Presented at the Eleventh Eurographics Rendering Workshop, June 2000.

Modeling and Rendering for Realistic Facial Animation

Stephen R. Marschner Brian Guenter Sashi Raghupathy
Microsoft Corporation[1]

Abstract. Rendering realistic faces and facial expressions requires good models for the reflectance of skin and the motion of the face. We describe a system for modeling, animating, and rendering a face using measured data for geometry, motion, and reflectance, which realistically reproduces the appearance of a particular person's face and facial expressions. Because we build a complete model that includes geometry and bidirectional reflectance, the face can be rendered under any illumination and viewing conditions. Our face modeling system creates structured face models with correspondences across different faces, which provide a foundation for a variety of facial animation operations.

1 Introduction

Modeling and rendering realistic faces and facial expressions is a difficult task on two levels. First, faces have complex geometry and motion, and skin has reflectance properties that are not modeled well by the shading models (such as Phong-like models) that are in wide use; this makes rendering faces a technical challenge. Faces are also a very familiar—possibly the most familiar—class of images, and the slightest deviation from real facial appearance or movement is immediately perceived as wrong by the most casual viewer.

We have developed a system that takes a significant step toward solving this difficult problem to this demanding level of accuracy by employing advanced rendering techniques and using the best available measurements from real faces wherever possible. Our work builds on previous rendering, modeling, and motion capture technology and adds new techniques for diffuse reflectance acquisition, structured geometric model fitting, and measurement-based surface deformation to integrate this previous work into a realistic face model.

2 Previous Work

Our system differs from much previous work in facial animation, such as that of Lee et al. [12], Waters [21], and Cassel et al. [2], in that we are not synthesizing animations using a physical or procedural model of the face. Instead, we capture facial movements in three dimensions and then replay them. The systems of Lee et al. and Waters are designed to make it relatively easy to animate facial expression manually. The system of Cassel et al. is designed to automatically create a dialog rather than to faithfully reconstruct a particular person's facial expression. The work of Williams [22] is more

[1]Email: stevemar@microsoft.com, bguenter@microsoft.com, sashir@microsoft.com.

similar to ours, but he used a single static texture image of a real person's face and tracked points only in 2D. Since we are only concerned with capturing and reconstructing facial performances, our work is unlike that of Essa and Pentland [6], which attempts to recognize expressions, or that of DeCarlo and Metaxas [5], which can track only a limited set of facial expressions.

The reflectance in our head model builds on previous work on measuring and representing the bidirectional reflectance distribution function, or BRDF [7]. Lafortune et al. [10] introduced a general and efficient representation for BRDFs, which we use in our renderer, and Marschner et al. [15] made image-based BRDF measurements of human skin, which serve as the basis for our skin reflection model. The procedure for computing the albedo map is related to some previous methods that compute texture for 3D objects, some of which deal with faces [16, 1] or combine multiple images [17] and some of which compute lighting-independent textures [23, 19, 18]. However, the technique presented here, which is closely related to that of Marschner [14], is unique in performing illumination correction with controlled lighting while at the same time merging multiple camera views on a complex curved surface.

Our procedure for consistently fitting the face with a generic model to provide correspondence and structure builds on the method of fitting subdivision surfaces due to Hoppe et al. [9]. Our version of the fitting algorithms adds vertex-to-point constraints that enforce correspondence of features, and includes a smoothing term that is necessary for the iteration to converge in the presence of these correspondences.

Our method for moving the mesh builds on previous work using the same type of motion data [8]. The old technique smoothed and decreased motions, but worked well enough to provide a geometry estimate for image-based reprojection; this paper adds additional computations required to reproduce the motion well enough that the shading on the geometry alone produces a realistic face.

The original contributions of this paper enter into each of the parts of the face modeling process. To create a structured, consistent representation of geometry, which forms the basis for our face model and provides a foundation for many further face modeling and rendering operations, we have extended previous surface fitting techniques to allow a generic face to be conformed to individual faces. To create a realistic reflectance model we have made the first practical use of recent skin reflectance measurements and added newly measured diffuse texture maps using an improved texture capture process. To animate the mesh we use improved techniques that are needed to produce surface shapes suitable for high-quality rendering.

3 Face Geometry Model

The geometry of the face consists of a skin surface plus additional surfaces for the eyes. The skin surface is derived from a laser range scan of the head and is represented by a subdivision surface with displacement maps. The eyes are a separate model that is aligned and merged with the skin surface to produce a complete face model suitable for high-quality rendering.

3.1 Mesh fitting

The first step in building a face model is to create a subdivision surface that closely approximates the geometry measured by the range scanner. Our subdivision surfaces are defined from a coarse triangle mesh using Loop's subdivision rules [13] with the

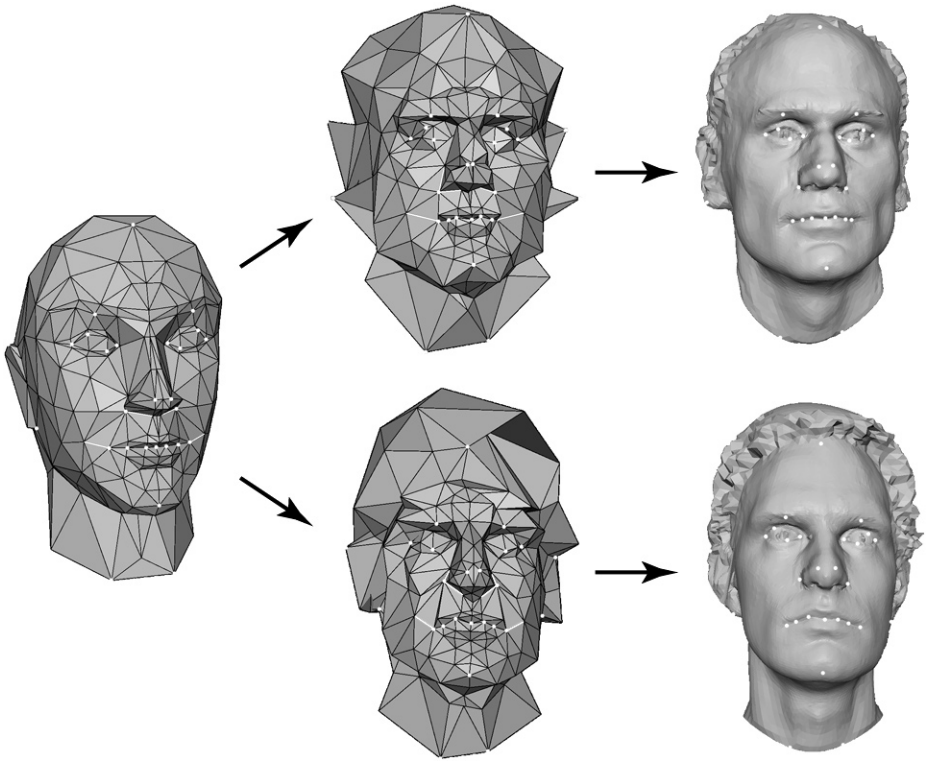

Fig. 1. Mapping the same subdivision control mesh to a displaced subdivision surface for each face results in a structured model with natural correspondence from one face to another.

addition of sharp edges similar to those described by Hoppe et al. [9].[2]

A single base mesh is used to define the subdivision surfaces for all our face models, with only the vertex positions varying to adapt to the shape of each different face. Our base mesh, which has 227 vertices and 416 triangles, is designed to have the general shape of a face and to provide greater detail near the eyes and lips, where the most complex geometry and motion occur. The mouth opening is a boundary of the mesh, and it is kept closed during the fitting process by tying together the positions of the corresponding vertices on the upper and lower lips. The base mesh has a few edges marked for sharp subdivision rules (highlighted in white in Figure 1); they serve to create corners at the two sides of the mouth opening and to provide a place for the sides of the nose to fold. Because our modified subdivision rules only introduce creases for chains of at least three sharp edges, our model does not have creases in the surface; only isolated vertices fail to have well-defined limit normals.

The process used to fit the subdivision surface to each face is based on the algorithm described by Hoppe et al. [9]. The most important differences are that we perform only the continuous optimization over vertex positions, since we do not want to alter the

[2] We do not use the non-regular crease masks, and when subdividing an edge between a dart and a crease vertex we mark only the new edge adjacent to the crease vertex as a sharp edge.

connectivity of the control mesh, and that we add feature constraints and a smoothing term. The fitting process minimizes the functional:

$$E(\mathbf{v}) = E_d(\mathbf{v}, \mathbf{p}) + \lambda E_s(\mathbf{v}) + \mu E_c(\mathbf{v})$$

where $\mathbf{v}$ is a vector of all the vertex positions, and $\mathbf{p}$ is a vector of all the data points from the range scanner. The subscripts on the three terms stand for distance, shape, and constraints.

The distance functional E_d measures the sum-squared distance from the range scanner points to the subdivision surface:

$$E_d(\mathbf{v}, \mathbf{p}) = \sum_{i=1}^{n_p} a_i \|p_i - \Pi(\mathbf{v}, p_i)\|^2$$

where p_i is the i^{th} range point and $\Pi(\mathbf{v}, p_i)$ is the projection of that point onto the subdivision surface defined by the vertex positions $\mathbf{v}$. The weight a_i is a Boolean term that causes points to be ignored when the scanner's view direction at p_i is not consistent with the surface normal at $\Pi(\mathbf{v}, p_i)$. We also reject points that are farther than a certain distance from the surface:

$$a_i = \begin{cases} 1 & \text{if } \langle s(p_i), n(\Pi(\mathbf{v}, p_i)) \rangle > 0 \text{ and } \|p_i - \Pi(\mathbf{v}, p_i)\| < d_0 \\ 0 & \text{otherwise} \end{cases}$$

where $s(p)$ is the direction toward the scanner's viewpoint at point p and $n(x)$ is the outward-facing surface normal at point x.

The smoothness functional E_s encourages the control mesh to be locally planar. It measures the distance from each vertex to the average of the neighboring vertices:

$$E_s(\mathbf{v}) = \sum_{j=1}^{n_v} \left\| v_j - \frac{1}{\deg(v_j)} \sum_{i=1}^{\deg(v_j)} v_{k_i} \right\|^2$$

The vertices v_{k_i} are the neighbors of v_j.

The constraint functional E_c is simply the sum-squared distance from a set of constrained vertices to a set of corresponding target positions:

$$E_c(\mathbf{v}) = \sum_{i=1}^{n_c} \|A_{c_i}\mathbf{v} - d_i\|^2$$

A_j is the linear function that defines the limit position of the j^{th} vertex in terms of the control mesh, so the limit position of vertex c_i is attached to the 3D point d_i. The constraints could instead be enforced rigidly by a linear reparameterization of the optimization variables, but we found that the soft-constraint approach helps guide the iteration smoothly to a desirable local minimum. The constraints are chosen by the user to match the facial features of the generic mesh to the corresponding features on the particular face being fit. Approximately 25 to 30 constraints (marked with white dots in Figure 1) are used, concentrating on the eyes, nose, and mouth.

Minimizing $E(\mathbf{v})$ is a nonlinear least-squares problem, because Π and a_i are not linear functions of $\mathbf{v}$. However, we can make it linear by holding a_i constant and approximating $\Pi(\mathbf{v}, p_i)$ by a fixed linear combination of control vertices. The fitting

process therefore proceeds as a sequence of linear least-squares problems with the a_i and the projections of the p_i onto the surface being recomputed before each iteration. The subdivision limit surface is approximated for these computations by the mesh at a particular level of subdivision. Fitting a face takes a small number of iterations (fewer than 20), and the constraints are updated according to a simple schedule as the iteration progresses, beginning with a high λ and low μ to guide the optimization to a very smooth approximation of the face, and progressing to a low λ and high μ so that the final solution fits the data and the constraints closely. The computation time in practice is dominated by computing $\Pi(\mathbf{v}, p_i)$.

To produce the mesh for rendering we subdivide the surface to the desired level, producing a mesh that smoothly approximates the face shape, then compute a displacement for each vertex by intersecting the line normal to the surface at that vertex with the triangulated surface defined by the original scan [11]. The resulting surface reproduces all the salient features of the original scan in a mesh that has somewhat fewer triangles, since the base mesh has more triangles in the more important regions of the face. The subdivision-based representation also provides a parameterization of the surface and a built-in set of multiresolution basis functions defined in that parameterization and, because of the feature constraints used in the fitting, creates a natural correspondence across all faces that are fit using this method. This structure is useful in many ways in facial animation, although we do not make extensive use of it in the work described in this paper; see Section 7.1.

3.2 Adding eyes

The displaced subdivision surface just described represents the shape of the facial skin surface quite well, but there are several other features that are required for a realistic face. The most important of these is the eyes. Since our range scanner does not capture suitable information about the eyes, we augmented the mesh for rendering by adding separately modeled eyes. Unlike the rest of the face model, the eyes and their motions (see Section 4.2) are not measured from a specific person, so they do not necessarily reproduce the appearance of the real eyes. However, their presence and motion is critical to the overall appearance of the face model.

The eye model (see Figure 2), which was built using a commercial modeling package, consists of two parts. The first part is a model of the eyeball, and the second part is a model of the skin surface around the eye, including the eyelids, orbit, and a portion of the surrounding face (this second part will be called the "orbit surface"). In order for the eye to become part of the overall face model, the orbit surface must be made to fit the individual face being modeled, and the two surfaces must be stitched together. This is done in two steps: first the two meshes are warped according to a weighting function defined on the orbit surface, so that the face and orbit are coincident where they overlap. Then the two surfaces are cut with a pair of concentric ellipsoids and stitched together into a single mesh.

4 Moving the Face

The motions of the face are specified by the time-varying 3D positions of a set of sample points on the face surface. When the face is controlled by motion-capture data these points are the markers on the face that are tracked by the motion capture system, but facial motions from other sources (see Section 7.1) can also be represented in this way. The motions of these points are used to control the face surface by way of a set of

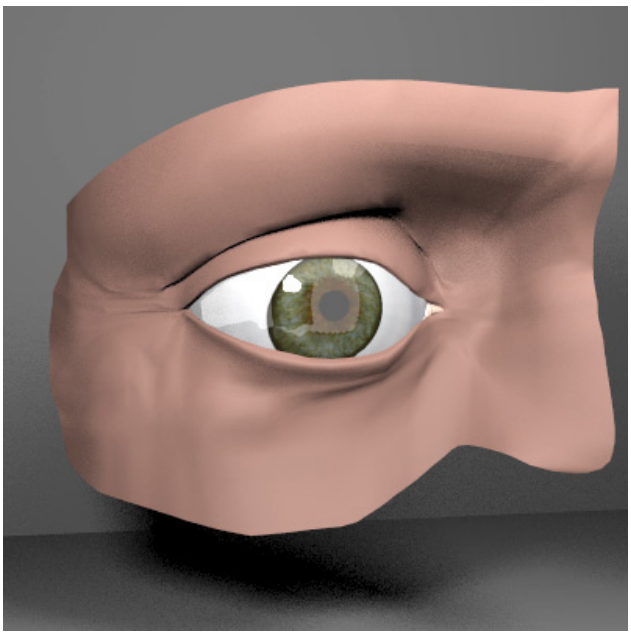

Fig. 2. The eye model.

control points that smoothly influence regions of the surface.

A discussion of the various methods for capturing facial motion is beyond the scope of this paper; we used the method of Guenter et al. [8] to acquire our face motion data.

4.1 Mesh deformation

The face is animated by displacing each vertex w_i of the triangle mesh from its rest position according to a linear combination of the displacements of a set of control points q_j. These control points correspond one-to-one with the sample points p_j that describe the motion. The influence of each control point on the vertices falls off with distance from the corresponding sample point, and where multiple control points influence a vertex their weights are normalized to sum to 1.

$$\Delta w_i = \frac{1}{\beta_i} \sum_j \alpha_{ij} \Delta q_j \quad ; \quad \alpha_{ij} = h(\|w_i - p_j\|/r)$$

$\beta_i = \sum_k \alpha_{ik}$ if vertex i is influenced by multiple control points and 1 otherwise. The parameter r controls the radius of influence of the control points. These weights are computed once, using the rest positions of the sample points and face mesh, so that moving the mesh for each frame is just a sparse matrix multiplication. For the weighting function we used $h(x) = \frac{1}{2} + \frac{1}{2} cos(\pi x)$.

Two types of exceptions to these weighting rules are made to handle the particulars of animating a face. Vertices and control points near the eyes and mouth are tagged as “above” and “below,” and controls that are, for example, above the mouth do not

influence the motions of vertices below the mouth. Also, a scalar texture map in the region around the eyes is used to weight the motions so that they taper smoothly to zero at the eyelids.

To move the face mesh according to a set of sample points, control point positions must be computed that will deform the surface appropriately. Using the same weighting functions described above, we compute how the sample points move in response to the control points. The result is a linear transformation: $\mathbf{p} = A\mathbf{q}$. Therefore if at time t we want to achieve the sample positions $\mathbf{p}_t$, we can use the control positions $\mathbf{q}_t = A^{-1}\mathbf{p}_t$. However, the matrix A can be ill-conditioned, so to avoid the undesirable surface shapes that are caused by very large control point motions we compute A^{-1} using the SVD and clamp the singular values of A^{-1} at a limit M. We used $M = 1.5$ for the results shown in this paper.[3]

4.2 Eye and head movement

In order to give the face a more lifelike appearance, we added procedurally generated motion to the eyes and separately captured rigid-body motion to the head as a whole.

The eyeballs are rotated according to a random sequence of fixation directions, moving smoothly from one to the next. The eyelids are animated by rotating the vertices that define them about an axis through the center of the eyeball, using weights defined on the eyelid mesh to ensure smooth deformations.

The rigid-body motion of the head is captured from the physical motion of a person's head by filming that motion while the person is wearing a hat marked with special machine-recognizable targets (the hat is patterned closely on the one used by Marschner et al. [15]). By tracking these targets in the video sequence, we computed the rigid motion of the head, which we then applied to the head model for rendering. This setup, which requires just a video camera, provides a convenient way to author head motion by demonstrating the desired actions.

5 Face Rendering

Rendering a realistic image of a face requires not just accurate geometry, but also accurate computation of light reflection from the skin, so we use a physically-based Monte Carlo ray tracer [3, 20] to render the face. This allows us to use arbitrary BRDFs to correctly simulate the appearance of the skin, which is not well approximated by simple shading models. The renderer also supports extended light sources, which, in rendering as in portrait photography, are needed to achieve a pleasing image. Two important deviations from physical light transport are made for the sake of computational efficiency: diffuse interreflection is disregarded, and the eyes are illuminated through the cornea without refraction.

Our reflectance model for the skin is based on the measurements of actual human faces made by Marschner et al. [15]. The measurements describe the average BRDFs of several subjects' foreheads and include fitted parameters for the BRDF model of Lafortune et al. [10], so they provide an excellent starting point for rendering a realistic face. However, they need to be augmented to include some of the spatial variation observed in actual faces. We achieve this by starting with the fit to the measured BRDF of one subject whose skin is similar to the skin of the face we rendered and dividing it into diffuse and specular components, then introducing a texture map to modulate each.

[3] The first singular value of A is 1.0.

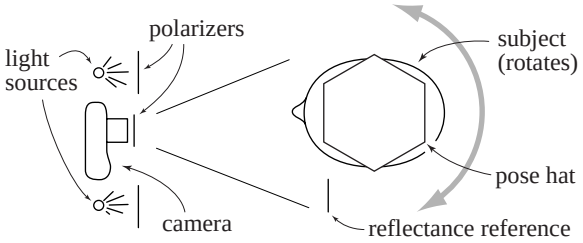

Fig. 3. Setup for measuring albedo maps.

The texture map for the diffuse component, or the albedo map, modulates the diffuse reflectance according to measurements taken from the subjects' actual faces as described in the next section. The specular component is modulated by a scalar texture map to remove specularity from areas (such as eyebrows and hair) that should not be rendered with skin reflectance and to reduce specularity on the lower part of the face to approximate the characteristics of facial skin. The result is a spatially varying BRDF that is described at each point by a sum of the generalized cosine lobes of Lafortune et al. [10].

5.1 Constructing the albedo map

We measured the albedo map, which must describe the spatially varying reflectance due to diffuse reflection, using a sequence of digital photographs of the face taken under controlled illumination. (See [14] for a more detailed description of a similar procedure.) The laboratory setup for taking the photographs is shown in Figure 3. The subject wears a hat printed with machine-recognizable targets to track head pose, and the camera stays stationary while the subject rotates. The only illumination comes from light sources at measured locations near the camera, and a black backdrop is used to reduce indirect reflections from spilled light. The lens and light sources are covered by perpendicular polarizers so that specular reflections are suppressed, leaving only the diffuse component in the images.

Since we know the camera and light source locations, we can use standard ray tracing techniques to compute the surface normal, the irradiance, the viewing direction, and the corresponding coordinates in texture space for each pixel in each image. Under the assumption that we are observing ideal Lambertian reflection, we can compute the Lambertian reflectance for a particular point in texture space from this information. Repeating this computation for every pixel in one photograph amounts to projecting the image into texture space and dividing by the computed irradiance due to the light sources to obtain a map of the diffuse reflectance across the surface (Figure 4, top row). In practice the projection is carried out by reverse mapping, with the outer loop iterating through all the pixels in the texture map, and stochastic supersampling is used to average over the area in the image that projects to a particular texture pixel.

The albedo map from a single photograph only covers part of the surface, and the results are best at less grazing angles, so we take a weighted average of all the individual maps to create a single albedo map for the entire face. The weighting function (Figure 4, second row) should give higher weights to pixels that are viewed and/or illuminated from directions nearly normal to the surface, and it should drop to zero well

Fig. 4. Building the albedo map. Top to bottom: two camera views of one subject projected to texture space; the associated weight maps; the merged albedo maps for two subjects; the albedo maps cleaned up for rendering.

before either viewing or illumination becomes extremely grazing. We chose the function $(\cos\theta_i \cos\theta_e - c)^p$ where θ_i and θ_e are the incident and exitant angles, and we use the values $c = 0.2$ and $p = 4$.

Before computing the albedo for a particular texture pixel, we verify that the pixel is visible and suitably illuminated. We trace multiple rays from points on the pixel to points on the light source and to the camera point, and mark the pixel as having zero, partial, or full visibility and illumination.[4] We only compute albedo for pixels that are fully visible, fully illuminated by at least one light source, and not partially illuminated by any light source. This ensures that partially occluded pixels and pixels that are in full-shadow or penumbra regions are not used.

Some calibration is required to make these measurements meaningful. We calibrated the camera's transfer curve using the method of Debevec and Malik [4]; we calibrated the light-camera system's flat-field response using a photograph of a large white card; and we calibrated the lens's focal length and distortion using the technique of Zhang [24]. We set the absolute scale factor using a reference sample of known reflectance. When image-to-image variation in light source intensity was a consideration, we controlled for that by including the reference sample in every image.

The texture maps that result from this process do a good job of automatically capturing the detailed variation in color across the face. In a few areas, however, the system cannot compute a reasonable result; also, the strap used to hold the calibration hat in place is visible. We remove these problems using an image editing tool, filling in blank areas with nearby texture or with uniform color. The bottom two rows of Figure 4 show the raw and edited albedo maps for comparison.

The areas where the albedo map does not provide reasonable results are where the surface is not observed well enough (e. g., under the chin) or is too intricately shaped to be correctly scanned and registered with the images (e. g., the ears). Neither of these types of areas requires the texture from the albedo map for realistic appearance—the first because they are not prominently visible and the second because the geometry provides visual detail—so this editing has relatively little effect on the appearance of the final renderings.

6 Results

Figure 5 shows several different aspects of the face model, using still frames from the accompanying video. In the top row the face is shown from several angles to demonstrate that the albedo map and measured BRDF realistically capture the distinctive appearance of the skin and its color variation over the entire face, viewed from any angle. In the second row the effects of rim and side lighting are shown, including strong specular reflections at grazing angles. Note that the light source has the same intensity and is at the same distance from the face for all three images; it is the directional variation in reflectance that leads to the familiar lighting effects seen in the renderings. In the bottom row expression deformations are applied to the face to demonstrate that the face still looks natural under normal expression movement.

7 Conclusions

We have described and demonstrated a system that addresses the challenge of modeling and rendering faces to the high standard of realism that must be met before an image as

[4] It is prudent to err on the large side when estimating the size of the light source.

Fig. 5. Renderings of the face model in different orientations, under different lighting, and with different expressions.

familiar as a human face can appear believable. Our philosophy is to use measurements whenever we can so that the face model actually resembles a real face. The geometry of the face is represented by a displacement-mapped subdivision surface that has consistent connectivity and correspondence across different faces, which provides basis functions and a parameterization that form a strong foundation for future work. The reflectance comes from previous BRDF measurements of human skin together with new measurements that combine several views into a single illumination-corrected texture map for diffuse reflectance. The motion comes from a previously described motion capture technique and is applied to the face model using an improved deformation method that produces motions suitable for shaded surfaces.

While many issues remain to be addressed before our model can pass a visual Turing test under close examination, our results demonstrate the value of using measured data in rendering and animating faces. The realism of the renderings is greatly enhanced by using the geometry, motion, and reflectance of real faces in a physically-based renderer.

7.1 Future work

One avenue of future work is to add more features to the face to increase realism: teeth and ears are good candidates. A more difficult but equally important problem is to add hair (including eyebrows, eyelashes, and facial hair if it is present). Another feature that would contribute to realism is the characteristic wrinkles that form in the face as it moves. Since our motion capture data does not resolve these wrinkles, they must be predicted from the face motion and added to the surface. Our consistent parameterization provides a way to position these features automatically on different faces.

An important way to broaden the applicability of our animation technique is to reuse motion data. Since it is inconvenient to do motion capture for each script that needs to be performed for each face, we are investigating ways to transform motion data recorded from one person's face so that it can be used for other people's faces; the correspondence provided by our generic face model is crucial to this effort. It would be even more useful to be able to assemble motion from existing examples for scripts containing arbitrary new speech and expressions, thus avoiding the need to do any new motion capture except for very unusual motions.

The algorithm for deforming the mesh is essentially one of sparse data interpolation, and new ways of solving this problem that use the parameterization and basis functions provided by the subdivision representation are yet another promising future direction.

7.2 Acknowledgements

Thanks to Peter Shirley for the ray tracer Eon, to Hugues Hoppe for many valuable discussions, and to the Cornell Program of Computer Graphics for the skin BRDF data.

References

1. Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Computer Graphics (SIGGRAPH 1999 Proceedings)*, pages 187–194, 1999.
2. Justine Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Computer Graphics (SIGGRAPH 1994 Proceedings)*, pages 413–420, August 1994.
3. R. L. Cook, T. Porter, and L. Carpenter. Distribution ray tracing. In *Computer Graphics (SIGGRAPH 1984 Proceedings)*, pages 165–174, July 1984.

4. Paul Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Computer Graphics (SIGGRAPH 1997 Proceedings)*, pages 369–378, August 1997.
5. Douglas DeCarlo and Dimitris Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings CVPR*, pages 231–238, 1996.
6. I. Essa and A. Pentland. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, 1997.
7. Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, 1995.
8. Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making faces. In *Computer Graphics (SIGGRAPH 1998 Proceedings)*, pages 55–67, July 1998.
9. Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Computer Graphics (SIGGRAPH 1994 Proceedings)*, pages 295–302, July 1994.
10. Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Nonlinear approximation of reflectance functions. In *Computer Graphics (SIGGRAPH 1997 Proceedings)*, pages 117–126, August 1997.
11. Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *Computer Graphics (SIGGRAPH 2000 Proceedings)*, July 2000. (Forthcoming).
12. Yuencheng Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Computer Graphics (SIGGRAPH 1995 Proceedings)*, pages 55–62, July 1995.
13. Charles Loop. *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, University of Utah, August 1987.
14. Stephen R. Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, August 1998.
15. Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. Image-based BRDF measurement including human skin. In *Rendering Techniques '99 (Proceedings of the Eurographics Workshop on Rendering)*, pages 131–144, June 1999.
16. Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *Computer Graphics (SIGGRAPH 1998 Proceedings)*, pages 75–84, July 1998.
17. C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Multiple textures stitching and blending on 3D objects. In *Rendering Techniques '99 (Proceedings of the Eurographics Workshop on Rendering)*, pages 119–130, June 1999.
18. Holly Rushmeier and Fausto Bernardini. Computing consistent normals and colors from photometric data. In *Proceedings of the Second International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, October 1999.
19. Holly Rushmeier, Fausto Bernardini, Joshua Mittleman, and Gabriel Taubin. Acquiring input for rendering at appropriate level of detail: Digitizing a Pietà. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop on Rendering)*, pages 81–92, June 1998.
20. Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *Transactions on Graphics*, 15(1):1–36, 1996.
21. Keith Waters. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (SIGGRAPH 1987 Proceedings)*, pages 17–24, July 1987.
22. Lance Williams. Performance-driven facial animation. In *Computer Graphics (SIGGRAPH 1990 Proceedings)*, pages 235–242, August 1990.
23. Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Computer Graphics (SIGGRAPH 1999 Proceedings)*, pages 215–224, August 1999.
24. Zhengyou Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, 1998.