Triangulated Spring Meshes. *Journal of Graphics Tools*, 3(2):21–41, 1998.

[23] F. J. Vesely. *Computational Physics: An Introduction*. Plenum Press, New York, 1994.

[24] K. Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In *Computer Graphics (SIGGRAPH '87 Conf. Proc.)*, pages 17–24, July 1987.

[25] K. Waters and J. Frisbie. A Coordinated Muscle Model for Speech Animation. In *Proc. Graphics Interface '95*, pages 163–170, May 1995.

[26] J. Wilhelms and A. Van Gelder. Anatomically Based Modeling. In *Computer Graphics (SIGGRAPH '97 Conf. Proc.)*, pages 173–180, August 1997.

[27] Y. Wu, P. Kalra, L. Moccozet, and N. Magnenat-Thalmann. Simulating Wrinkles and Skin Aging. *The Visual Computer*, 15(4):183–198, 1999.

[28] Y. Wu, N. Magnenat-Thalmann, and D. Thalmann. A Plastic-Visco-Elastic Model for Wrinkles in Facial Animation and Skin Aging. In *Proc. Pacific Graphics '94*, pages 201–214, August 1994.

Figure 12: Snapshots from an animation sequence (left to right). The face mesh consists of 3246 triangles. The animation runs with 5 fps on an `sgi` O2 (250 MHz) and with 16 fps on a 1.1 GHz Linux PC.

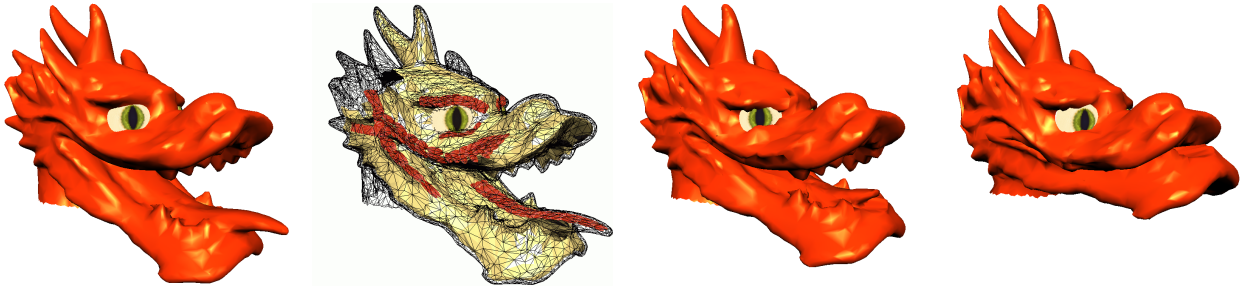

Figure 13: Construction of an animatable model from artificial head geometry. Left to right: Input mesh with eyes added; approximated "skull" and user-designed muscles; fierce expression; sad expression.

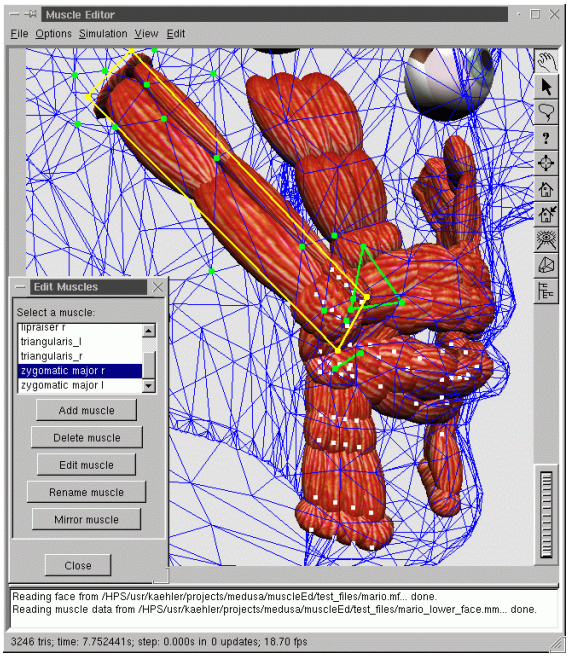

Figure 14: Visual information while editing a muscle: the muscle grid of the currently edited muscle (yellow); the skin vertices influenced by this muscle (green dots); muscle control points attached to the jaw (white dots); merged muscle segments (connected by green lines).

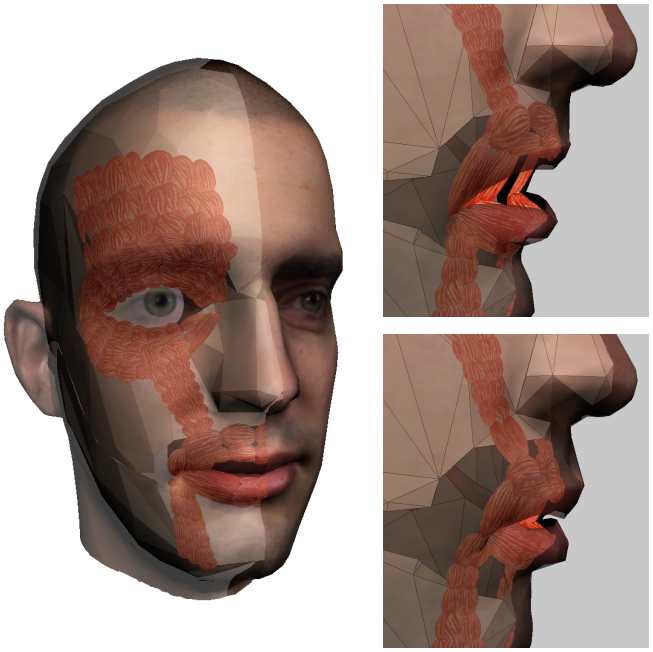

Figure 15: Left: The head model contains skull and jaw, eyeballs, and several groups of muscles. To display these interior components, the right half of the facial skin is rendered semi-transparent. Right: Muscles around the mouth: relaxed, mouth slightly open (top) and contracted, lips forming an "o" (bottom).

Errata

This electronic version of the document contains three corrections over the original version published in the proceedings of Graphics Interface 2001:

page 4, right column, equation for sphincter contraction	$\|\cdot\| \rightarrow (\cdot)$
page 6, right column, line 2	$\tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/2 \rightarrow \tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/4$
page 7, left column, line 14	$\tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/2 \rightarrow \tau^s + (\tau^m_{\text{min}} + \tau^m_{\text{max}})/4$