Resynthesizing Facial Animation through 3D Model-Based Tracking

Frédéric Pighin[†] Richard Szeliski [‡] David H. Salesin[†‡]

[†]University of Washington
[‡]Microsoft Research

Abstract

Given video footage of a person's face, we present new techniques to automatically recover the face position and the facial expression from each frame in the video sequence. A 3D face model is fitted to each frame using a continuous optimization technique. Our model is based on a set of 3D face models that are linearly combined using 3D morphing. Our method has the advantages over previous techniques of fitting directly a realistic 3-dimensional face model and of recovering parameters that can be used directly in an animation system. We also explore many applications, including performance-driven animation (applying the recovered position and expression of the face to a synthetic character to produce an animation that mimics the input video), relighting the face, varying the camera position, and adding facial ornaments such as tattoos and scars.

1 Introduction

There are many techniques and tools that can be used to create facial animations. These tools can be as simple as a pencil and a sketchpad or as complex as a physically-based 3D face model. In either case, creating believable facial animations is a very delicate task that only talented and well-trained animators can achieve. An alternative is to generate facial animations based on video footage of a real person's face. The face can be tracked throughout the video by recovering the position and expression at each frame. This information can then be optionally modified further to create a novel animation.

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 than 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 allows the robust estimation of the face parameters is thus a very challenging task.

This problem can be made easier by using markers on the face such as heavy makeup [23] or a set of colored dots stuck onto the actor's face [12, 25]. Once the position of the markers has been determined, the position of the face and the facial features can easily be derived. Williams [25] pioneered this technique, tracking 2D points on a single image. Guenter *et al.* [12] 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 can be leveraged to extract information from a video of an unmarked face. These tracking techniques can be classified into two groups depending on whether they use a two-dimensional or three-dimensional model.

Two-dimensional facial feature trackers can be based on deformable or rigid patches [3, 7, 13], or on edge or feature detectors [4, 16, 18]. The use of a 2D model limits the range of face orientations that can be handled. For large variations of face orientation, the three-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 correspond to the target image. Terzopoulos and Waters [23] 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 [11] built a physically-based face model and developed a control-theoretic technique to fit it to a sequence of images. Decarlo and Metaxas [8] employed a similar model that incorporates possible variations in head shape using anthropometric measurements. Comparable techniques [1, 6, 17] are used in model-based image coding schemes where the model parameters provide a compact representation of video frames. These approaches use a 3D articulated face model to derive a model for facial motion. Using the estimated motion, the 3D model can be animated to synthesize an animation similar to the input sequence. However, the synthesized images are not used for 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.* [15] model the face with a texture-mapped cylinder. The textures are extracted by building a mosaic from the input image sequence. Schodl *et al.* [22] use a more detailed geometric model to estimate the face position and orientation. These two models however allow neither the facial expression nor the identity of the face in the target image to be estimated. Vetter and Blanz [24] 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 attempt to estimate the identity of the person but rather to track the person's facial expression and head pose.

In this paper, we present our 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 based on the work of Pighin *et al.* [20] and uses 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.

Tracking faces can be employed to create novel facial animations. For instance the extracted facial parameters can be reused to animate synthetic faces. Such performance-driven animations produce a sequence that mimics the original input video [10, 19, 25]. Performance-driven animations are useful for avatar control, identity hiding, and easy home-made animations. Another application is to modify the input video to produce a novel animation or special effects. For instance, Bregler *et al.* [5] segment the input video into small units corresponding to visemes. These video segments can then be rescheduled according to a different soundtrack to produce a novel video.

In this paper, we present several applications. We reuse the recovered face parameters to generate performance-driven animations. Moreover, since the fitted model provides us with an approximation of the geometry of the face, we can alter the input video. These alterations include changing the lighting conditions, changing the camera angle, and adding facial decorations such as tattoos and scars.

The remainder of this paper first describes the model fitting technique we developed and then illustrates several ways to generate novel animations or special effects from the original footage.

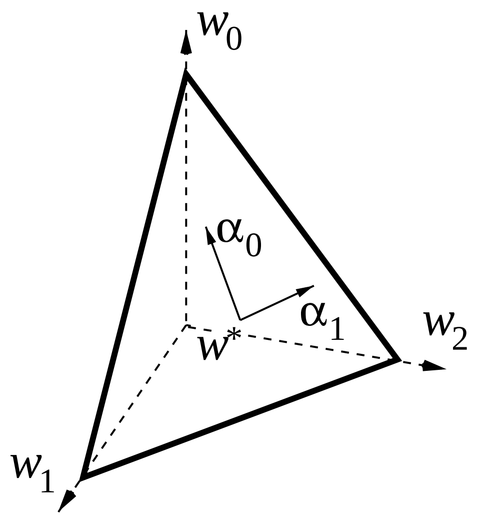


Figure 1: Relationship between blending weights and expression parameters with 3 basic expressions.

2 Model fitting

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

2.1 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 varying locally the percentage of each basic expression on different parts of the face, a wide range of expressions can be generated.

We synthesize realistic face models using a technique developed by Pighin *et al.* [20]. The 3D models are recovered by fitting a generic face model to a set of photographs of a person's face. The texture map is extracted by combining the different photographs into a single cylindrical texture map.

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 face, and a rotation $\boldsymbol{R}$, which indicates its orientation. The facial expression is determined by a set of blending weights $w_1, ..., w_n$, one for each basic expression in the model. We constrain these weights to sum to 1. To enforce this constraint, we use a set of expression parameters $\alpha_1, ..., \alpha_{n-1}$, which are obtained by projecting the weights w_k onto the hyperplane $\sum_k w_k = 1$. The vector of blending weights $\boldsymbol{w} = (w_1, ..., w_n)$ is decomposed using the vector of expressions parameters $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n-1})$:

$$\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{\alpha} + \boldsymbol{w}^* \tag{1}$$

where $\boldsymbol{Q}$ is a matrix whose column vectors span the hyperplane $\sum_k w_k = 0$ and $\boldsymbol{w}^*$ is a vector belonging to $\sum_k w_k = 1$. Figure 1 illustrates the relationship between the blending weights and the expression parameters. 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 used in our experiments is shown in figure 2.

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 $\hat{\boldsymbol{I}}$:

$$\hat{\boldsymbol{I}} = \sum_k w_k \hat{\boldsymbol{I}}_k \tag{2}$$

2.2 Optimization procedure

Fitting a linear combination of faces to an image has already been explored by different research groups. Edwards *et al.* [9] 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 [14] 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.

To fit our model to a target face image $\boldsymbol{I}_t$, we developed 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 $\hat{\boldsymbol{I}}(\boldsymbol{p})$:

$$\begin{aligned} \epsilon(\boldsymbol{p}) &= \frac{1}{2}\left\|\boldsymbol{I}_t - \hat{\boldsymbol{I}}(\boldsymbol{p})\right\|^2 + D(\boldsymbol{p}) \\ &= \frac{1}{2}\sum_j [\boldsymbol{I}_t(x_j, y_j) - \hat{\boldsymbol{I}}(\boldsymbol{p})(x_j, y_j)]^2 + D(\boldsymbol{p}) \end{aligned} \tag{3}$$

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 function $D(\boldsymbol{p})$ is a piecewise-quadratic function of the following form:

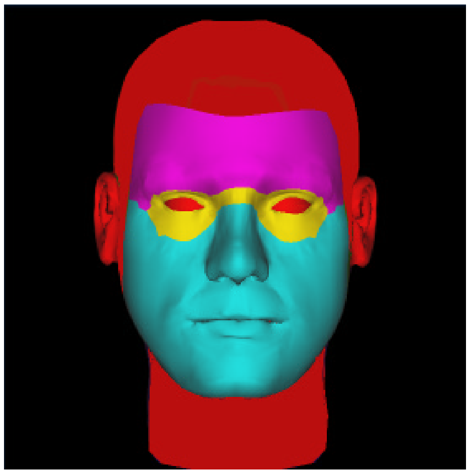


Figure 2: Partitioning of the face. The different regions are rendered with different colors.

$$D(\boldsymbol{p}) = c\sum_k [\min(0, w_k)^2 + \max(0, w_k - 1)^2] \tag{4}$$

where c is a constant.

The parameters are fitted 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) algorithm [21] to minimize the function ϵ. We tried other optimization techniques such as steepest descent and Newton (BFGS), but the LM algorithm produced better results for the estimation of the expression parameters.

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:

$$[\boldsymbol{J}(\boldsymbol{p})^T\boldsymbol{J}(\boldsymbol{p}) + \lambda\boldsymbol{I} + H(\boldsymbol{p})]\delta\boldsymbol{p} = -\boldsymbol{J}(\boldsymbol{p})^T[\boldsymbol{I}_t - \hat{\boldsymbol{I}}(\boldsymbol{p})] - \nabla D(\boldsymbol{p}) \tag{5}$$

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

2.3 Computing the Jacobian images

We explored 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

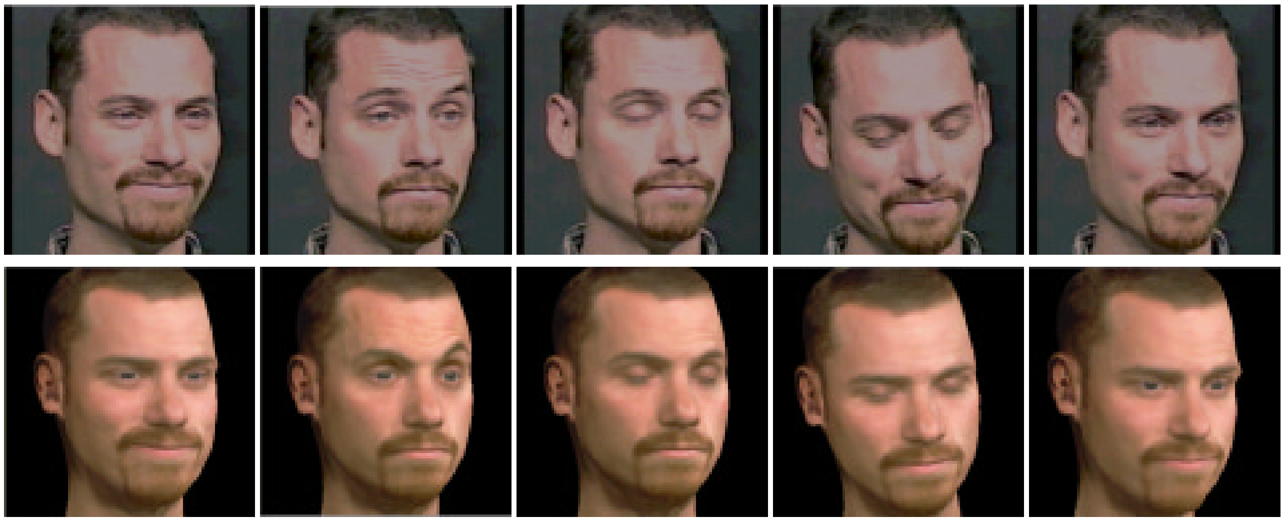


Figure 3: Tracking sequence. The bottom row shows the result of fitting our model to the target images on the top row.

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.

Finite differences

We use central differences to approximate the value of the Jacobian. This is more costly than using forward differences but produces a better approximation of the derivatives. Each entry of the Jacobian matrix is computed using the following formula:

$$\frac{d\hat{\boldsymbol{I}}}{dp_i}(\boldsymbol{p}) \approx \frac{\hat{\boldsymbol{I}}(\boldsymbol{p} + \Delta p_i \boldsymbol{e}_i) - \hat{\boldsymbol{I}}(\boldsymbol{p} - \Delta p_i \boldsymbol{e}_i)}{2\Delta p_i} \tag{6}$$

where $\boldsymbol{e}_i$ is the **i**-th vector of the canonical basis and Δp_i is a variation of the parameter p_i. In practice, we determined appropriate values for Δp_i by experimentation.

Analytical Jacobian

Given that $\hat{\boldsymbol{I}}$ is a function of the position in the image plane (x_j, y_j) for a given pixel and of the model parameters p_i, we have:

$$\frac{d\hat{\boldsymbol{I}}}{dp_i} = \frac{\partial \hat{\boldsymbol{I}}}{\partial x_j}\frac{\partial x_j}{\partial p_i} + \frac{\partial \hat{\boldsymbol{I}}}{\partial y_j}\frac{\partial y_j}{\partial p_i} + \frac{\partial \hat{\boldsymbol{I}}}{\partial p_i}, \tag{7}$$

The vector $(\frac{\partial \hat{\boldsymbol{I}}}{\partial x_j}, \frac{\partial \hat{\boldsymbol{I}}}{\partial y_j})$ is the intensity gradient of image $\hat{\boldsymbol{I}}$ at pixel j. It is computed using a Sobel filter [2]. The partial derivative $\frac{\partial \hat{\boldsymbol{I}}}{\partial p_i}$ can be computed by differentiating equation (2):

$$\frac{\partial \hat{\boldsymbol{I}}}{\partial p_i} = \sum_k \frac{\partial w_k}{\partial p_i}\hat{\boldsymbol{I}}_k. \tag{8}$$

To compute the partial derivatives $\frac{\partial x_j}{\partial p_i}$ and $\frac{\partial y_j}{\partial p_i}$, 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:

$$\frac{\partial x_j}{\partial p_i} = \frac{\partial P_x}{\partial \boldsymbol{m}'_j}^T \frac{\partial \boldsymbol{m}'_j}{\partial p_i} \tag{9}$$

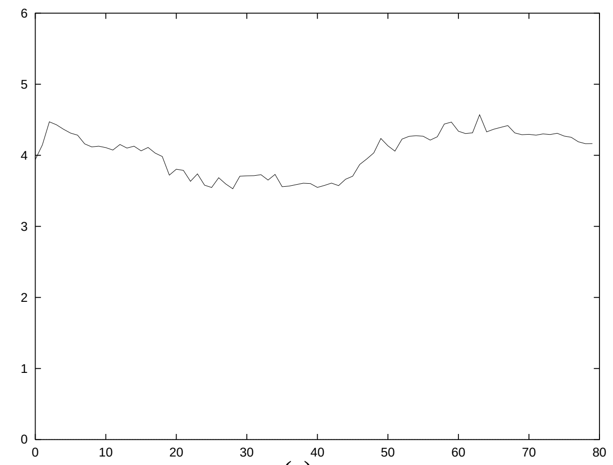
$$\frac{\partial y_j}{\partial p_i} = \frac{\partial P_y}{\partial \boldsymbol{m}'_j}^T \frac{\partial \boldsymbol{m}'_j}{\partial p_i}$$

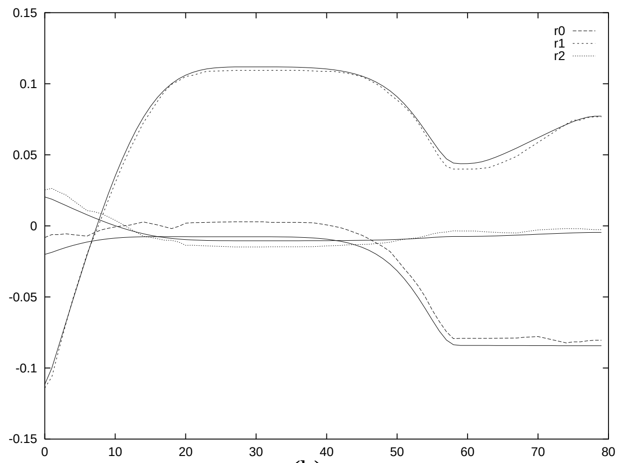
where $\boldsymbol{m}'_j = \boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$. We detail the computation of the partial derivatives of P, $\boldsymbol{R}\boldsymbol{m}_j + \boldsymbol{t}$ and w_k in Appendix A.

2.4 Tracking results

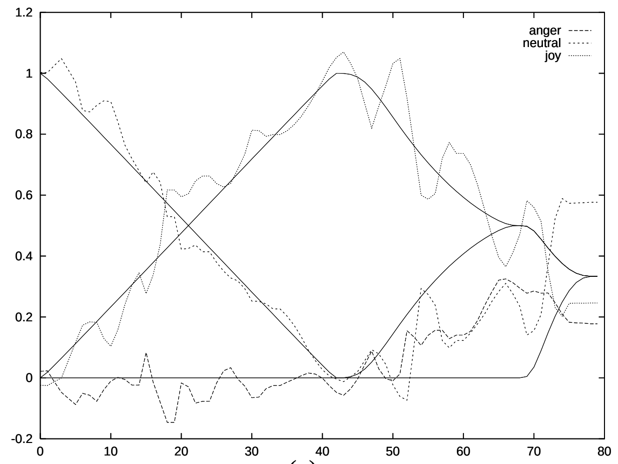
The experiments we ran use a partition of the face into three regions: the mouth area, the eyes area, and the forehead, as illustrated in figure 2. In all our experiments, the values of the parameters are initialized with a rough visual guess.

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 expressions parameters, the finite differences Jacobian produced better results than the analytical version. This difference can


(a)



(b)



(c)

Figure 4: Plots for tracking synthetic sequence. The RMS error, rotation parameters expressed in radian (b), and expression parameters (c) are plotted as functions of frame numbers. The ground truth values of the parameters (solid) are compared to the estimated values (dots and dashes).

be explained in terms of the difference in behavior of the error function on these two set of parameters. The dependence of the error function on the position parameters is fairly smooth, while its dependence with respect to the expression parameters is not.

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 cameras were used to build the model textures and a camcorder 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 by using bandpass filtering on both the target images and the model renderings when estimating the expression parameters.

We explored different ways of choosing the expression parameters. These parameters are specified by the basis $\boldsymbol{Q}$. Each parameter represents a direction in the linear space of expressions spanned by the basic expressions $E_1, ..., E_n$. A natural set of directions is the differences between a particular expression, for instance E_1, and the rest of the expressions: $E_2 - E_1, ..., E_n - E_1$. This parameterization does not take into account possible similarities between expressions and could result in directions that are correlated. Choosing a set of orthogonal direction can improve the robustness of the facial parameters estimation. We used an orthogonalization technique (Gram-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.

We used a simple prediction scheme to fit the model to a sequence of frames. The parameters are linearly extrapolated using the results from the two previous frames.

Figure 3 illustrates some tracking results using our technique. The model used four basic expressions: sleepy, joy, sulky, and neutral. Each frame took on average 2 minutes to be fitted at a 300 by 300 resolution. Although these performances are far from permitting real time tracking, all the applications we explored can be done as postprocessing on a video. A limitation of our approach is seen in the second column where the shape of the eyebrows in the fitted model do not match these of the target frame. This problem occurs because this particular eyebrow shapes are not included in the linear space spanned by the set of basic expressions.

To further test our tracking technique, we generated a synthetic animation using our face model. The model used three basic expressions: anger, neutral, and joy. We analyzed this sequence with the same model. The plots in figure 4 describe the tracking results as a function of frame number. The evo-

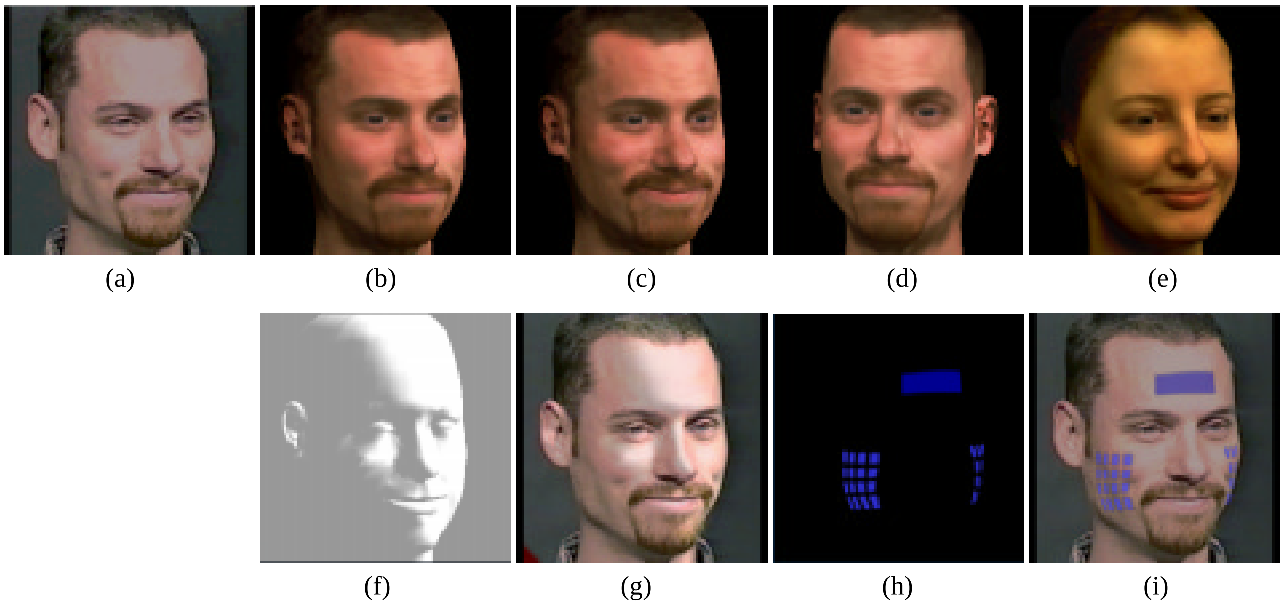


Figure 5: Applications. Animations of synthetic models are shown on the top row. The model (rendered in (b)) has been fitted to the target image (a). For the same frame, the facial expression can be exaggerated (c), the viewpoint can be changed (d), or the animation parameters can be used for a different model (e). The bottom row illustrates modifications of the target frame (a). The lighting has been changed in (g) by introducing a synthetic light source. A greyscale rendering of the lighted estimated geometry (f) was used to modulate the target image. Finally, synthetic elements can be added to the face such as the tattoos in (i). These tattoos were rendered separately using the 3D model (h) and then composited on top of the target image.

lution of the root mean square error is given in figure 4(a). The estimated head position and the blending parameters in the forehead region for the same sequence are illustrated in figure 4(b) and figure 4(c) respectively. The position and orientation parameters are estimated fairly precisely, whereas the estimated values of the expression parameters are significantly different from their true values. Although the expression parameters are not accurately estimated, the rendered fitted model at each frame is very close to the input frames.

3 Applications

Once we have recovered the position and the geometry of the face at each frame, we can use this information to generate novel animations. We considered two different types of applications. The first one is to apply the recovered parameters to a synthetic face to produce a performance-driven animation. The second set of applications is to alter the original video footage using the recovered 3D geometry. These alterations include changing the lighting, and adding computer graphics elements to decorate the face. Figure 5 illustrates these applications. A set of mpeg movie files provide more complete examples. A description of these files can be found in Appendix B.

3.1 Performance-driven animation

The recovered parameters can be used to animate any face model using an animation tool. The animation parameters can be edited to produce different animations; for instance, the expressions can be exaggerated or the viewpoint can be changed. These applications are illustrated in figure 5. The target image in figure 5(a) has been fitted to the model rendered in 5(b). As shown in figure 5(c), the same expression can be exaggerated. This exaggeration was obtained by multiplying the expression parameters by a positive constant. For a teleconferencing system the synthetic reconstruction can be seen from a different angle (figure 5(d)) to simulate the relative positions of the different participants. Finally, the animation parameters can be used to animate a different character (figure 5(e)).

3.2 Relighting

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 [26] tackle this problem for images of buildings taken under different illumination conditions. In our approach we did not try to recover the albedo of the face; instead, we just modulate the colors in the target image. The addition of a virtual light source is simulated as follows. First, the estimated face geometry is rendered in a grey color using the synthetic light source (figure 5(f)). Then this rendering is used to scale the colors in the original frame. Figure 5(g) shows the original frame with a dramatic change in lighting.

3.3 Adding facial decorations

The geometry and the texture of the face can be modified in the original footage. For instance, we could modify the geometry by adding a nose ring to the person in the video. This modification is done by rendering the modified face geometry with the original frame as a texture map. The texture map coordinates are computed by projecting the mesh vertices onto the image plane using the recovered camera parameters.

Modifying the face texture allows us to add decorations to the face (e.g., tattoos and scars). These modifications are introduced by rendering the recovered geometry with a cylindrical texture map bearing the decorations (figure 5(h)). The rendering is then composited over the original frame (figure 5(i)). Notice how the tattoos are deformed and occluded according to the face geometry. These effects would be impossible to obtain using a 2D tracking technique.

4 Conclusion and future work

In this paper, we have introduced a novel technique for estimating the head's position and 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. The quality of the model allows better tracking results than with other methods. We use a continuous optimization technique to fit the model and give an analytical derivation of the computations.

We have also presented a wide range of applications that demonstrate the usefulness of this technique, including simple performance-driven animations for changing viewpoint, identity, emotion, and more subtle special effects such as relighting the face or adding tattoos.

In the future, we would like to improve our system in different ways:

We used bandpass filtering to correct for color differences between the target images and the model renderings. These adjustments could also be done by adding a lighting model to our face model. The parameters of the lighting models would then be estimated along with the rest of the face parameters.

Our 3D modeling-from-images technique results in a fairly coarse approximation of the face geometry and texture. Using a 3D scanner would provide more accurate models that could substantially improve the tracking results.

Finally, we could employ more effective prediction techniques such as Kalman filtering.

As we try to build computer systems that interact with their users in a more natural fashion, analyzing face images is becoming a key issue. We believe that modeling, animating, and tracking the human face as a linear combination of sample faces is a powerful paradigm.

References

[1] K. Aizawa and H. Harashima. Model-based analysis synthesis image coding (MBASIC) system for a person's face. *Signal Processing: Image Communication*, pages 139–152, 1994.

[2] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

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

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

[5] C. Bregler, M. Covell, and M. Slaney. Video Rewrite: driving visual speech with audio. In *SIGGRAPH 97 Conference Proceedings*, pages 353–360. ACM SIGGRAPH, August 1997.

[6] C.S. Choi, K. Aizawa, H. Harashima, and T. Takebe. 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, June 1994.

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

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

[9] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Interpreting face images using active appearance models. In *Proceedings, Third Workshop on Face and Gesture Recognition*, pages 300–305. 1998.

[10] I. Essa, S. Basu, T. Darell, and A. Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. *Computer Animation Conference*, pages 68–79, June 1996.

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

[12] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH 98 Conference Proceedings*, pages 55–66. ACM SIGGRAPH, July 1998.

[13] G.D. Hager and P.N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings, Computer Vision and Pattern Recognition*, pages 403–410. 1996.

[14] M.J. Jones and T. Poggio. Hierarchical morphable models. In *Proceedings, International Conference on Computer Vision*, pages 820–826. 1998.

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

[16] A. Lanitis, C.J. Taylor, and T.F. Cootes. A unified approach for coding and interpreting face images. In *Fifth International Conference on Computer Vision (ICCV 95)*, pages 368–373. Cambridge, Massachusetts, June 1995.

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

[18] K. Matsino, C.W. Lee, S. Kimura, and S. Tsuji. Automatic recognition of human facial expressions. In *Proceedings of the*

IEEE, pages 352–359. 1995.

[19] E. Petajean and H.P. Graf. Robust face feature analysis for automatic speechreading and character animation. In *Proceedings od International Conference on Automatic Facial and Gesture Recognition*, pages 357–362. 1996.

[20] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*, pages 75–84. ACM SIGGRAPH, July 1998.

[21] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[22] A. Schodl, A. Ario, and I. Essa. Head tracking using a textured polygonal model. In *Workshop on Perceptual User Interfaces*, pages 43–48. 1998.

[23] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 569–579. june 1993.

[24] T. Vetter and V. Blanz. Estimating coloured 3D face models from single images: an example based approach. In *Proceedings, European Conference on Computer Vision*, pages 499–513. 1998.

[25] L. Williams. Performance-driven facial animation. In *SIGGRAPH 90 Conference Proceedings*, volume 24, pages 235–242. August 1990.

[26] Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In *SIGGRAPH 98 Conference Proceedings*, pages 207–217. ACM SIGGRAPH, July 1998.

Appendix

Appendix A: Analytical Jacobian

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

Partial derivatives of P

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

$$\frac{\partial P_x}{\partial \boldsymbol{m}} = \begin{pmatrix} \frac{f}{Z} \\ 0 \\ -f\frac{X}{Z^2} \end{pmatrix} \quad \text{and} \quad \frac{\partial P_y}{\partial \boldsymbol{m}} = \begin{pmatrix} 0 \\ \frac{f}{Z} \\ -f\frac{Y}{Z^2} \end{pmatrix}$$

Partial derivatives of $\boldsymbol{Rm}_j + \boldsymbol{t}$

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

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

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 $\boldsymbol{m}_j$ is a linear combination of points from the basic expressions $\boldsymbol{m}_{jk}$. We can then apply the following derivation:

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

- 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 $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$, and is given by Rodriguez's formula:

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

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

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

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

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

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

$$\boldsymbol{R} \leftarrow \hat{\boldsymbol{R}}\boldsymbol{R}$$

- 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):

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

Appendix B: Movie files

A set of mpeg files illustrating our tracking results and applications are available on-line at http://www.cs.washington.edu/research/projects/grail2/www/projects/realface/tracking.html.

Tracking results: trk1.mpg, trk2.mpg, trk3.mpg, and trk4.mpg.

Applications:

- exg3.mpg: exaggeration of facial expressions.
- chv3.mpg: change in viewpoint.
- fem2.mpg: animation transposed to different model.
- rel2.mpg: relighting of video footage.
- tat2.mpg: addition of synthetic elements.