Modeling and Animating Realistic Faces from Images

Frédéric Pighin Richard Szeliski[‡] David H. Salesin [† ‡]
University of Southern California [†]University of Washington [‡]Microsoft Research

Abstract

We present a new set of techniques for modeling and animating realistic faces from photographs and videos. Given a set of face photographs taken simultaneously, our modeling technique allows the interactive recovery of a textured 3D face model. By repeating this process for several facial expressions, we acquire a set of faces models that can be linearly combined to express a wide range of expressions. Given a video sequence, this linear face model can be used to estimate the face position, orientation, and facial expression at each frame. We illustrate these techniques on several datasets and demonstrate robust estimations of detailed face geometry and motion.

1 Introduction

Realistic face synthesis is one of the most fundamental problems in computer graphics — and one of the most difficult. Indeed, attempts to model and animate realistic human faces date back to the early 70's (Parke 1972), with many dozens of research papers published since. The applications of face animation include such diverse fields as character animation for films and advertising, computer games (Inc 1993), video teleconferencing (Choi, Kiyoharu, Harashima and Takebe 1994), user-interface agents and avatars (Thórisson 1997), and facial surgery planning (Koch, Gross, Carls, von Büren, Fankhauser and Parish 1996, Vannier, Marsh and Warren 1983). So far, no perfectly realistic face animation has ever been generated by computer: no "face animation Turing test" has ever been passed.

There are several factors that make realistic face animation so elusive. First, the human face is an extremely complex geometric form. For example, the human face models used in Pixar's *Toy Story* had several thousand control points each (E. Ostby, Pixar Animation Studios 1997). Moreover, the face exhibits countless tiny creases and wrinkles, as well as subtle variations in color and texture — all of which are crucial for our comprehension and appreciation of facial expressions. Rendering the face is a challenging task too: the multiple skin layers absorb and scatter incoming light in very complex ways. As difficult as the face is to model and render, it is even more problematic to animate, since face movement is a product of the underlying skeletal and muscular forms, as well as the mechanical properties of the skin and sub-cutaneous layers (which vary in thickness and composition in different parts of the face). All of these problems are enormously magnified by the fact that we as humans have an uncanny ability to read expressions — an ability that is not merely a learned skill, but part of our deep-rooted instincts. For facial expressions, the slightest deviation from truth is something any person will immediately detect.

Analyzing face images is the inverse problem of face images synthesis. Given a face image or stream of images, the goal of the analysis is to extract information such as the face position and orientation, the facial expression, or the identity of the person. These problems are the focus of many research efforts in computer vision. This interest is easily explained by the multiple applications that rely on algorithms for face analysis. For instance, in telesurveillance, human faces must be first located in an image stream and then tracked over time; eventually the face can be matched to a database to identify the person. In the field of human computer interfaces, researchers strive to develop algorithms that would allow a computer to recognize our facial expressions or track our gaze and react accordingly. In teleconferencing, dramatic compression rates can be achieved by interpreting face images and reducing them to a few parameters that can be sent efficiently over a communication network. Just as for the synthesis problem, the complexity of the human face makes the analysis of face images a daunting task. This difficulty

comes from the multiple appearances that the human face can take as the facial expression, the lighting conditions, and the identity of the person change. A robust technique must take into account all these sources of variations.

2 Methodology

The synthesis and the analysis of face images are dual problems. On the one hand, the goal is to generate realistic face images; on the other hand the goal is to extract information from face images. However, we believe some of the most powerful approaches to solve these problems combine both synthesis and analysis.

Face images of real persons (photographs) provide a wealth of information that can be leveraged for modeling and animating realistic synthetic faces. New images can be generated from these sample images using *image-based modeling and rendering* techniques. The goal of these techniques is to reconstruct a continuous image function using a set of sample images. Thus the original samples can be interpolated or extrapolated to produce novel images. For the case of face images, these novel images can introduce changes in viewpoint, in the lighting conditions, or the facial expression.

The analysis of face images is usually done using a model of the face. Such a model provides structured knowledge about the type of information that is sought from the images. For instance, a model for identifying individuals from mug shots could consist of an annotated database of face photographs. The analysis results from a comparison between the information provided by the model and the target image. A particular approach is to use an *analysis-by-synthesis* technique where the model is able to produce images that can be directly compared to the input images. This approach permits to take advantage of a realistic face model and estimates parameters that can be reused in an animation system.

In this paper, we develop techniques to model and animate 3-dimensional face models from images, using these image-based approaches. We build face models from a set of photographs of a person's face. We also develop a technique to estimate the position, orientation, and expression of the face in a video sequence. These techniques blur the line between computer graphics and computer vision as we explore the fertile synergy between these two fields.

3 Related work

Realistic face modeling and animation have been the topics of many research efforts stemming both in the computer graphics and computer vision communities. In this section, we describe various research projects pertaining to face modeling, face animation, and face tracking. We also position our approach among these efforts.

3.1 Modeling

Several methods can be used to capture a 3D face model of a real person's face. First, the face geometry and texture can be automatically acquired using laser-based cylindrical scanners such as those produced by Cyberware (Cyb 1990). A second approach is based on photogrammetric techniques in which images are used to create precise geometry (Moffitt and Mikhail 1980, Slama 1980). Our modeling technique falls into this category. Compared with face modeling methods that utilize a laser scanner, our technique uses simpler acquisition equipment (regular cameras), and it is capable of extracting texture maps of higher resolution. (Cyberware scans typically produce a cylindrical grid of 512 by 256 samples). The price we pay for these advantages is the need for user intervention in the modeling process.

The earliest photogrammetric techniques applied to face modeling and animation employed grids that were drawn directly on the human subject's face (Parke 1972, Parke 1974). One consequence of these grids, however, is that the images used to construct geometry can no longer be used as valid texture maps for the subject. More recently, several methods have been proposed for modeling the face photogrammetrically without the use of grids (Ip and Yin 1996, Kurihara and Arai 1991). These modeling methods are similar in concept to the modeling technique described in this article. However, these previous techniques use a small predetermined set of features to deform the generic face mesh to

the particular face being modeled, and offer no mechanism to further improve the fit. Such an approach may perform poorly on faces with unusual features or other significant deviations from the normal. Our system, by contrast, gives the user complete freedom in specifying the correspondences, and enables the user to refine the initial fit as needed. Another advantage of our technique is its ability to handle fairly arbitrary camera positions and lenses, rather than using a fixed pair that are precisely oriented. Our method is similar, in concept, to the work done in architectural modeling by Debevec *et al.* (Debevec, Taylor and Malik 1996), where a set of annotated photographs are used to model buildings starting from a rough description of their shape.

Computer vision techniques can be used to extract geometry from images. These techniques can rely on stereo images (Devernay and Faugeras 1994), shading (Leclerc and Bobick 1991), or silhouettes (Proesman, Gool and Oosterlinck 1996). Approaches more specific to faces include the work by Lengagne *et al.* (Lengagne, Fua and Monga 1998). Starting from a set of photographs, they extract a depth map of the face using a stereo algorithm. Using a minimization technique, a 3D face model is then deformed to fit the map. Fua and Miccio (Fua and Miccio 1998) developed a more general model fitting scheme that incorporates multiple sources of information (stereo, shading, silhouette, and features). Each source defines a set of constraints to the minimization problem.

The degree of variations between human faces is such that developing robust analysis methods is very difficult and fully automatic techniques seldom produce perfect results. We believe that the user should be able to steer the modeling process to reach her objective. The role of the modeling tools is to assist the user in reaching her goal in a fast and painless way. This "power assist" (Kass, Witkin and Terzopoulos 1987) approach has been successfully applied to architecture modeling (Debevec et al. 1996) and image interpretation (Kass et al. 1987). The modeling tool we develop follows this approach: the user specifies a set of facial features between the images; the modeling tool assists in finding these features and provides a visualization of the resulting extracted model. The recovered model can be compared to the photographs to evaluate the goodness of the fit.

The skin has very complex reflectance properties that are difficult to reproduce. Indeed, the only attempt at modeling realistically the reflectance properties of the skin was done by Hanrahan and Kruegger (Hanrahan and Kruger 1993). Using a physically-based model of the skin and a subsurface scattering model they simulate the skin complex reflectance properties. This model is unfortunately very complex and the results are not very convincing. A different approach is to use face images taken under different illumination conditions. Hallinan (Hallinan 1994) takes such an image-based approach by modeling lighting variations in face images using eigenfaces (Turk and Pentland 1987). The model is built by including images of faces shot under a set of controlled lighting conditions. The study shows that the variations in lighting can be accounted for by a subspace of low dimension (about 5) for a simple Lambertian model with specular lobes. While the reflectance properties of the human face are far more complex than the proposed models, this might be enough for certain applications. Debevec *et al.* (Debevec, Hawkins, Tchou, Duiker, Sarokin and Sagar 2000) offers a thorough study of skin's reflectance. Thanks to a complex aparatus (the *lightstage*), they manage to capture a dense set of images of a face lighted from different directions. They reuse these samples to illuminate the face under arbitrary lighting conditions. They also develop a limited skin reflectance model and fit it to the reflectance samples. In this paper, we do not address this issue extensively. We render faces with the lighting conditions captured when the photographs were taken. We explore two methods for texturing our 3D face model. A view-independent method that uses a cylindrical texture map and a view-dependent method that blends between different photographs as a function of the virtual viewpoint. The view-dependent technique provides crisper results, in particular for specular highlights.

3.2 Model-based tracking

Recovering the face position and the facial expression automatically from a video is a difficult problem. The difficulty comes from the wide range of appearances that can be generated by the human face under different orientations, illumination conditions, and facial expressions. The creation of a model that encompasses these variations and supports the robust estimation of the face parameters is thus a very challenging task.

This problem can be made easier using markers on the face such as heavy makeup (Terzopoulos and

Waters 1993) or a set of colored dots stuck onto the actor's face (Guenter, Grimm, Wood, Malvar and Pighin 1998, Williams 1990). Once the position of the markers has been determined, the position of the face and the facial features can easily be derived. Williams (Williams 1990) pioneered this technique, tracking 2D points on a single image. Guenter *et al.* (Guenter et al. 1998) extended this approach to tracking points in 3D using multiple images.

The use of markers on the face limits the type of video that can be processed; instead, computer vision techniques are not used to extract information from a video of an unmarked face. These tracking techniques can be classified into two groups depending on whether they use 2-dimensional or 3-dimensional models.

Two-dimensional facial feature trackers can be based on deformable or rigid patches (Black and Yacoob 1995, Covell 1996, Hager and Belhumeur 1996), or on edge or feature detectors (Blake and Isard 1998, Lanitis, Taylor and Cootes 1995, Matsino, Lee, Kimura and Tsuji 1995). Using a 2D model limits the range of face orientations that can be handled. For large variations of face orientation, the 3-dimensional properties of the face need to be modeled.

Different approaches have been taken to fit a 3D face model to an image or a sequence of images. One approach is to build a physically-based model that describes the muscular structure of the face. These models are fitted by computing which set of muscle activations best corresponds to the target image. Terzopoulos and Waters (Terzopoulos and Waters 1993) estimated muscle contractions from the displacement of a set of face markers. Other techniques compute an optical flow from the image sequence and decompose the flow into muscle activations. Essa and Pentland (Essa and Pentland 1997) built a physically-based face model and developed a control-theoretic technique to fit it to a sequence of images. Decarlo and Metaxas (Decarlo and Metaxas 1998) employed a similar model that incorporates possible variations in head shape using anthropometric measurements. Comparable techniques (Choi, Aizawa, Harshima and Takebe 1994, Li, Roivainen and Forchheimer 1993) are used in model-based image coding schemes where the model parameters provide a compact representation of the video frames.

These approaches use a 3D articulated face model to derive a model for face motion. Using the estimated motion from an input video, the 3D model can be animated to synthesize an animation similar to the input sequence. However, the synthesized images are not leveraged to perform the analysis. More recent techniques employ an *analysis-by-synthesis* approach where target face images are analyzed by comparing them to synthetic face images. La Cascia *et al.* (Cascia, Isidoro and Sclaroff 1998) model the face with a texture-mapped cylinder. The textures are extracted by building a mosaic from the input image sequence. Schodl *et al.* (Schodl, Ario and Essa 1998) use a more detailed geometric model to estimate the face position and orientation. These two models however do not permit the estimation of the facial expression or the identity of the face in the target image. Vetter and Blanz (Vetter and Blanz 1998) built a more general statistical model of the human face by processing a set of example faces that includes variations in identity and expression. Their model consists of a linear combination of scanned 3D face models. They assume that the head pose is known and compute which linear combination of the sample faces best fit the target image.

Vetter and Blanz's linear combination approach is similar to ours. Our morphing technique however is different. Their model separates shape and texture information. We believe these quantities are correlated and we consider expression vectors that include both shape and texture. Our goal too is different; we do not estimate the identity of the person but rather track the person's facial expression and head pose. The fact that our model is specialized for a given person enables better tracking results, and our realistic face models permit an analysis-by-synthesis approach.

In this paper, we present a model-based face tracking technique. Our technique consists of fitting a 3D face model to each frame in the input video. This face model is a linear combination of 3D face models, each corresponding to a particular facial expression. The use of realistic face models allows us to match realistic renderings of faces to the target images. Moreover, we directly recover parameters that can be used in an animation system. To fit the model, we minimize an error function over the set of facial expressions and face positions spanned by the model. The fitting process employs a continuous optimization technique. Because our 3D head model generates more realistic images than previous face animation systems, we can directly use an analysis-by-synthesis approach and get better tracking results than with other methods.

4 Overview of paper

The remainder of this paper is broken into three sections.

Section 5 describes our face modeling technique. Given a set of photographs of a person's face, a generic 3D face model is fitted to estimate both the face geometry and the camera parameters. Using this geometric data, a texture map can be extracted from the photographs.

Section 6 presents a tracking technique that uses our realistic face models in an analysis-by-synthesis approach. Given a video featuring a person, we recover for each frame the position of the head, its orientation, and the facial expression. Our face model is a linear combination of textured face meshes each representing a key expression. Our fitting method uses continuous optimization techniques to estimate the parameters of the face model.

Sections 7 and 8 concludes this paper with a summary of our work and some suggestions for future research directions.

The material that appears in this paper has been presented from a computer graphics perspective in two publications. The first one was published in the Proceedings of the ACM SIGGRAPH Conference in 1998 (Pighin, Hecker, Lischinski, Szeliski and Salesin 1998) and covers Section 5. It also describes a keyframe animation system based on a collection of face meshes. The second one was published in the Proceedings of the International Conference on Computer Vision in 1999 (Pighin, Szeliski and Salesin 1999) and covers Section 6 .

5 Modeling faces from photographs

Creating realistic face models is a very challenging problem. The human face is an extremely complex geometric form. Moreover, the face exhibits countless tiny creases and wrinkles, as well as subtle variations in color and texture — all of which are crucial for our comprehension and appreciation of facial expressions. One way to ensure some level of realism is to base the model on a real person's face. In this section we present a technique to model a person's face from a set of photographs corresponding to different views of the face.

Our modeling approach is based on photogrammetric techniques in which images are used to create precise geometry and texture information. We model faces by interactively fitting a 3D generic face model to a set of images. Our fitting process consists of several basic steps. First, we capture multiple views of a human subject (with a given facial expression) using cameras at arbitrary locations. Next, we digitize these photographs and manually mark a small set of initial corresponding points on the face in the different views (typically, corners of the eyes and mouth, tip of the nose, etc.). These points are then used to automatically recover the camera parameters (position, focal length, etc.) corresponding to each photograph, as well as the 3D positions of the marked points in space. The 3D positions are then used to deform a generic 3D face mesh to fit the face of the particular human subject. At this stage, additional corresponding points may be marked to refine the fit. Finally, we extract one or more texture maps for the 3D model from the photos. Either a single view-independent texture map can be extracted, or the original images can be used to perform view-dependent texture mapping.

The rest of this section is structured as follows. First, we describe in detail the technique we develop to recover the face geometry from a set of images. Then we explain how texture information is extracted from the photos once the geometry has been estimated. Finally, Section 5.3 illustrates the application of our modeling technique to several sets of face images.

5.1 Model fitting

The task of the model-fitting process is to adapt a generic face model to fit an individual's face and facial expression. As input to this process, we take several images of the face from different viewpoints (Figure 1a) and a generic face model (we use the generic face model created with Alias|Wavefront (Ali 1995) shown in Figure 1c). The generic face model provides information about the general topology of the human face and an initial estimate of the face geometry. The photographs provide geometric and texture information about a specific face. Fitting the model is done by manually specifying a set of correspondences between the face model and feature points on the photos. First, a few features points are

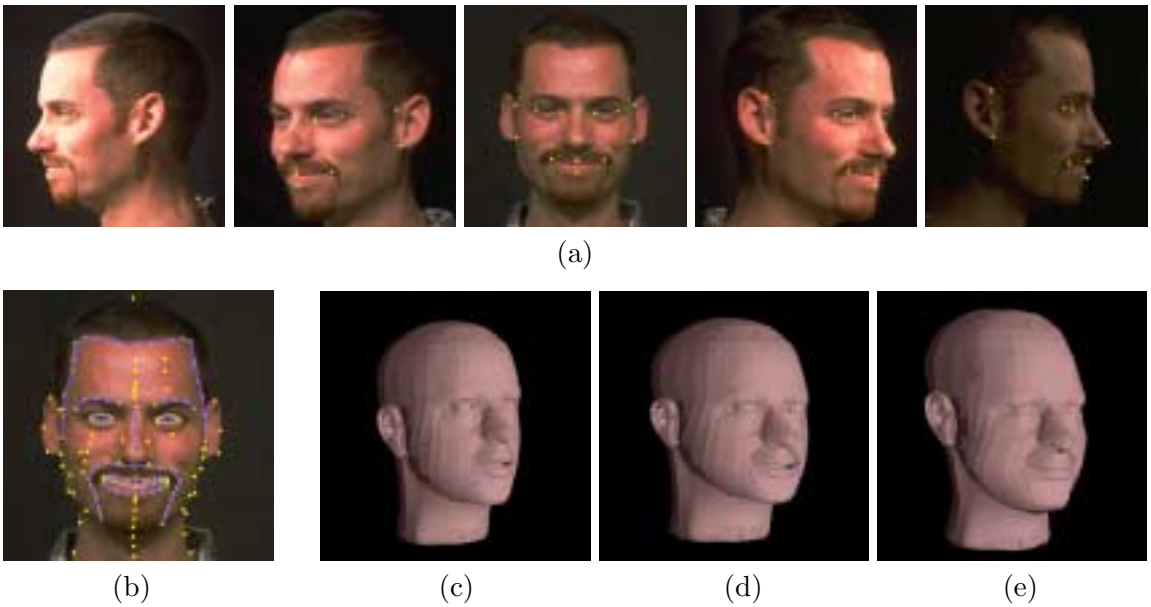


Figure 1: **Model-fitting process.** (a) a set of input images with marked feature points, (b) facial features annotated using a set of curves, (c) generic face geometry (shaded surface rendering), (d) face adapted to initial 13 feature points (after pose estimation) (e) face after 99 additional correspondences have been given.

chosen (13 in this case, shown in the frames of Figure 1a) to recover the camera pose. These same points are also used to refine the generic face model (Figure 1d). The model can be further refined by drawing corresponding curves in the different views (Figure 1b). The output of the process is a face model that has been adapted to fit the face in the input images (Figure 1e), along with a precise estimate of the camera pose corresponding to each input image.

The model-fitting process consists of three stages. In the *pose recovery* stage, we apply computer vision techniques to estimate the viewing parameters (position, orientation, and focal length) for each of the input cameras. We simultaneously recover the 3D coordinates of a set of *feature points* on the face. These feature points are selected interactively from among the face mesh vertices, and their positions in each image (where visible) are specified by hand. The *scattered data interpolation* stage uses the estimated 3D coordinates of the feature points to compute the positions of the remaining face mesh vertices. In the *shape refinement* stage, we specify additional correspondences between face vertices and image coordinates to improve the estimated shape of the face (while keeping the camera pose fixed). These three stages are described in detail in the rest of this chapter.

5.1.1 Pose recovery

The goal of pose recovery is to estimate, for each photograph, the internal parameters (focal length) and external parameters (position and orientation) of the camera. This problem is solved using an iterative minimization technique: starting with a rough knowledge of the camera positions (e.g., frontal view, side view, etc.) and of the 3D shape (given by the generic head model), we iteratively improve the pose and the 3D shape estimates in order to minimize the difference between the predicted and observed feature point positions. Our formulation is based on the non-linear least squares structure-from-motion algorithm introduced by Szeliski and Kang (Szeliski and Kang 1994). However, unlike the method they describe, which uses the Levenberg-Marquardt algorithm to perform a complete iterative minimization over all of the unknowns simultaneously, we break the problem down into a series of linear least squares problems that can be solved using very simple and numerically stable techniques (Golub and Van Loan 1996, Press, Flannery, Teukolsky and Vetterling 1992).

To formulate the pose recovery problem, we associate a rotation matrix $\boldsymbol{R}^k$ and a translation vector $\boldsymbol{t}^k$ with each camera pose k. (The three rows of $\boldsymbol{R}^k$ are $\boldsymbol{r}_x^k$, $\boldsymbol{r}_y^k$, and $\boldsymbol{r}_z^k$, and the three entries in $\boldsymbol{t}^k$ are t_x^k, t_y^k, t_z^k.) We write each 3D feature point as $\boldsymbol{m}_i$, and its 2D screen coordinates in the k-th image as (x_i^k, y_i^k).

Assuming that the origin of the (x, y) image coordinate system lies at the optical center of each image (i.e., where the optical axis intersects the image plane), the traditional 3D projection equation for a camera with a focal length f^k (expressed in pixels) can be written as

$$x_i^k = f^k \frac{\boldsymbol{r}_x^k \cdot \boldsymbol{m}_i + t_x^k}{\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i + t_z^k} \qquad y_i^k = f^k \frac{\boldsymbol{r}_y^k \cdot \boldsymbol{m}_i + t_y^k}{\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i + t_z^k} \tag{1}$$

(This is just an explicit rewriting of the traditional projection equation $\boldsymbol{x}_i^k \quad \boldsymbol{R}^k \boldsymbol{m}_i + \boldsymbol{t}^k$ where $\boldsymbol{x}_i^k = (x_i^k, y_i^k, f^k)$.)

There is a *scale ambiguity* in this structure from motion problem, i.e., if we scale all of the $\boldsymbol{m}$'s and $\boldsymbol{t}$'s up by the same amount, the answer is the same. There is also a pose ambiguity (we can rotate and/or translate all points and cameras by inverse amounts). In practice, we get around these difficulties by giving some initial camera position guesses and point position guesses. Since we iterate downhill in energy, there is no reason for the answer to be affected by the ambiguity.

Instead of using (1) directly, we reformulate the problem to estimate inverse distances to the object (Szeliski and Kang 1994). Let $\eta^k = 1/t_z^k$ be this inverse distance and $s^k = f^k \eta^k$ be a world-to-image scale factor. The advantage of this formulation is that the scale factor s^k can be reliably estimated even when the focal length is long, whereas the original formulation has a strong coupling between the f^k and t_z^k parameters.

Performing these substitutions, we obtain

$$\begin{aligned} x_i^k &= s^k \frac{\boldsymbol{r}_x^k \cdot \boldsymbol{m}_i + t_x^k}{1 + \eta^k \boldsymbol{r}_z^k \cdot \boldsymbol{m}_i} \\ y_i^k &= s^k \frac{\boldsymbol{r}_y^k \cdot \boldsymbol{m}_i + t_y^k}{1 + \eta^k \boldsymbol{r}_z^k \cdot \boldsymbol{m}_i}. \end{aligned}$$

If we let $w_i^k = (1 + \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i))^{\ -1}$ be the inverse denominator, and collect terms on the left-hand side, we get

$$\begin{aligned} w_i^k \quad x_i^k + x_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i) \quad s^k (\boldsymbol{r}_x^k \cdot \boldsymbol{m}_i + t_x^k) &= 0 \\ w_i^k \quad y_i^k + y_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i) \quad s^k (\boldsymbol{r}_y^k \cdot \boldsymbol{m}_i + t_y^k) &= 0 \end{aligned} \tag{2}$$

Note that these equations are linear in each of the unknowns that we wish to recover, i.e., $\boldsymbol{m}_i$, t_x^k, t_y^k, η^k, s^k, and $\boldsymbol{R}^k$, if we ignore the variation of w_i^k with respect to these parameters. When solving these equations, we consider that the scalars w_i^k are constant and update their values as better estimates of $\boldsymbol{m}_i$, t_x^k, t_y^k, η^k, s^k, and $\boldsymbol{R}^k$ become available. Using this method, the set of equations 2 defines a weighted linear least squares problem.

Given estimates for initial values, we can solve for different subsets of the unknowns. In our current algorithm, we solve for the unknowns in five steps: first s^k, then $\boldsymbol{m}_i$, $\boldsymbol{R}^k$, t_x^k and t_y^k, and finally η^k. This order is chosen to provide maximum numerical stability given the crude initial pose and shape estimates. For each parameter or set of parameters chosen, we solve for the unknowns using linear least squares. More details about the systems solved are given in Appendix A. The simplicity of this approach is a result of solving for the unknowns in five separate stages, so that the parameters for a given camera or 3D point can be recovered independently of the other parameters.

5.1.2 Scattered data interpolation

Once we have computed an initial set of coordinates for the feature points $\boldsymbol{m}_i$, we use these values to deform the remaining vertices on the face mesh. We construct a smooth interpolation function that gives the 3D displacements between the original point positions and the new adapted positions for every vertex

in the original generic face mesh. Constructing such an interpolation function is a standard problem in scattered data interpolation. Given a set of known displacements $\boldsymbol{u}_i = \boldsymbol{m}_i - \boldsymbol{m}_i^{(0)}$ away from the original positions $\boldsymbol{m}_i^{(0)}$ at every constrained vertex i, construct a function that gives the displacement $\boldsymbol{u}_j$ for every unconstrained vertex j.

There are several considerations in choosing the particular data interpolant (Nielson 1993). The first consideration is the embedding space, that is, the domain of the function being computed. In our case, we use the original 3D coordinates of the points as the domain. (An alternative would be to use some 2D parameterization of the surface mesh, for instance, the cylindrical coordinates described in Section 5.2.) We therefore attempt to find a smooth vector-valued function $\boldsymbol{f}(\boldsymbol{m})$ fitted to the known data $\boldsymbol{u}_i = \boldsymbol{f}(\boldsymbol{m}_i)$, from which we can compute $\boldsymbol{u}_j = \boldsymbol{f}(\boldsymbol{m}_j)$.

There are also several choices for how to construct the interpolating function (Nielson 1993). We use a method based on *radial basis functions*, that is, functions of the form

$$\boldsymbol{f}(\boldsymbol{m}) = \sum_i \boldsymbol{c}_i \phi(\|\boldsymbol{m} - \boldsymbol{m}_i\|),$$

where $\phi(r)$ are radially symmetric basis functions. A more general form of this interpolant also adds some low-order polynomial terms to model global, e.g., affine, deformations (Lee, Chwa, Shin and Wolberg 1995, Lee, Wolberg, Chwa and Shin 1996, Nielson 1993). In our system, we use an affine basis as part of our interpolation algorithm, so that our interpolant has the form:

$$\boldsymbol{f}(\boldsymbol{m}) = \sum_i \boldsymbol{c}_i \phi(\|\boldsymbol{m} - \boldsymbol{m}_i\|) + \boldsymbol{M}\boldsymbol{m} + \boldsymbol{t}, \tag{3}$$

To determine the coefficients $\boldsymbol{c}_i$ and the affine components $\boldsymbol{M}$ and $\boldsymbol{t}$, we solve a set of linear equations that includes the interpolation constraints $\boldsymbol{u}_i = \boldsymbol{f}(\boldsymbol{m}_i)$, as well as the constraints $\sum_i \boldsymbol{c}_i = 0$ and $\sum_i \boldsymbol{c}_i \boldsymbol{m}_i{}^{\mathrm{T}} = 0$, which remove affine contributions from the radial basis functions.

Many different functions for $\phi(r)$ have been proposed (Nielson 1993). After experimenting with a number of functions, we have chosen to use an exponential function $\phi(r) = e^{-r/64}$, with units measured in inches.

Figure 1d shows the shape of the face model after having interpolated the set of computed 3D displacements at 13 feature points shown in Figure 1 and applied them to the entire face.

5.1.3 Correspondence-based shape refinement

After warping the generic face model into its new shape, we can further improve the shape by specifying additional correspondences. Since these correspondences may not be as easy to locate correctly, we do not use them to update the camera pose estimates. Instead, we simply solve for the values of the new feature points $\boldsymbol{m}_i$ using a simple least-squares fit, which corresponds to finding the point nearest the intersection of the viewing rays in 3D. We can then re-run the scattered data interpolation algorithm to update the vertices for which no correspondences are given. This process can be repeated until we are satisfied with the shape.

Figure 1e shows the shape of the face model after 99 additional correspondences have been specified. To facilitate the annotation process, we grouped vertices into polylines. Each polyline corresponds to an easily identifiable facial feature such as the eyebrow, eyelid, lips, chin, or hairline. The features can be annotated by outlining them with hand-drawn curves on each photograph where they are visible. The curves are automatically converted into a set of feature points by stepping along them using an arc-length parameterization. Corresponding curves on different views are parameterized independently. An alternative would be to parameterize one of the curves, chosen as a reference, and use the epipolar geometry to parameterize the other curves. For each point on the reference curve, the epipolar geometry associates a line on the other views. Intersecting these lines with the curve provides a parameterization. Figure 1b shows annotated facial features using a set of curves on the front view.

5.2 Texture extraction

In this section, we describe the process of extracting the texture maps necessary for rendering photorealistic images of a reconstructed face model from various viewpoints.

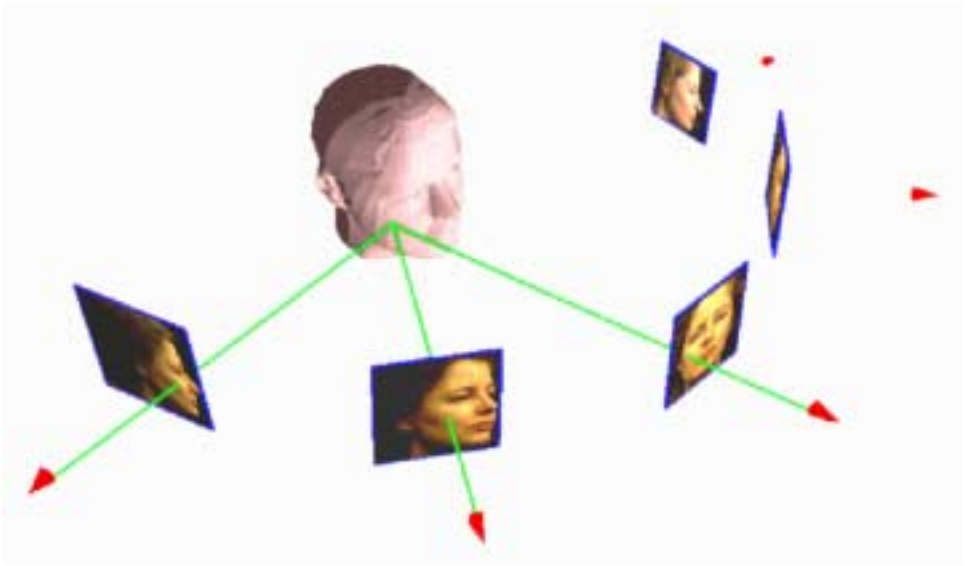


Figure 2: **Sampling the photographs.** The texture value at a point on the model's surface is determined by sampling the photographs where this point is visible.

The texture extraction problem can be defined as follows. Given a collection of photographs, the recovered viewing parameters, and the fitted face model, compute for each point $\boldsymbol{m}$ on the face model its texture color $T(\boldsymbol{m})$.

Each point $\boldsymbol{m}$ may be visible in one or more photographs; therefore, we must identify the corresponding point in each photograph and decide how these potentially different values should be combined (blended) together. The photographs are sampled by projecting $\boldsymbol{m}$ using the recovered camera parameters. Figure 2 illustrates this process. There are two principal ways to blend values from different photographs: *view-independent blending*, resulting in a texture map that can be used to render the face from any viewpoint; and *view-dependent blending*, which adjusts the blending weights at each point based on the direction of the current viewpoint (Debevec et al. 1996, Pulli, Cohen, Duchamp, Hoppe, Shapiro and Stuetzle 1997). Rendering takes longer with view-dependent blending, but the resulting image is of slightly higher quality (see Figure 5).

5.2.1 Weight maps

As outlined above, the texture value $T(\boldsymbol{m})$ at each point on the face model can be expressed as a convex combination of the corresponding colors in the photographs:

$$T(\boldsymbol{m}) = \frac{\sum_k g^k(\boldsymbol{m})\, C^k(x^k, y^k)}{\sum_k g^k(\boldsymbol{m})}.$$

Here, C^k is the image function (color at each pixel of the k-th photograph,) and (x^k, y^k) are the image coordinates of the projection of $\boldsymbol{m}$ onto the k-th image plane. The *weight map* $g^k(\boldsymbol{m})$ is a function that specifies the contribution of the k-th photograph to the texture at each face surface point.

The construction of these weight maps is probably the trickiest and the most interesting component of our texture extraction technique. There are several important considerations that must be taken into account when defining a weight map:

1. *Self-occlusion:* $g^k(\boldsymbol{m})$ should be zero unless $\boldsymbol{m}$ is front-facing with respect to the k-th image and visible in it.
2. *Smoothness:* the weight map should vary smoothly, in order to ensure a seamless blend between different input images.

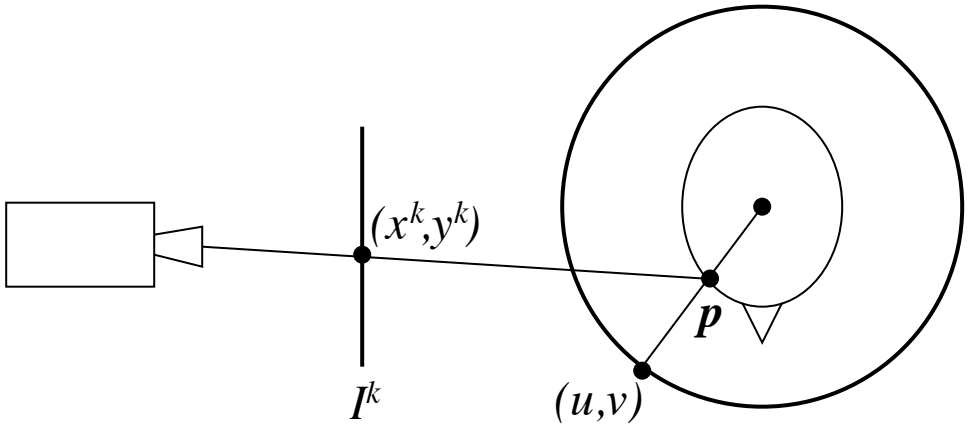


Figure 3: **Geometry for texture extraction.**

3. *Positional certainty:* $g^k(\boldsymbol{m})$ should depend on the "positional certainty" (Kurihara and Arai 1991) of $\boldsymbol{m}$ with respect to the k-th image. The positional certainty is defined as the dot product between the surface normal at $\boldsymbol{m}$ and the k-th direction of projection.
4. *View similarity:* for view-dependent texture mapping, the weight $g^k(\boldsymbol{m})$ should also depend on the angle between the direction of projection of $\boldsymbol{m}$ onto the j-th image and its direction of projection in the new view.

Previous authors have taken only a subset of these considerations into account when designing their weighting functions. For example, Kurihara and Arai (Kurihara and Arai 1991) use positional certainty as their weighting function, but they do not account for self-occlusion. Akimoto *et al.* (Akimoto, Suenaga and Wallace 1993) and Ip and Yin (Ip and Yin 1996) blend the images smoothly, but address neither self-occlusion nor positional certainty. Debevec *et al.* (Debevec et al. 1996), who describe a view-dependent texture mapping technique for modeling and rendering buildings from photographs, do address occlusion but do not account for positional certainty. (It should be noted, however, that positional certainty is less critical in photographs of buildings, since most buildings do not tend to curve away from the camera.)

To facilitate fast visibility testing of points on the surface of the face from a particular camera pose, we first render the face model using the recovered viewing parameters and save the resulting depth map from the Z-buffer. Then, with the aid of this depth map, we can quickly classify the visibility of each face point by applying the viewing transformation and comparing the resulting depth to the corresponding value in the depth map.

5.2.2 View-independent texture mapping

In order to support rapid display of the textured face model from any viewpoint, it is desirable to blend the individual photographs together into a single texture map. This texture map is constructed on a virtual cylinder enclosing the face model. The mapping between the 3D coordinates on the face mesh and the 2D texture space is defined using a cylindrical projection, as in several previous papers (Chen, State and Banks 1995, Kurihara and Arai 1991, Lee, Terzopoulos and Waters 1995).

For view-independent texture mapping, we will index the weight map g^k by the (u, v) coordinates of the texture being created. Each weight $g^k(u, v)$ is determined by the following steps:

1. Construct a feathered visibility map F^k for each image k. These maps are defined in the same cylindrical coordinates as the texture map. We initially set $F^k(u, v)$ to 1 if the corresponding face point $\boldsymbol{m}$ is visible in the k-th image, and to 0 otherwise. The result is a binary visibility map, which is then smoothly ramped (feathered) from 1 to 0 in the vicinity of the boundaries (Szeliski and Shum 1997). A cubic polynomial is used as the ramping function.
2. Compute the 3D point $\boldsymbol{m}$ on the surface of the face mesh whose cylindrical projection is (u, v) (see Figure 3). This computation is performed by casting a ray from (u, v) on the cylinder towards the

(a)

(b)

(c)

Figure 4: **Example of cylindrical texture.** A cylindrical texture (part (a)) has been extracted from the images shown in Figure 1. Part (b) and part (c) shows the projection of the front view and its associated weight map respectively.

Figure 5: **Comparison between view-independent and view-dependent texture mapping.** The view-independent rendering is shown on the left and the view-dependent rendering on the right. Higher frequency details are visible in the view-dependent rendering.

cylinder's axis. The first intersection between this ray and the face mesh is the point $\boldsymbol{m}$. (Note that there can be more than one intersection for certain regions of the face, most notably the ears. These special cases are discussed in Section 5.2.4.) Let $P^k(\boldsymbol{m})$ be the positional certainty of $\boldsymbol{m}$ with respect to the k-th image.

3. Set weight $g^k(u, v)$ to the product $F^k(u, v)\, P^k(\boldsymbol{m})$.

For view-independent texture mapping, we will compute each pixel of the resulting texture $T(u, v)$ as a weighted sum of the original image functions, indexed by (u, v).

Figure 4 illustrates the extraction of a cylindrical texture map from the photographs in Figure 1a.

Certain areas of the head (e.g., top or back of the head) might not be visible in any of the photographs. This results in areas or holes in the texture map that cannot be colored. However, we might need a texture map that covers the whole 3D face model. We used a simple algorithm to fill these uncolored areas. Our hole filling algorithm iteratively fills the holes in the texture map by expanding the filled area by one pixel until all the pixels in the texture have been colored. The extrapolated pixels are colored by averaging nearby filled pixels. An alternative is to construct an image pyramid of the texture (Gortler, Grzeszczuk, Szeliski and Cohen 1996). The pyramid is processed in a first phase by filling in the missing pixels at the boundary starting from the highest resolution and moving up in the pyramid. In a second phase, the filled pixels are propagated down the pyramid to fill the missing color information.

5.2.3 View-dependent texture mapping

The main disadvantage of the view-independent cylindrical texture map described above is that its construction involves blending together resampled versions of the original images of the face. Because of this resampling, and also because of slight registration errors, the resulting texture is slightly blurry. This problem can be alleviated to a large degree using a view-dependent texture map (Debevec et al. 1996) in which the blending weights are adjusted dynamically, according to the current view.

For view-dependent texture mapping, we render the model several times, each time using a different input photograph as a texture map, and blend the results. More specifically, for each input photograph, we associate texture coordinates and a blending weight with each vertex in the face mesh. (The rendering hardware performs perspective-correct texture mapping along with linear interpolation of the blending weights.)

Given a viewing direction $\boldsymbol{d}$, we first select the subset of photographs used for the rendering and

then assign blending weights to each of these photographs. (Pulli *et al.* (Pulli et al. 1997) select three photographs based on a Delaunay triangulation of a sphere surrounding the object.) Since our cameras were positioned roughly in the same plane, we select just the two photographs whose view directions $\boldsymbol{d}$ and $\boldsymbol{d}^{+1}$ are the closest to $\boldsymbol{d}$ and blend between the two.

In choosing the view-dependent term $V^k(\boldsymbol{d})$ of the blending weights, we wish to use just a single photo if that photo's view direction matches the current view direction precisely, and to blend smoothly between the nearest two photos otherwise. We used the simplest possible blending function having this effect:

$$V^k(\boldsymbol{d}) \quad = \quad \begin{array}{ll} \boldsymbol{d} \cdot \boldsymbol{d}^k \quad \boldsymbol{d} \cdot \boldsymbol{d}^{+1} & \text{if } \ell \quad k \quad \ell+1 \\ 0 & \text{otherwise} \end{array}$$

For the final blending weights $g^k(\boldsymbol{m}, \boldsymbol{d})$, we then use the product of all three terms $F^k(x^k, y^k)\, P^k(\boldsymbol{m})\, V^k(\boldsymbol{d})$.

View-dependent texture maps have several advantages over cylindrical texture maps. First, they can make up for some lack of detail in the model. Second, whenever the model projects onto a cylinder with overlap, a cylindrical texture map will not contain data for some parts of the model. This problem does not arise with view-dependent texture maps if the geometry of the mesh matches the photograph properly. One disadvantage of the view-dependent approach is its higher memory requirements and slower speed due to the multi-pass rendering. Another drawback is that the resulting images are much more sensitive to any variations in exposure or lighting conditions in the original photographs.

5.2.4 Eyes, teeth, ears, and hair

The parts of the mesh that correspond to the eyes, teeth, ears, and hair are textured in a separate process. The eyes and teeth are usually partially occluded by the face; hence it is difficult to extract a texture map for these parts in every facial expression. The ears have an intricate geometry with many folds and usually fail to project without overlap on a cylinder. The hair has fine-detailed texture that is difficult to register properly across facial expressions. For these reasons, each of these face elements is assigned an individual texture map. The texture maps for the eyes, teeth, and ears are computed by projecting the corresponding mesh part onto a selected input image where that part is clearly visible (the front view for eyes and teeth, side views for ears). To keep the hair texture crisp, we render it using a constant geometry and a single texture map. This geometry and texture are taken from a predetermined facial expression.

The eyes and the teeth are usually partially shadowed by the eyelids and the mouth respectively. We approximate this shadowing by modulating the brightness of the eye and teeth texture maps according to the size of the eyelid and mouth openings. The modulating functions are quadratic polynomials of the size of the openings. The coefficients of the functions were chosen by fitting this model to the actual photographs used during modeling.

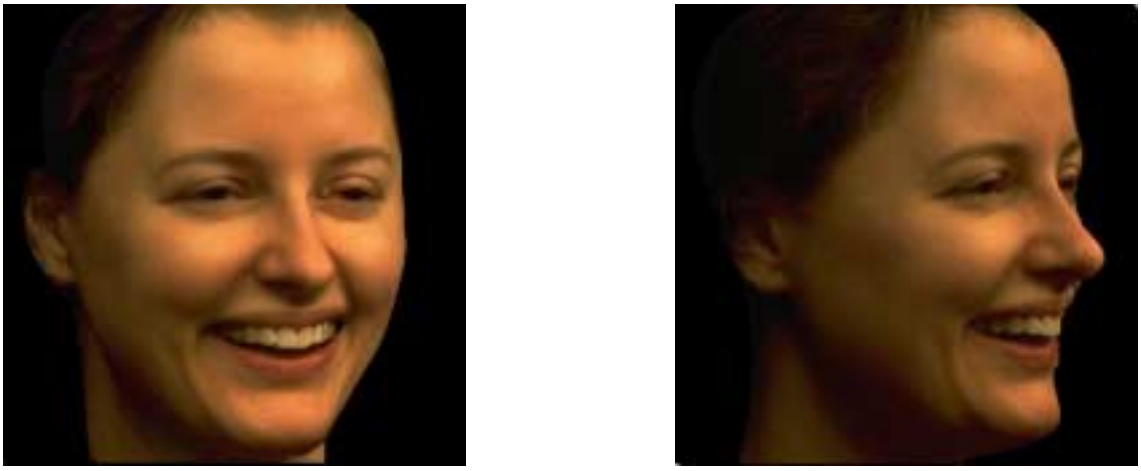
5.3 Modeling results

In this section, we present several results that illustrate our modeling and rendering techniques.

In order to test our technique, we photographed both a man (J.R.) and a woman (Karla) in a variety of facial expressions. The photography was performed using five cameras simultaneously. The cameras were not calibrated in any particular way, and the lenses had different focal lengths. Since no special attempt was made to illuminate the subject uniformly, the resulting photographs exhibited considerable variation in both hue and brightness. The photographs were digitized using the Kodak PhotoCD process. Five typical images (cropped to the size of the subject's head) are shown in Figure 1a.

Figures 6 and 7 show both the input images (on the top) and synthetic reproductions using view-dependent texture mapping (at the bottom). Note that the hair of the female model is not faithfully reproduced. The geometry of the hair is very intricate and cannot be accurately modeled using our generic face model.

Input photographs



Rendered model

Figure 6: **Modeling a joy expression (Karla).**

Input photographs

Rendered model

Figure 7: **Modeling a happy expression (J.R.).**

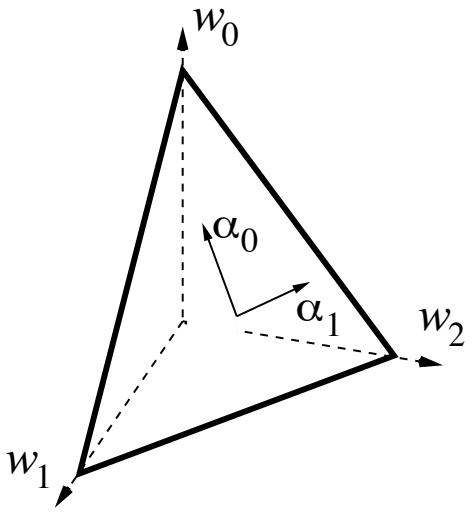


Figure 8: **Relationship between blending weights and expression parameters.** This example uses 3 basic expressions. The expression parameters represent a plane in the 3-dimensional expression space.

6 Model-based face tracking

In this section, we present a novel face tracking technique that uses a realistic 3-dimensional face model. The face model is a linear combination of a set of textured face meshes each corresponding to a facial expression. The fitting process follows an analysis-by-synthesis approach, where renderings of the realistic face model are compared to the target video frames.

The rest of this section is organized as follows. We first start by describing the face model and its parameterization. Then, we describe how the model is fitted to a video sequence and show some tracking results.

6.1 Model fitting

In this section, we describe how the face model is parameterized. We also discuss in detail the optimization technique used to fit the model to a sequence of images.

6.1.1 3D face model

The model we propose to fit to the video is a linear combination of 3D texture-mapped models, each corresponding to a particular basic facial expression. Examples of basic expressions are joy, anger, sadness, surprise, disgust, pain. By locally varying the percentage of each basic expression in different parts of the face, a wide range of expressions can be generated.

Our model represents a full human head, but only the facial mask is parameterized, while the rest of the face (e.g., neck and hair) is constant. The face is parameterized by a vector of parameters $\boldsymbol{p} = p_1, ..., p_n$. These parameters form two subsets: the position parameters and the expression parameters.

The position parameters include a translation $\boldsymbol{t}$, which indicates the position of the center of the head, and a rotation $\boldsymbol{R}$, which indicates its orientation.

We represent facial expressions in a vector space, where the dimensions are a set of expressions called basic expressions. A facial expression is a linear combination of basic expressions. Such combination is determined by a set of blending weights $w_1, ..., w_m$, one for each basic expression. We constrain these weights to sum to 1; hence all the vectors of blending weights belong to the subspace $_k\, w_k = 1$. Since this subspace has $m \quad 1$ dimensions, parameterizing expressions with these blending weights introduces redundancy in our model. We use an alternative parameterization to reduce the number of parameters that needs to be estimated. We parameterize the subspace by selecting an orthonormal basis of $m \quad 1$

vectors and a point in the subspace. These vectors provides a new parameterization of facial expressions that we call expression parameters and write as $\alpha = (\alpha_1, ..., \alpha_{m\ 1})$. These two parameterizations are used for different purposes: the blending weights specifie the linear combination of basic expressions and are used for rendering the face model, the expression parameters provide a more compact representation of facial expressions and are used for analysis. The mapping between the vector of blending weights $\boldsymbol{w}$ and the vector of expression parameters α can be expressed as the following transformation:

$$\boldsymbol{w} = \boldsymbol{Q}\alpha + \boldsymbol{w} \tag{4}$$

where $\boldsymbol{Q}$ is a matrix whose column vectors span the hyperplane $_k\, w_k = 0$, and $\boldsymbol{w}$ is a vector belonging to $_k\, w_k = 1$. Figure 8 illustrates the relationship between the blending weights and the expression parameters. In practice, we choose $\boldsymbol{w}$ to be a vector whose coordinates are all equal to $\frac{1}{n}$ which defines the mean of the basic expressions. We explore different ways of defining the matrix Q that determines the expression parameters. Each expression parameter represents a direction in the linear space of expressions spanned by the basic expressions $E_1, ..., E_m$. A natural set of directions is the differences between a particular expression, for instance E_1, and the rest of the expressions: $E_2 \quad E_1, ..., E_m \quad E_1$. This parameterization does not take into account possible similarities between expressions and could result in directions that are correlated. Choosing a set of orthogonal directions can improve the robustness of the face parameters estimation. We used an orthogonalization technique (Gramm-Schmidt) to generate an orthonormal basis. Because the sets of basic expressions we chose were small and contained fairly distinct facial expressions, using an orthogonal basis improved only slightly the estimation of the parameters. For a larger set of basic expressions, applying a principal component analysis to the basic expressions might produce a small set of parameters for the principal uncorrelated directions.

To span a wider range of facial expressions, the face is split into several regions that can each be controlled independently. We thus use a set of expression parameters $\{\alpha_l\}$ for each region. The partition we use in our experiments is shown in figure 9.

Rendering the model is done by linearly combining the basic expressions using 3D morphing. A consensus geometry is computed by linearly interpolating the basic expression geometries using the blending weights w_k. The model is rendered multiple times, once for each basic expression. These renderings $\hat{\boldsymbol{I}}_k$ are then blended together using the weights w_k to produce the final image $\hat{\boldsymbol{I}}$:

$$\hat{\boldsymbol{I}} = \sum_k w_k \hat{\boldsymbol{I}}_k \tag{5}$$

We use hardware rendering to render our face model. The blending weights are used to specify the alpha channel of the images $\hat{\boldsymbol{I}}_k$. Gouraud shading then interpolates these values smoothly across each triangle. To render correctly an expression blend that uses weights outside of the range [0..1] of alpha values allowed by the hardware, we scale and shift the weights values to map the range of weights [0.5..1.5] into the alpha range [0..1]. During the final software compositing the alpha values read from the frame buffer are transformed back into the original blending weights. We do not allow extrapolation outside of the range [0.5..1.5] to avoid generating unrealistic face blends.

6.1.2 Optimization procedure

Fitting a linear combination of faces to an image has already been explored by different research groups. Edwards *et al.* (Edwards, Taylor and Cootes 1998) developed a model built out of registered face images. In their work, a principal component analysis is applied to the set of faces to extract a set of parameters. The dependencies between the parameters and the images generated by the model are approximated by a linear function. This approximation is then used in an iterative optimization technique, using a variant of the steepest descent algorithm. Jones and Poggio (Jones and Poggio 1998) constructed a similar 2D model and use a more sophisticated fitting technique. They use a stochastic gradient method that estimates the derivatives by sampling a small number of pixels. The technique we develop uses second-order derivatives of the error function. We also investigate both analytical and finite-difference computations of the partial derivatives.

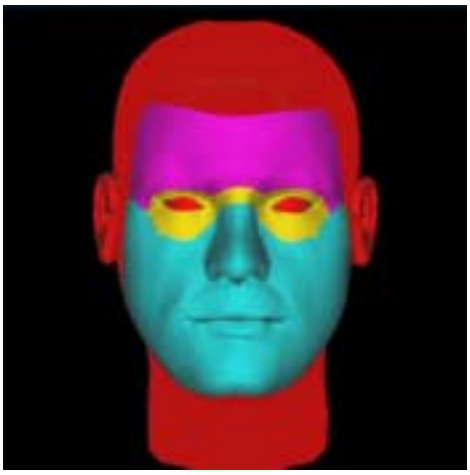


Figure 9: **Partition of the face.** The different regions are rendered with different colors. The parts of the face that are constant are rendered in red.

To fit our model to a target face image $\boldsymbol{I}_t$, we develop an optimization method whose goal is to compute the model parameters $\boldsymbol{p}$ yielding a rendering of the model $\hat{\boldsymbol{I}}(\boldsymbol{p})$ that best resembles the target image. An error function $\epsilon(\boldsymbol{p})$ is used to evaluate the discrepancy between $\boldsymbol{I}_t$ and $\hat{\boldsymbol{I}}(\boldsymbol{p})$:

$$\epsilon(\boldsymbol{p}) = \frac{1}{2} \ \boldsymbol{I}_t \quad \hat{\boldsymbol{I}}(\boldsymbol{p}) \ ^2 + D(\boldsymbol{p}) = \frac{1}{2} \quad _j \ [\boldsymbol{I}_t(x_j, y_j) \quad \hat{\boldsymbol{I}}(\boldsymbol{p})(x_j, y_j)]^2 + D(\boldsymbol{p})$$

where (x_j, y_j) corresponds to a location of a pixel on the image plane and $D(\boldsymbol{p})$ is a penalty function that forces each blending weight to remain close to the interval [0..1]. The penalty limits the range of facial expressions and helps avoid unrealistic faces. The function $D(\boldsymbol{p})$ is a piecewise-quadratic function of the following form:

$$D(\boldsymbol{p}) = c \quad _k \ [\min(0, w_k)^2 + \max(0, w_k \quad 1)^2]$$

where c is a constant. This function is a sum of two terms: the first one limits the range of negative values that the parameters can take, the second one limits the range of values above 1.

Fitting is done in several phases that optimize different sets of parameters. We start by estimating the position parameters and then independently estimate the expression parameters in each region. Treating these parameters independently allows us to apply different optimization methods to them. All the parameters are estimated using variants of the Levenberg-Marquardt (LM) (Press et al. 1992) algorithm to minimize the function ϵ.

The LM algorithm is a continuous optimization technique based on a second-order Taylor expansion of ϵ. This iterative algorithm starts from an initial estimate of the parameters and computes at each iteration a new set of values that reduces ϵ further. The parameters are updated by taking a step $\delta\boldsymbol{p}$ at each iteration. This step is given by the expression:

$$[\boldsymbol{J}(\boldsymbol{p})^T \boldsymbol{J}(\boldsymbol{p}) + \lambda \boldsymbol{I} + H(\boldsymbol{p})]\delta\boldsymbol{p} = \quad \boldsymbol{J}(\boldsymbol{p})^T [\boldsymbol{I}_t \quad \hat{\boldsymbol{I}}(\boldsymbol{p})] \qquad D(\boldsymbol{p}) \tag{6}$$

where $\boldsymbol{I}$ is the identity matrix, $J(\boldsymbol{p})$ is the Jacobian of $\hat{\boldsymbol{I}}(\boldsymbol{p})$, $\quad D(\boldsymbol{p})$ and $H(\boldsymbol{p})$ are the gradient and Hessian of $D(\boldsymbol{p})$, respectively, and λ is a parameter used to tune the optimization.

6.1.3 Computing the Jacobian

We explore two ways of computing the Jacobian $\boldsymbol{J}(\boldsymbol{p})$. The first one uses finite differences and requires sampling the error function multiple times at each iteration. The second one uses an analytical expression

of the Jacobian and does not require evaluating ϵ. On the one hand, the analytical Jacobian can be more efficient than the finite-difference approximation if evaluating the error function is expensive. On the other hand, using finite differences may produce better results if the error function is not smooth enough. In our case, evaluating the error function is fairly time consuming because it involves rendering the 3D face model. In our experiments, both methods produced very similar results. We preferred the analytical method for its superior performances and used the numerical method to validate our computations.

Finite differences

We use central differences to approximate the value of the Jacobian. Each entry of the Jacobian matrix is computed using the following formula:

$$\frac{\partial \hat{\boldsymbol{I}}}{\partial p_i}(\boldsymbol{p}) \quad \frac{\hat{\boldsymbol{I}}(\boldsymbol{p} + \Delta p_i \boldsymbol{e}_i) \quad \hat{\boldsymbol{I}}(\boldsymbol{p} \quad \Delta p_i \boldsymbol{e}_i)}{2\Delta p_i}$$

where $\boldsymbol{e}_i$ is the **i**-th vector of the canonical basis.

Analytical Jacobian

An alternative way of computing the Jacobian is to derive an analytical expression. This requires examining the dependencies $\hat{\boldsymbol{I}}$ with respect to its parameters. Let us consider a variation of one of the face parameters p_i. As p_i varies, $\hat{\boldsymbol{I}}$ can change in two ways. First the position and the geometry of the face model can change, resulting in a displacement or flow in image space. This phenomenon induces a geometric term $\boldsymbol{G}$ in the Jacobian. Secondly, if p_i is an expression parameter, the way the images $\hat{\boldsymbol{I}}_k$ are composited varies. We will call this second term the blending term $\boldsymbol{B}$. Hence, the Jacobian $\boldsymbol{J}$ is a sum of two terms:

$$\boldsymbol{J} = \boldsymbol{G} + \boldsymbol{B}$$

or in terms of the columns of $\boldsymbol{J}$:

$$\frac{\partial \hat{\boldsymbol{I}}}{\partial p_i} = \boldsymbol{G}_{p_i} + \boldsymbol{B}_{p_i}$$

for a given parameter. Let us examine these two terms starting with $\boldsymbol{G}$. The optical flow due to the changes in the model position and geometry can be expressed using the chain rule:

$$\boldsymbol{G}_{p_i} = \frac{\partial \hat{\boldsymbol{I}}}{\partial x_j}\frac{\partial x_j}{\partial p_i} + \frac{\partial \hat{\boldsymbol{I}}}{\partial y_j}\frac{\partial y_j}{\partial p_i},$$

The vector $(\frac{\hat{\boldsymbol{I}}}{x_j}, \frac{\hat{\boldsymbol{I}}}{y_j})$ is the intensity gradient of image $\hat{\boldsymbol{I}}$ at pixel j. It is computed using a Sobel filter (Ballard and Brown 1982). The blending term B can be derived by differentiating equation 5:

$$\boldsymbol{B}_{p_i} = \quad \frac{\partial w_k}{\partial p_i}\hat{\boldsymbol{I}}_k$$
$$k$$

Bringing the two terms together, we obtain the following expression for the Jacobian:

$$\frac{\partial \hat{\boldsymbol{I}}}{\partial p_i} = \frac{\partial \hat{\boldsymbol{I}}}{\partial x_j}\frac{\partial x_j}{\partial p_i} + \frac{\partial \hat{\boldsymbol{I}}}{\partial y_j}\frac{\partial y_j}{\partial p_i} + \quad \frac{\partial w_k}{\partial p_i}\hat{\boldsymbol{I}}_k, \tag{7}$$
$$k$$

Each term in the previous equation is a function defined over the image plane and can be represented as an image. To compute the partial derivatives $\frac{x_j}{p_i}$ and $\frac{y_j}{p_i}$, we need to study the dependencies between the coordinates in the image plane (x_j, y_j) and the model's parameters. The 2D point (x_j, y_j) is obtained by projecting a 3D point, $\boldsymbol{m}_j$, onto the image plane. We can thus rewrite $(x_j, y_j) = P(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})$, where P is a projection whose optical axis is the z-axis. The transformation P has two components P_x and P_y, so that $x_j = P_x(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})$ and $y_j = P_y(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})$. Using the chain rule, we obtain:

Figure 10: **Tracking synthetic sequence.** The bottom row shows the result of fitting our model to the target synthetic images on the top row.

$$\frac{\partial x_j}{\partial p_i} = \frac{\partial P_x}{\partial \boldsymbol{m}_j}^T \frac{\partial \boldsymbol{m}_j}{\partial p_i}$$

$$\frac{\partial y_j}{\partial p_i} = \frac{\partial P_y}{\partial \boldsymbol{m}_j}^T \frac{\partial \boldsymbol{m}_j}{\partial p_i}$$

where $\boldsymbol{m}_j = \boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$. We detail the computation of the partial derivatives of P, $\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$ and w_k in Appendix A.

6.2 Tracking results

The experiments we ran divided the face into three regions: the mouth region, the eye region, and the forehead region as illustrated in Figure 9.

We tried different optimization variants and studied which ones were most appropriate for the different parameters. For the position parameters, the analytical Jacobian produced good results and allowed faster computations. For the expression parameters, the finite differences Jacobian produced better results than the analytical version. This difference can be explained in terms of the difference in behavior of the error function with respect to these two sets of parameters. The behavior of the error function with respect to the position parameters is fairly smooth, while its behavior with respect to the expression parameters is not.

The initial values of the parameters for the first frame are specified interactively. The estimated parameters are then used as initial values for the second frames. For subsequent frames, we used a simple prediction scheme to determine the initial values: the parameters are linearly extrapolated using the results from the two previous frames.

We also noticed that there were important color differences between the target images and the renderings of the model. These differences were introduced by the different optical systems used to capture the data: several film cameras were used to build the 3D model textures, and a camcorder was used to record the video footage. Our optimization technique would try to compensate for these color differences by adjusting the expression parameters. We reduced this problem using bandpass filtering on both the target images and the model renderings when estimating the expression parameters.

To test our fitting method, we generated a synthetic animation sequence using the face model employed for the analysis. Once the sequence has been fit, we can compare the ground truth values of

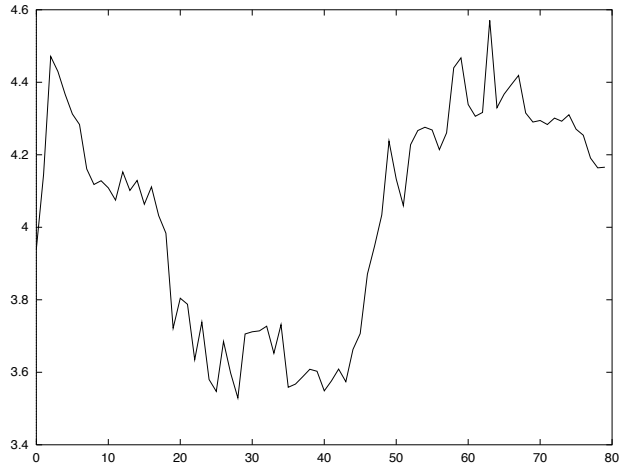


Figure 11: **Tracking synthetic sequence: RMS error.**

the parameters with the estimated values. The synthesized sequence uses three basic expressions: anger, neutral and joy. These expressions were blended uniformly over the whole face. The fitted model included the same basic expression. The initial guess was determined by hand for the position and set to the correct facial expression (neutral). Figure 10 offers a side by side comparison of target frames and rendered models. The expression weights are compared in Figure 12, Figure 13, and Figure 14 for the eye region, the mouth region and the forehead region respectively. The estimated values appear to be pretty far from the true values. The weights associated with the anger expression seems to be the hardest to estimate. However, the resulting rendered facial expressions are very close to the target expressions. This can be explained by the fact that many combinations of the basic expressions produce very similar renderings. The comparisons between position parameters are shown in Figures 15 and 16. For the position parameters, the estimated values are fairly close to the true values.

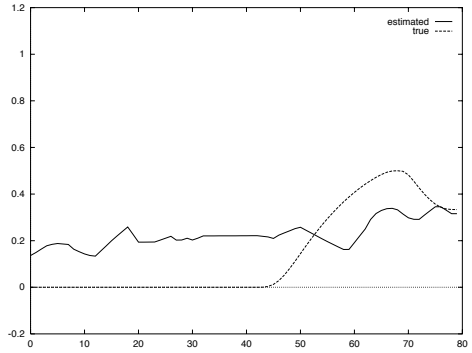
Figure 17 illustrates some tracking results using our technique on some input video. The model used 4 basic expressions: sleepy, joy, sulky, and neutral. Each frame took on the average 2 minutes to be fitted at a 300 by 300 resolution. Although this performance is very far from permitting real time tracking, all the applications we explore can be done as postprocessing on a video. A limitation of our approach is seen in the second column, where the shape of the left eyebrow in the fitted model does not match that in the target frame. This problem occurs because this particular eyebrow shape is not included in the linear space spanned by the set of basic expressions.

The plots in Figure 18 show the tracked parameter values as a function of frame number for the tracking sequence illustrated in Figure 17. Figure 19 illustrates the evolution of the root mean square error for the same sequence as a function of frame number.

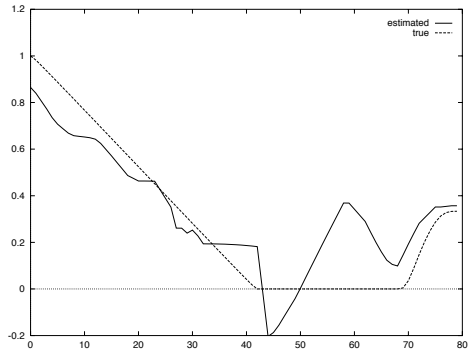
We tested the robustness of our technique by performing different tests with a synthetic animation sequence. We considered the sequence displayed in Figure 10 and performed some alterations on the frame. These alterations include scaling the colors and the adding noise. We performed further test using 4 or 5 expressions in the model instead of the 3 expressions that were used to generate the synthetic sequence. In all these tests, the model was fitted successfully to the video frames.

Our experience with this technique showed that the estimation of the face position and orientation is relatively easy compared to the estimation of the expression parameters. In particular, as the number of expression parameters (i.e., the number of blended expressions) grows, the technique is less and less reliable. In our examples, we had to choose a set of expressions that would be a good match for the input video. This limited the length of the video sequences that we could process.

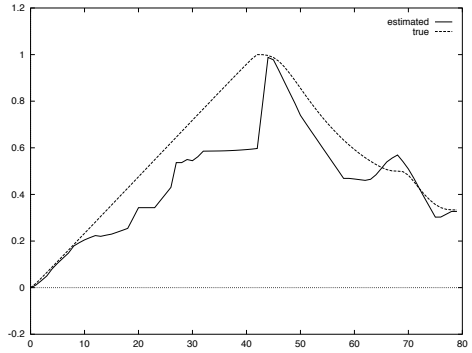
A set of mpeg files illustrating this tracking technique through several experiments can be found at the following url: http://grail.cs.washington.edu/projects/realface/tracking.html



Anger

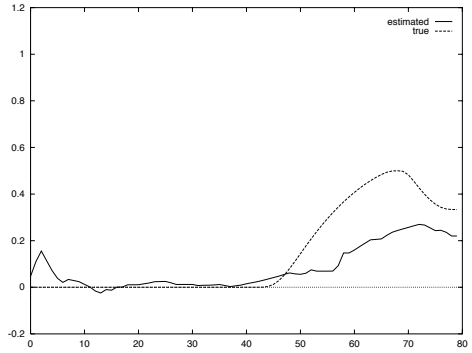


Neutral

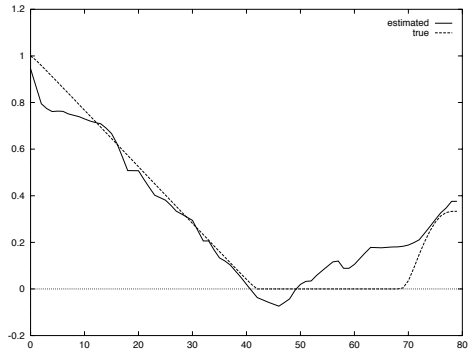


Joy

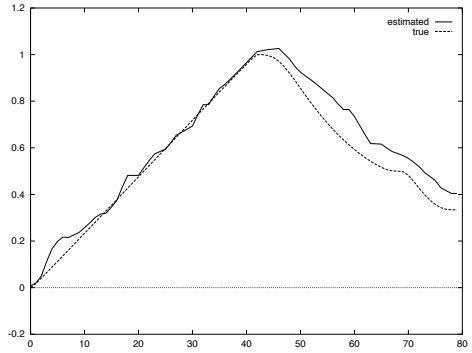
Figure 12: **Tracking synthetic sequence: eyes region.**



Anger

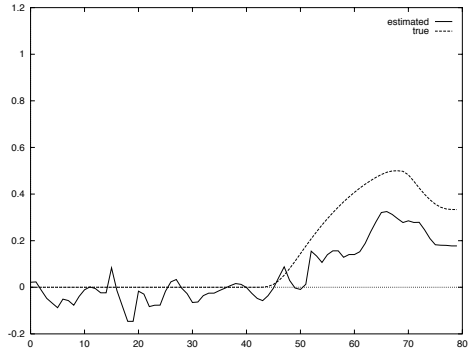


Neutral

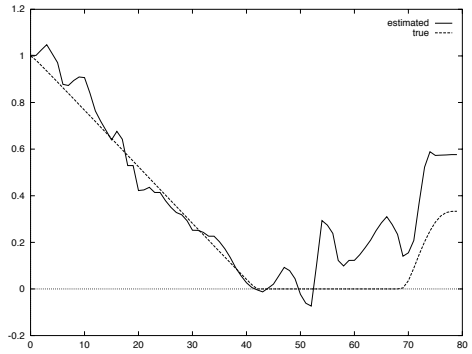


Joy

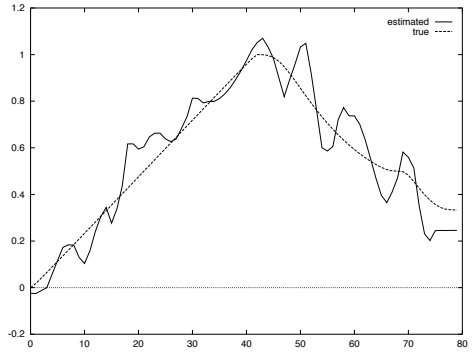
Figure 13: **Tracking synthetic sequence: mouth region.**



Anger



Neutral



Joy

Figure 14: **Tracking synthetic sequence: forehead region.**

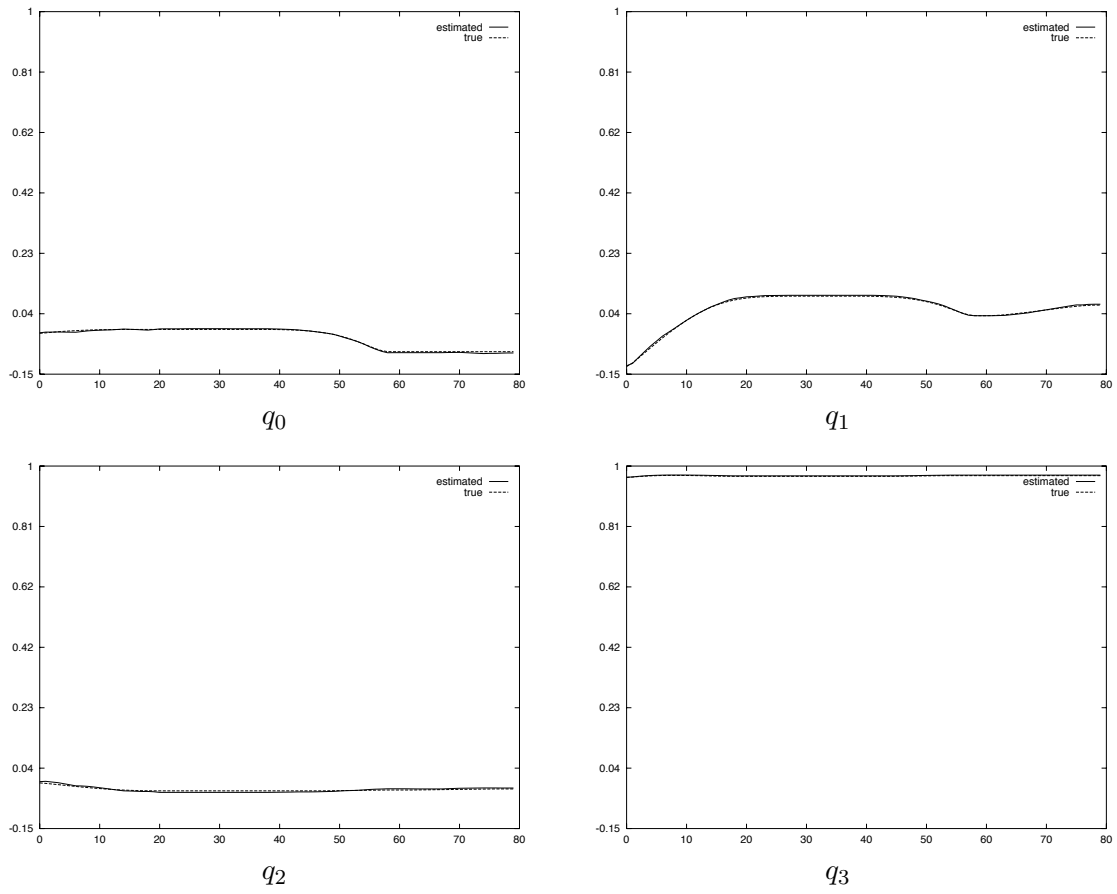


Figure 15: **Tracking synthetic sequence: rotation (represented as a quaternion).**

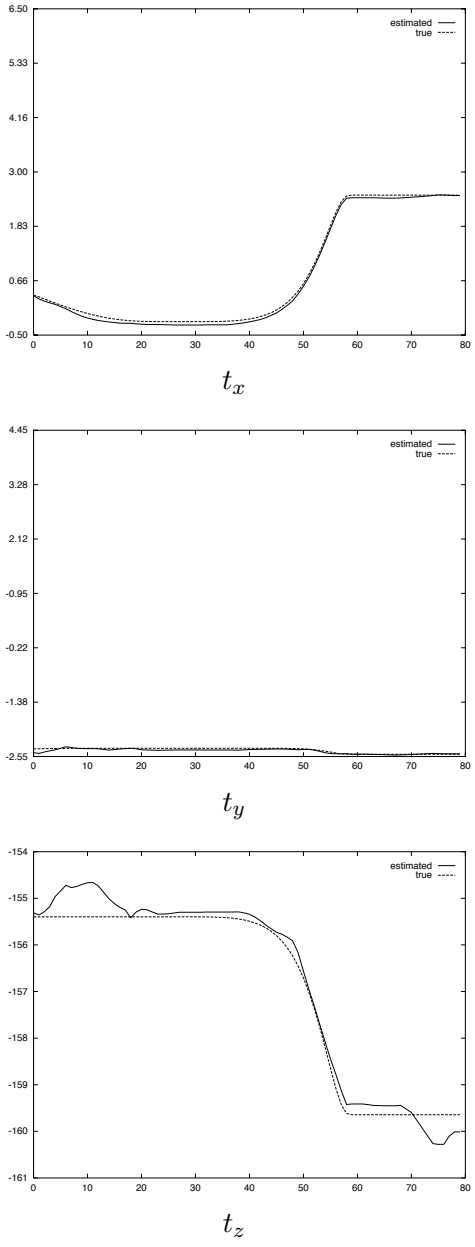


Figure 16: **Tracking synthetic sequence: translation.**

Figure 17: **Tracking example.** The bottom row shows the result of fitting our model to the target images on the top row.

7 Summary

In this paper, we explore the synthesis of realistic face animations from face images and videos.

We have presented a modeling technique to generate realistic texture-mapped 3-dimensional models from photographs. Our technique consists of fitting a generic face model to a set of photographs. The model is iteratively fitted by manually specifying a set of correspondences. We presented a simple minimization technique to recover the camera parameters for each photograph. Finally, we discussed the extraction of view-independent and view-dependent texture maps.

Compared to other face modeling technique such as 3D scanning our technique has the advantage of requiring less expensive capture equipment. Moreover, 3D scanners usually only provide fairly limited texture information (e.g., a 512 x 256 texture map for a Cyberware scanner). These advantages come at the expense of user intervention. The need for user intervention cannot be avoided until reliable automatic techniques are developed.

We have also introduced a novel technique for estimating the head's position and the facial expression from video footage. We use a realistic face model in an analysis-by-synthesis approach. Our model is a linear combination of 3D face models, which can be generated by photogrammetric techniques or 3D scanning. We use a continuous optimization technique to fit the model and give an analytical derivation of the computations. The quality of the model allows better tracking results than with other methods. Because it is based on an analysis-by-synthesis approach, this technique provides a parametrization of the face that can be used for both rendering and analyzing face images. This property could be leveraged in a model-based coding video system, such as MPEG-4, where a small set of Face Animation Parameters (FAPS) are transmitted to render a face image.

8 Future directions

In this paper, we presented techniques that we believe bring the state of the art in face animation a step further towards realism. There are however important future research areas that need to be explored in order to create a synthetic actor indistinguishable from a real person.

Automatic modeling. Our modeling technique requires the user to specify a set of annotations on face photographs. Automating this process would represent a major improvement. This is a challenging problem in particular for the areas of the face that have little texture information such as the cheeks, but recent results on 2D face modeling in computer vision (Lanitis et al. 1995) give us cause for hope. Blanz and Vetter (Blanz and Vetter 1999) propose a fully automatic technique using a model built from a linear combination of 3D texture-mapped models.

(a)

(b)

(c)

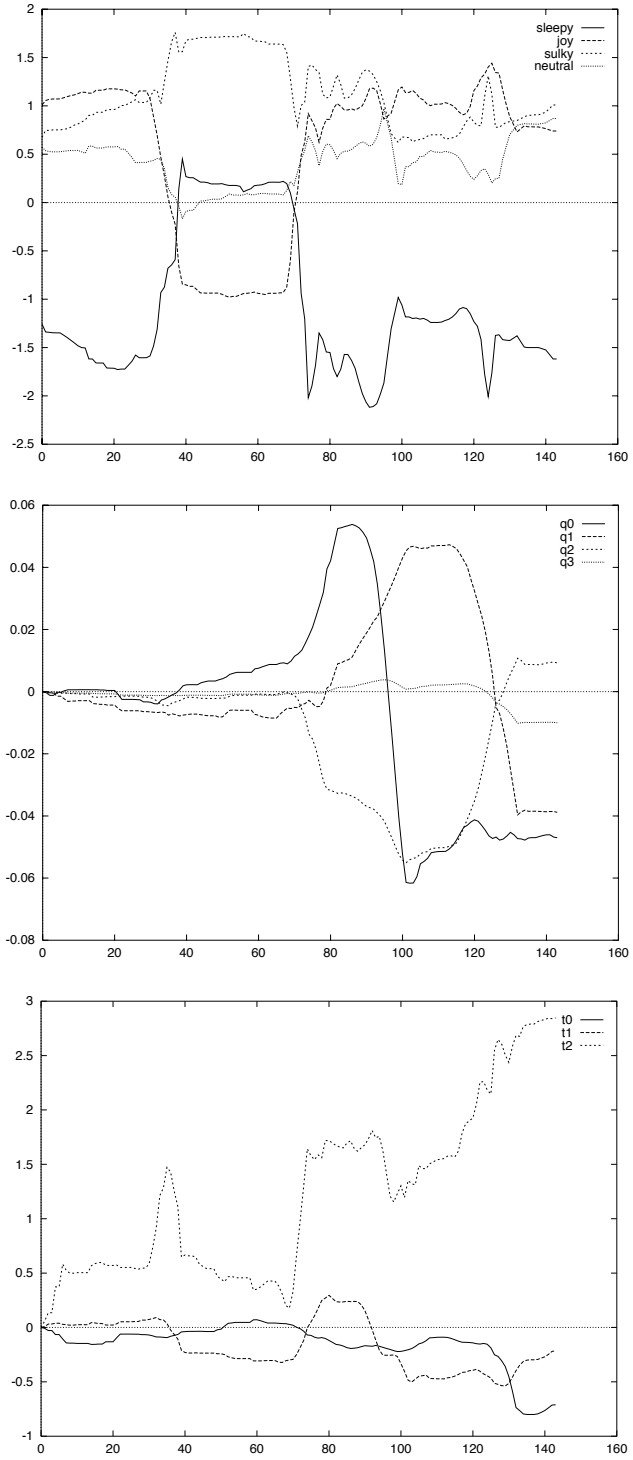


Figure 18: **Parameter plots for sample tracking sequence.** The estimated values of the face parameters are plotted as a function of the frame number. Plot (a) displays the expression weights for the mouth area, plot (b) the rotation parameters and plot (c) the translation parameters. The plots for the position parameters have been shifted so that they have the same origins.

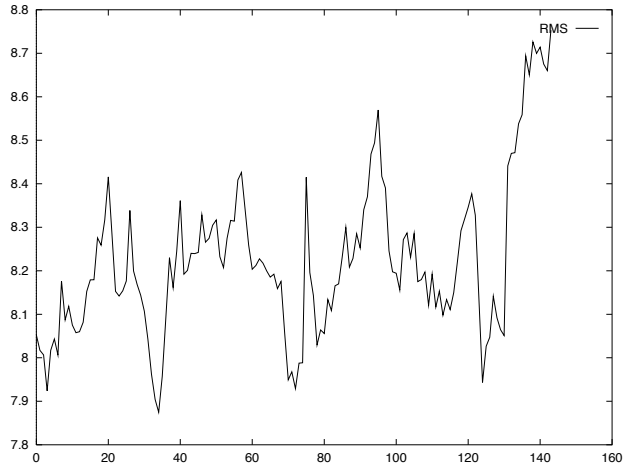


Figure 19: **Error plot for sample tracking sequence.** The RMS error is shown as a function of frame number.

Texture relighting. Currently, extracted textures reflect the lighting conditions under which the photographs were taken. Relighting techniques should be developed for seamless integration of our face models with other elements. Relighting images of real scenes is a difficult problem since it requires estimating the reflectance properties of the objects in the scene. Yu and Malik (Yu and Malik 1998) tackle this problem for images of buildings taken under different illumination conditions. More recently Blanz and Vetter (Blanz and Vetter 1999) developed a technique to extract a 3D face model from a single photograph. They use the Phong illumination model to extract the face albedo from the photo. Although this model seems to work quite well for analyzing face images, for synthesizing face images a more complex model would probably be necessary. Such model was developed recently at the university of Berkeley (Debevec et al. 2000) and has led to very promising results.

Speech animation. The example animation we created features a character reacting to a narration. We chose this particular setting because it is difficult to animate a speaking character with our current system. Animating speech using our technique would require the modeling of multiple mouth positions. It is unclear how many positions would be necessary to produce realistic results: on the one hand hundreds of visemes could be necessary (Bregler, Covell and Slaney 1997); on the other hand traditional cell animation uses only a dozen different positions (Thomas, Johnston and Johnston 1995). Beyond the modeling issue, animating speech through a keyframe animation system would be very tedious. Automatic lip-synching to a video or voice recording would be a much more effective approach (Bregler et al. 1997).

Human Hair. There are typically about 100,000 hair strands on the average person's head, each with a diameter from 40 to 120 microns. This intricate geometry is not well approximated by a polygonal model unless they are fairly short. As a consequence, our modeling technique does not allow us to model hair faithfully. Moreover, since hair is semi-transparent and can be highly specular it is challenging to render realistically. Recent research on synthesizing artificial human hair include Watanabe and Suenaga (Watanabe and Suenaga 1992), Anjyo *et al.* (Anjyo, Usami and Kurihara 1992), and Rosenblum *et al.* (Rosenblum, Carlson and III 1991). All these approaches model single strands of hair and try to simulate the reflectance and dynamic properties of hair. Unfortunately they do not generate convincing photorealistic hair renderings.

9 Conclusion

The synthesis of realistic facial animations is a formidable endeavor. Approaching realism requires an ever increasing attention to small details. The images and animations we synthesized in the scope of this work are a step further towards the creation of synthetic actors that are indistinguishable from real

actors. Producing similar results with a physically-based approach would be hard to achieve since it would require a very accurate simulation of the human face. Our technique has the additional advantage that it does not require expensive hardware capture equipment such as a 3D scanner. We believe image-based approaches are very promising for synthesizing realistic face images and realistic face motions.

Analyzing face images to estimate face motions proved to be also very difficult. In particular, the non-rigid deformations introduced by changes in facial expressions are quite complex to recover. By using a realistic model of the face, we were able to solve this problem without making simplifying assumption about the human face.

References

Akimoto, T., Suenaga, Y. and Wallace, R.: 1993, Automatic creation of 3D facial models, *IEEE Computer Graphics and Applications* **13**(5), 16–22.

Ali: 1995, *Alias V7.0*.

Anjyo, K., Usami, Y. and Kurihara, T.: 1992, A simple method for extracting the natural beauty of hair, *SIGGRAPH 92 Conference Proceedings*, ACM SIGGRAPH, pp. 111–120.

Ballard, D. and Brown, C.: 1982, *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey.

Black, M. and Yacoob, Y.: 1995, Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image otions, *Proceedings, International Conference on Computer Vision*, pp. 374–381.

Blake, A. and Isard, M.: 1998, *Active Contrours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*, Addison Wesley.

Blanz, T. and Vetter, T.: 1999, A morphable model for the synthesis of 3d faces, *SIGGRAPH 99 Conference Proceedings*, ACM SIGGRAPH.

Bregler, C., Covell, M. and Slaney, M.: 1997, Video rewrite: driving visual speech with audio, *SIGGRAPH 97 Conference Proceedings*, ACM SIGGRAPH, pp. 353–360.

Cascia, M. L., Isidoro, J. and Sclaroff, S.: 1998, Head tracking via robust registration in texture map images, *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, D., State, A. and Banks, D.: 1995, Interactive shape metamorphosis, *1995 Symposium on Interactive 3D Graphics*, ACM SIGGRAPH, pp. 43–44.

Choi, C., Aizawa, K., Harshima, H. and Takebe, T.: 1994, Analysis and synthesis of facial image sequences in model-based image coding, *IEEE Transactions on Circuits and Systems for Video Technology* **4**(3), 257–274.

Choi, C. S., Kiyoharu, Harashima, H. and Takebe, T.: 1994, Analysis and synthesis of facial image sequences in model-based image coding, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, pp. 257 – 275.

Covell, M.: 1996, Eigen-points: control-point location using principal component analysis, *Proceedings, Second International Conference on Automatic Face and Gesture Recognition*, pp. 122–127.

Cyb: 1990, *4020/RGB 3D Scanner with Color Digitizer*.

Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W. and Sagar, M.: 2000, Acquiring the reflectance field of a human face, *SIGGRAPH 2000 Conference Proceedings*, ACM SIGGRAPH, pp. 35–42.

Debevec, P., Taylor, C. and Malik, J.: 1996, Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach, *SIGGRAPH 96 Conference Proceedings*, ACM SIGGRAPH, pp. 11–20.

Decarlo, D. and Metaxas, D.: 1998, Deformable model-based shape and motion analysis from images using motion residual error, *Proceedings, First International Conference on Computer Vision*, pp. 113–119.

Devernay, F. and Faugeras, O.: 1994, Computing differential properties of 3d shapes from stereoscopic images without 3d models, *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pp. 208–213.

E. Ostby, Pixar Animation Studios: 1997, Personal communication.

Edwards, G., Taylor, C. and Cootes, T.: 1998, Interpreting face images using active appearance models, *Proceedings, Third Workshop on Face and Gesture Recognition*, pp. 300–305.

Essa, I. and Pentland, A.: 1997, Coding, analysis, interpretation, and recognition of facial expressions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(7), 757–763.

Faugeras, O.: 1993, *Three-dimensional computer vision: A geometric viewpoint*, MIT Press, Cambridge, Massachusetts.

Fua, P. and Miccio, C.: 1998, From regular images to animated heads: a least square approach, *Proceedings, European Conference on Computer Vision*, pp. 188–202.

Golub, G. and Van Loan, C.: 1996, *Matrix Computation, third edition*, The John Hopkins University Press, Baltimore and London.

Gortler, S., Grzeszczuk, R., Szeliski, R. and Cohen, M.: 1996, The Lumigraph, *SIGGRAPH 96 Conference Proceedings*, ACM SIGGRAPH, pp. 43–54.

Guenter, B., Grimm, C., Wood, D., Malvar, H. and Pighin, F.: 1998, Making faces, *SIGGRAPH 98 Conference Proceedings*, ACM SIGGRAPH, pp. 55–66.

Hager, G. and Belhumeur, P.: 1996, Real-time tracking of image regions with changes in geometry and illumination, *Proceedings, Computer Vision and Pattern Recognition*, pp. 403–410.

Hallinan, P.: 1994, A low-dimensional representation of human faces for arbitrary lighting conditions, *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pp. 58–66.

Hanrahan, P. and Kruger, W.: 1993, Reflection from layered surfaces due to surface scattering, *SIGGRAPH 93 Conference Proceedings*, pp. 165–174.

Inc, B. S. T.: 1993, *Beginning Reading Software*, Sierra On-Line, Inc.

Ip, H. and Yin, L.: 1996, Constructing a 3D individualized head model from two orthogonal views, *The Visual Computer* **12**, 254–266.

Jones, M. and Poggio, T.: 1998, Hierarchical morphable models, *Proceedings, International Conference on Computer Vision*, pp. 820–826.

Kass, M., Witkin, A. and Terzopoulos, D.: 1987, Snakes: Active contour models, *Proceedings, First International Conference on Computer Vision*, pp. 259–268.

Koch, R., Gross, M., Carls, F. R., von Büren, D., Fankhauser, G. and Parish, Y.: 1996, Simulating facial surgery using finite element methods, *SIGGRAPH 96 Conference Proceedings*, ACM SIGGRAPH, pp. 421–428.

Kurihara, T. and Arai, K.: 1991, A transformation method for modeling and animation of the human face from photographs, *in* N. M. Thalmann and D. Thalmann (eds), *Computer Animation 91*, Springer-Verlag, Tokyo, pp. 45–58.

Lanitis, A., Taylor, C. and Cootes, T.: 1995, A unified approach for coding and interpreting face images, *Fifth International Conference on Computer Vision (ICCV 95)*, Cambridge, Massachusetts, pp. 368–373.

Leclerc, Y. and Bobick, A.: 1991, The direct computation of height from shading, *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*.

Lee, S., Chwa, K., Shin, S. and Wolberg, G.: 1995, Image metamorphosis using snakes and free-form deformations, *SIGGRAPH 95 Conference Proceedings*, ACM SIGGRAPH, pp. 439–448.

Lee, S., Wolberg, G., Chwa, K. and Shin, S.: 1996, Image metamorphosis with scattered feature constraints, *IEEE Transactions on Visualization and Computer Graphics* **2**(4).

Lee, Y., Terzopoulos, D. and Waters, K.: 1995, Realistic modeling for facial animation, *SIGGRAPH 95 Conference Proceedings*, ACM SIGGRAPH, pp. 55–62.

Lengagne, R., Fua, P. and Monga, O.: 1998, 3d face modeling using stereo and differential constraints, *Proceedings of the Third International Conference on Automatic Face and Gesture Recognition*.

Li, H., Roivainen, P. and Forchheimer, R.: 1993, 3-d motion estimation in model-based facial image coding, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(6), 545–555.

Matsino, K., Lee, C., Kimura, S. and Tsuji, S.: 1995, Automatic recognition of human facial expressions, *Proceedings of the IEEE*, pp. 352–359.

Moffitt, F. and Mikhail, E.: 1980, *Photogrammetry*, 3 edn, Harper & Row, New York.

Nielson, G.: 1993, Scattered data modeling, *IEEE Computer Graphics and Applications* **13**(1), 60–70.

Parke, F.: 1972, Computer generated animation of faces, *Proceedings ACM annual conference.* .

Parke, F.: 1974, *A parametric model for human faces*, PhD thesis, University of Utah, Salt Lake City, Utah. UTEC-CSc-75-047.

Pighin, F., Hecker, J., Lischinski, D., Szeliski, R. and Salesin, D.: 1998, Synthesizing realistic facial expressions from photographs., *SIGGRAPH 98 Conference Proceedings*, ACM SIGGRAPH, pp. 75–84.

Pighin, F., Szeliski, R. and Salesin, D.: 1999, Resynthesizing facial animation through 3d model-based tracking, *Proceedings, International Conference on Computer Vision*.

Press, W., Flannery, B., Teukolsky, S. and Vetterling, W.: 1992, *Numerical Recipes in C: The Art of Scientific Computing*, second edn, Cambridge University Press, Cambridge, England.

Proesman, M., Gool, L. V. and Oosterlinck, A.: 1996, Active acuisition of 3d shape for moving objects, *International Conference on Image Processing*.

Pulli, K., Cohen, M., Duchamp, T., Hoppe, H., Shapiro, L. and Stuetzle, W.: 1997, View-based rendering: Visualizing real objects from scanned range and color data, *Proc. 8th Eurographics Workshop on Rendering*.

Rosenblum, R., Carlson, W. and III, E. T.: 1991, Physically-based facial modeling, analysis, and animation, *Journal of Visualization and Computer Animation* **2**(4), 141–148.

Schodl, A., Ario, A. and Essa, I.: 1998, Head tracking using a textured polygonal model, *Workshop on Perceptual User Interfaces*, pp. 43–48.

Slama, C. (ed.): 1980, *Manual of Photogrammetry*, fourth edn, American Society of Photogrammetry, Falls Church, Virginia.

Szeliski, R. and Kang, S.: 1994, Recovering 3D shape and motion from image streams using nonlinear least squares, *Journal of Visual Communication and Image Representation* **5**(1), 10–28.

Szeliski, R. and Shum, H.: 1997, Creating full view panoramic image mosaics and texture-mapped models, *SIGGRAPH 97 Conference Proceedings*, ACM SIGGRAPH, pp. 251–258.

Terzopoulos, D. and Waters, K.: 1993, Analysis and synthesis of facial image sequences using physical an anatomical models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 569–579.

Thomas, F., Johnston, O. and Johnston, C.: 1995, *The Illusion of Life*, Hyperion.

Thórisson, K.: 1997, Gandalf: An embodied humanoid capable of real-time multimodal dialogue with people, *First ACM International Conference on Autonomous Agents*.

Turk, M. and Pentland, A.: 1987, Eigenfaces for recognition, *Journal of Cognitive Neuroscience* **3**(1), 71–87.

Vannier, M., Marsh, J. and Warren, J. O.: 1983, Three-dimentional computer graphics for craniofacial surgical planning and evaluation, *SIGGRAPH 83 Conference Proceedings*, Vol. 17, ACM SIGGRAPH, pp. 263–273.

Vetter, T. and Blanz, V.: 1998, Estimating coloured 3d face models from single images: an example based approach, *Proceedings, European Conference on Computer Vision*, pp. 499–513.

Watanabe, Y. and Suenaga, Y.: 1992, A trigonal prism-based method for hair image generation, *IEEE Computer Graphics and Applications* **12**(1), 47–53.

Williams, L.: 1990, Performance-driven facial animation, *SIGGRAPH 90 Conference Proceedings*, Vol. 24, pp. 235–242.

Yu, Y. and Malik, J.: 1998, Recovering photometric properties of architectural scenes from photographs, *SIGGRAPH 98 Conference Proceedings*, ACM SIGGRAPH, pp. 207–217.

A Least squares for pose recovery from photographs

To solve for a subset of the parameters given in Equation (2), we use linear least squares. In general, given a set of linear equations of the form

$$\boldsymbol{a}_j \cdot \boldsymbol{x} \quad b_j = 0, \tag{8}$$

we solve for the vector $\boldsymbol{x}$ by minimizing

$$\sum_j (\boldsymbol{a}_j \cdot \boldsymbol{x} \quad b_j)^2. \tag{9}$$

Setting the partial derivative of this sum with respect to $\boldsymbol{x}$ to zero, we obtain

$$\sum_j (\boldsymbol{a}_j \boldsymbol{a}_j^T)\boldsymbol{x} \quad b_j \boldsymbol{a}_j = 0, \tag{10}$$

i.e., we solve the set of *normal equations* (Golub and Van Loan 1996)

$$\sum_j \boldsymbol{a}_j \boldsymbol{a}_j^T \quad \boldsymbol{x} = \sum_j b_j \boldsymbol{a}_j. \tag{11}$$

More numerically stable methods such as QR decomposition or Singular Value Decomposition (Golub and Van Loan 1996) can also be used to solve the least squares problem, but we have not found them to be necessary for our application.

To update one of the parameters, we simply pull out the relevant linear coefficient $\boldsymbol{a}_j$ and scalar value b_j from Equation (2). For example, to solve for $\boldsymbol{m}_i$, we set

$$\begin{aligned} \boldsymbol{a}_{2k+0} &= w_i^k(x_i^k \eta^k \boldsymbol{r}_z^k \quad s^k \boldsymbol{r}_x^k), \qquad b_{2k+0} = w_i^k(s^k t_x^k \quad x_i^k) \\ \boldsymbol{a}_{2k+1} &= w_i^k(y_i^k \eta^k \boldsymbol{r}_z^k \quad s^k \boldsymbol{r}_y^k), \qquad b_{2k+1} = w_i^k(s^k t_y^k \quad y_i^k). \end{aligned}$$

For a scalar variable like s^k, we obtain scalar equations

$$\begin{aligned} \boldsymbol{a}_{2k+0} &= w_i^k(\boldsymbol{r}_x^k \cdot \boldsymbol{m}_i + t_x^k), \qquad b_{2k+0} = w_i^k \;\; x_i^k + x_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i) \\ \boldsymbol{a}_{2k+1} &= w_i^k(\boldsymbol{r}_y^k \cdot \boldsymbol{m}_i + t_y^k), \qquad b_{2k+1} = w_i^k \;\; y_i^k + y_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{m}_i) \;\; . \end{aligned}$$

Similar equations for $\boldsymbol{a}_j$ and b_j can be derived for the other parameters t_x^k, t_y^k, and η^k. Note that the parameters for a given camera k or 3D point i can be recovered independently of the other parameters.

Solving for rotation is a little trickier than for the other parameters, since $\boldsymbol{R}$ must be a valid rotation matrix. Instead of updating the elements in $\boldsymbol{R}_k$ directly, we replace the rotation matrix $\boldsymbol{R}^k$ with $\tilde{\boldsymbol{R}}\boldsymbol{R}^k$ (Szeliski and Shum 1997), where $\tilde{\boldsymbol{R}}$ is given by Rodriguez's formula (Faugeras 1993):

$$\tilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}, \theta) = \hat{\boldsymbol{I}} + \sin\theta \boldsymbol{X}(\hat{\boldsymbol{n}}) + (1 \quad \cos\theta)\boldsymbol{X}^2(\hat{\boldsymbol{n}}), \tag{12}$$

where θ is an incremental rotation angle, $\hat{\boldsymbol{n}}$ is a rotation axis, and $\boldsymbol{X}(\boldsymbol{v})$ is the cross product operator

$$\boldsymbol{X}(\boldsymbol{v}) = \begin{bmatrix} 0 & v_z & v_y \\ v_z & 0 & v_x \\ v_y & v_x & 0 \end{bmatrix}. \tag{13}$$

A first order expansion of $\tilde{\boldsymbol{R}}$ in terms of the entries in $\boldsymbol{v} = \theta\hat{\boldsymbol{n}} = (v_x, v_y, v_z)$ is given by $\hat{\boldsymbol{I}} + \boldsymbol{X}(\boldsymbol{v})$.

Substituting into Equation (2) and letting $\boldsymbol{q}_i = \boldsymbol{R}^k\boldsymbol{m}_i$, we obtain

$$\begin{array}{cccc} w_i^k & x_i^k + x_i^k\eta^k(\tilde{\boldsymbol{r}}_z^k \cdot \boldsymbol{q}_i) & s^k(\tilde{\boldsymbol{r}}_x^k \cdot \boldsymbol{q}_i + t_x^k) & = 0 \\ w_i^k & y_i^k + y_i^k\eta^k(\tilde{\boldsymbol{r}}_z^k \cdot \boldsymbol{q}_i) & s^k(\tilde{\boldsymbol{r}}_y^k \cdot \boldsymbol{q}_i + t_y^k) & = 0, \end{array} \tag{14}$$

where $\tilde{\boldsymbol{r}}_x^k = (1, \quad v_z, v_y)$, $\tilde{\boldsymbol{r}}_y^k = (v_z, 1, \quad v_x)$, $\tilde{\boldsymbol{r}}_z^k = (\quad v_y, v_x, 1)$, are the rows of $[\hat{\boldsymbol{I}} + \boldsymbol{X}(\boldsymbol{v})]$. This expression is linear in (v_x, v_y, v_z), and hence leads to a 3 3 set of normal equations in (v_x, v_y, v_z). Once the elements of $\boldsymbol{v}$ have been estimated, we can compute θ and $\hat{\boldsymbol{n}}$, and update the rotation matrix using

$$\boldsymbol{R}^k \qquad \tilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}^k, \theta^k)\boldsymbol{R}^k.$$

B Analytical Jacobian for face tracking

We detail further the computation of the analytical Jacobian started in section 2.3. The partial derivatives of P, $\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$ and w_k with respect to a parameter p_i are examined.

Partial derivatives of P

P is a mapping from 3D to 2D that takes a point $\boldsymbol{m} = (X, Y, Z)$ as a parameter.

$$\frac{\partial P_x}{\partial \boldsymbol{m}} = \begin{array}{c} \frac{f}{Z} \\ 0 \\ f\frac{X}{Z^2} \end{array} \qquad \text{and} \qquad \frac{\partial P_y}{\partial \boldsymbol{m}} = \begin{array}{c} 0 \\ \frac{f}{Z} \\ f\frac{Y}{Z^2} \end{array}$$

Partial derivatives of $\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$

The partial derivative with respect to p_i can be expanded as:

$$\frac{\partial(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})}{\partial p_i} = \frac{\partial \boldsymbol{R}}{\partial p_i}\boldsymbol{m}_j + \boldsymbol{R}\frac{\partial \boldsymbol{m}_j}{\partial p_i} + \frac{\partial \boldsymbol{t}}{\partial p_i}$$

This expression depends on the nature of p_i.

- If p_i is a blending weight ($p_i = \alpha_l$): since the model geometry is a linear combination of the basic expressions geometry, the point m_j is a linear combination of points from the basic expressions: m_{jk}. We can then apply the following derivations:

$$\frac{\partial(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})}{\partial \alpha_l} = \boldsymbol{R}\frac{\partial \boldsymbol{m}_j}{\partial \alpha_l} = \boldsymbol{R}\frac{\partial(\quad_k w_k\boldsymbol{m}_{jk})}{\partial \alpha_l} = \boldsymbol{R} \quad_k \frac{\partial w_k}{\partial \alpha_l}\boldsymbol{m}_{jk}$$

- If $p_i = \boldsymbol{R}$: we replace the rotation matrix $\boldsymbol{R}$ with $\hat{\boldsymbol{R}}\boldsymbol{R}$; where $\hat{\boldsymbol{R}}$ is parameterized by an angular velocity $\omega = (\omega_x, \omega_y, \omega_z)$, and is given by Rodriguez's formula:

$$\hat{\boldsymbol{R}}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + sin\theta\boldsymbol{X}(\hat{\boldsymbol{n}}) + (1 \quad cos\theta)\boldsymbol{X}^2(\hat{\boldsymbol{n}})$$

  where $\theta = \ \omega\ $, $\hat{\boldsymbol{n}} = \omega/\theta$ and $\boldsymbol{X}(\hat{\boldsymbol{n}})$ is the cross product operator:

$$\boldsymbol{X}(\omega) = \begin{array}{ccc} 0 & \omega_z & \omega_y \\ \omega_z & 0 & \omega_x \\ \omega_y & \omega_x & 0 \end{array}$$

  A first order approximation of $\hat{\boldsymbol{R}}$ is given by $\boldsymbol{I} + \boldsymbol{X}(\omega)$. The rotation $\hat{\boldsymbol{R}}\boldsymbol{R}$ is parameterized by the vector ω. We are thus interested in computing the partial derivatives of $\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$ according to some ω_l, component of ω:

$$\frac{\partial(\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t})}{\partial \omega_l} = \frac{\partial \boldsymbol{R}}{\partial \omega_l}\boldsymbol{m}_j = \frac{\partial(\boldsymbol{I} + \boldsymbol{X}(\omega))}{\partial \omega_l}\boldsymbol{R}\boldsymbol{m}_j = \frac{\partial \boldsymbol{X}(\omega)}{\partial \omega_l}\boldsymbol{R}\boldsymbol{m}_j$$

Once a step in the LM algorithm has been computed, the rotation $\boldsymbol{R}$ is updated using

$$\boldsymbol{R} \quad \hat{\boldsymbol{R}}\boldsymbol{R}$$

- If $p_i = t_x$, or $p_i = t_y$, or $p_i = t_z$, then the partial derivative is equal to $(1, 0, 0)$, or $(0, 1, 0)$, or $(0, 0, 1)$ respectively.

Partial derivatives of w_k

Clearly:

$$\frac{\partial w_k}{\partial p_i} = 0, \qquad \text{if } p_i \text{ is not an expression parameter.}$$

If p_i is an expression parameter, for instance α_l, then the partial derivative can be computed by differentiating equation (1):

$$\frac{\partial w_k}{\partial \alpha_l} = \boldsymbol{Q}(k, l)$$