Geometry-Driven Photorealistic Facial Expression Synthesis

Qingshan Zhang, Zicheng Liu, *Senior Member*, *IEEE*, Baining Guo, *Senior Member*, *IEEE*, Demetri Terzopoulos, *Fellow*, *IEEE*, and Heung-Yeung Shum, *Senior Member*, *IEEE*

Abstract—Expression mapping (also called performance driven animation) has been a popular method for generating facial animations. A shortcoming of this method is that it does not generate expression details such as the wrinkles due to skin deformations. In this paper, we provide a solution to this problem. We have developed a geometry-driven facial expression synthesis system. Given feature point positions (the *geometry*) of a facial expression, our system automatically synthesizes a corresponding expression image that includes photorealistic and natural looking expression details. Due to the difficulty of point tracking, the number of feature points required by the synthesis system is, in general, more than what is directly available from a performance sequence. We have developed a technique to infer the missing feature point motions from the tracked subset by using an example-based approach. Another application of our system is expression editing where the user drags feature points while the system interactively generates facial expressions with skin deformation details.

Index Terms—Facial animation, expression mapping, expression details, facial expressions, performance-driven animation.

1 Introduction

Realistic facial expression synthesis has been one of the most interesting yet difficult problems in computer graphics. There has been much research in this area, and the reader is referred to the book by Parke and Waters [21] for an excellent survey.

Expression mapping (also called performance-driven animation) [6], [15], [32], [21] has been a popular method for generating facial animations. For example, this technique was used to produce some of the facial animations in the renowned film "Tony de Peltrie." Given an image of a subject's neutral face and another image of this person's face with an expression, the positions of the facial features (eyes, eye brows, mouth, etc.) on both images are located either manually or through some automatic method. The difference vector is then added to the feature positions of a new face to generate the new expression for that face through geometry-controlled image warping [2], [15]. A shortcoming of this method, as pointed out in [16], is that it does not produce expression details such as wrinkles caused by skin deformations. The technique proposed by Liu et al. [16] requires the expression ratio image from the performer, which sometimes can be difficult to obtain.

In this paper, we propose a solution which does not require ratio images from the performer. Instead, we require a set of example expressions of the target face, which can be obtained offline. We call our system a geometry-driven facial expression synthesis system. Given the feature point positions (the *geometry*) of a facial expression, our system automatically synthesizes the corresponding expression image with photorealistic and natural looking expression details. Because the number of feature points required by the synthesis system is, in general, more than what is available from the performer due to the difficulty of tracking, we have developed a technique to infer the feature point motions from a subset of tracked points through an example-based approach. Another application of our system is expression editing, where the user drags the feature points while the system interactively generates facial expressions with skin deformation details.

An early version of this paper was presented elsewhere [34]. Here, we extend the work mainly in three areas. First, we study the performance of the motion propagation algorithm when the number of input feature points is small. Convergence measurement results are presented. Furthermore, we show side-by-side comparisons of the synthesized expressions by using a small versus a large number of input feature points. Second, we extend the previous system to allow head pose changes by the performer. We have developed techniques to estimate global motion and perform motion compensation before expression mapping. Finally, we have developed algorithms to estimate the 3D poses of the performer and map them to the target model. We show a new example of expression mapping from a live sequence with relatively large head pose changes and automatically tracked feature points. The accompanying video shows the results of the enhanced expression mapping together with head pose changes.

- *Q. Zhang is with the Research and Innovation Center, Alcatel Shanghai Bell Ltd., D301, Building 3, No. 388 Ningqiao Road, Pudong, Shanghai. E-mail: Qingshan.ZHANG@alcatel-sbell.co.cn.*
- *Z. Liu is with Microsoft Research, One Microsoft Way, Redmond, WA 98052. E-mail: zliu@microsoft.com.*
- *B. Guo and H.-Y. Shum are with Microsoft Research Asia, 6/F Beijing Sigma Center, No. 49 Zhichun Rd., Haidian District, Beijing, China 100080. E-mail: {bainguo, hshum}@microsoft.com.*
- *D. Terzopoulos is with the New York University Media Research Lab, 719 Broadway, 12th Floor, New York, NY 10003-6806. E-mail: dt@cs.nyu.edu.*

Manuscript received 3 Sept. 2004; revised 3 May 2005; accepted 13 June 2005; published online 9 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0100-0904.

The remainder of the paper is organized as follows: We review related work in Section 2. From Section 3 through Section 9, we describe our algorithm for both 2D and 3D. In Section 10, we describe the feature point inference. The enhanced expression mapping is described in Section 11. Section 12 describes motion compensation for handling head pose changes. In Section 13, we describe the expression editing application. The results are shown in Section 14. Section 15 discusses the limitations of our system. Finally, we conclude in Section 16.

2 Related Work

Facial animation has attracted considerable attention (e.g., [20], [25], [31], [18], [28], [2], [13], [22], [27], [5], [23]). In this section, we will discuss prior research that is most closely related to our work.

There has been a lot of success on speech driven facial animation [5], [4], [9], [8]. Speech driven facial animation systems are mainly concerned about the mouth region, while our method is mainly concerned with facial expressions. An interesting analogy is that speech driven animation systems use audio signals to derive mouth images, while our system uses feature point motions to derive facial images. It would be interesting to combine these two techniques to generate speech-driven facial animations with expressions.

Toelg and Poggio [29] proposed an example-based video compression architecture. They divided the face into subregions. For each subregion, they used image correlation to find the best match in the example database and send the index over the network to the receiver.

Guenter et al. [11] developed a system that uses digital cameras to capture 3D facial animations. Noh and Neumann [19] developed the expression cloning technique to map the geometric motions of one person's expression to a different person.

An effective approach to generate photorealistic facial expressions with details is the morph-based approach [2], [27], [5], [23]. In particular, Pighin et al. [23] used convex combinations of the geometries and textures of example face models to generate photorealistic facial expressions. Their system can perform both expression mapping and expression editing. For expression mapping, their system maps one person's expression to another person by transferring the convex combination coefficients. A difference between our method and theirs is that we do not require example expressions from the performer. For expression editing, their system provides a set of easy-to-use tools and interfaces to allow a user to design facial expressions interactively. Their system was mainly designed for offline authoring and it requires a user to specify blending weights manually in order to obtain a desired expression. By contrast, our system automatically computes the blending weights. Furthermore, we have developed a technique to infer the feature point motions from a subset. By combining these two techniques, we can enhance the traditional expression mapping system with expression details. In another paper, Pighin et al. [24] used the expression morphing model to reconstruct 3D expressions from a video sequence. Their system did not try to map the facial expressions to a different face model.

Liu et al. [16] proposed a technique, called the expression ratio image, to map one person's expression details to a different person's face. In addition to feature point motions, their method requires an expression ratio image from the performer. In contrast, our method requires only the motions of feature points from the performer. In situations where the feature point positions of an expression are given, but no expression ratio images are available for this geometry, our method is more useful. For example, when manipulating the facial feature points in expression editing applications, the user is unlikely to find an image of a different person with exactly the same expression. In expression mapping applications, if the performer has markers on his/her face or if there are lighting variations due to head pose changes, the ratio images may be difficult to create.

Joshi et al. [12] developed a data-driven approach to segment a face into subregions thus providing local control to the user. A difference between their work and ours is that we automatically propagate motion to different levels of detail, while their technique requires the user to specify explicitly the segmentation threshold which determines the level of control. When applied to expression mapping, automatic motion propagation is critical.

Pyun et al. [26] proposed a different expression mapping approach where both source and target expressions are parameterized based on examples, and the expression mapping is performed by transferring the parameters of the source expressions. Compared to their technique, our approach does not require example expressions from the source.

Blanz et al. [3] developed a system to create 3D animations from a single face image or a video. Their system automatically estimates the 3D face mesh, pose, and lighting parameters from a single image. It then transfers a 3D facial expression of a different person to the reconstructed face model by mapping the geometric difference vectors.

3 Geometry-Driven Expression Synthesis

Given the feature point positions of a facial expression, one possibility to compute the corresponding expression image would be to use some mechanism, such as physical simulation [13], to determine the geometric deformations for each point on the face and, then, render the resulting surface. The problem is that it is difficult to model detailed skin deformations such as expression wrinkles, and it is also difficult to render a face model so that it looks photorealistic. We instead take an example-based approach.

Pighin et al. [23] demonstrated that one can generate photorealistic facial expressions through a convex combination of example expressions. Let $E_i = (G_i, I_i)$, $i = 0, \ldots, m$, denote a set of example expressions where G_i represents the geometry and I_i is the texture image. We assume that all the texture images I_i are pixel aligned. Let $H(E_0, E_1, \ldots, E_m)$ be the space of all possible convex combinations of these examples, i.e.,

$$H(E_0, E_1, \ldots, E_m) = \left\{ \left(\sum_{i=0}^{m} c_i G_i, \sum_{i=0}^{m} c_i I_i \right) \mid \sum_{i=0}^{m} c_i = 1,\ c_0, \ldots, c_m \geq 0 \right\}. \quad (1)$$

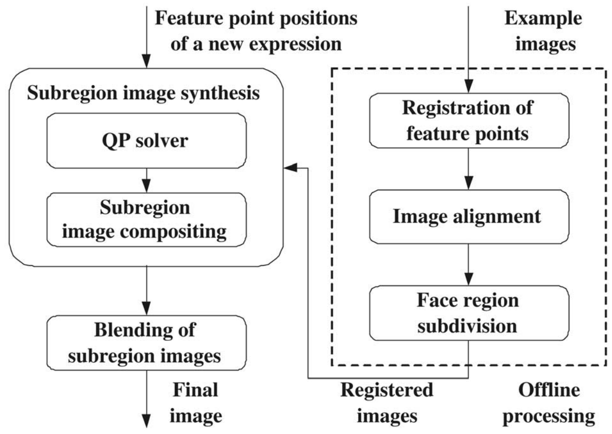

Feature point positions of a new expression
Example images
Subregion image synthesis
QP solver
Subregion image compositing
Registration of feature points
Image alignment
Face region subdivision
Blending of subregion images
Final image
Registered images
Offline processing

Fig. 1. An overview of the geometry-driven expression synthesis system.

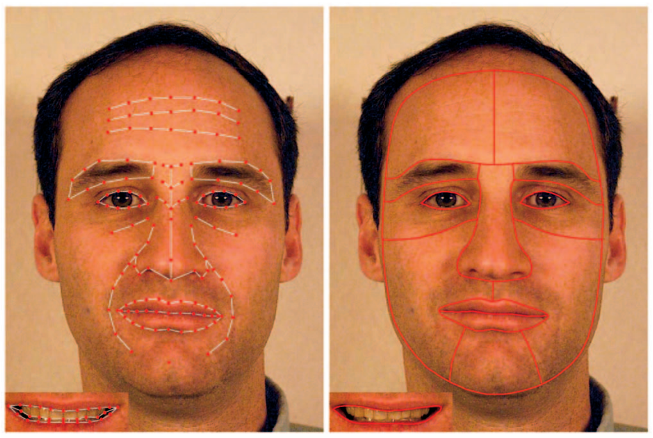

(a) (b)

Fig. 2. (a) Feature points. (b) Face region subdivision.

Note that each expression in $H(E_0, E_1, \ldots, E_m)$ has a geometric component $G = \sum_{i=0}^{m} c_i G_i$ and a texture component $I = \sum_{i=0}^{m} c_i I_i$. Since the geometric component is much easier to obtain than the texture component, we propose using the geometric component to infer the texture component. One simple idea is to first project a geometric component G to the convex hull spanned by $G_0, \ldots, G_m$ and, then, to use the resulting coefficients to composite the example images and obtain the desired texture image.

A problem with this approach is that the space of $H(E_0, E_1, \ldots, E_m)$ is very limited. A person can have expression wrinkles in different facial regions, and the combinatorics become intractable. So, we subdivide the face into a number of subregions. For each subregion, we use the associated geometry to compute the subregion's texture image. The subregion images are then seamlessly blended together to produce the final expression image.

We first describe our algorithm for 2D cases where the geometry of an expression comprises the facial feature points projected on the image plane. In Section 9, we extend the algorithm to 3D.

4 System Overview

Fig. 1 presents an overview of our system. It consists of an offline processing unit and a runtime unit. The example images are processed offline only once. At runtime, the system takes as input the feature point positions of a new expression and produces the final expression image. In the following sections, we describe each of the function blocks in more detail.

5 Offline Processing of the Example Images

5.1 Feature Points

Fig. 2a shows the feature points that we use in our system. At the bottom left corner are the feature points of the teeth area when the mouth is open. There are 134 feature points in total. Given a face image, it is possible to compute face features automatically [14] since the number of example images is small in our system (10 to 15 examples per person). We choose to mark the feature points of the example images manually.

5.2 Image Alignment

After we obtain the markers of the feature points, we align all the example images with a standard image which is shown in Fig. 3a. The reason to create this standard image is that we need to have the mouth open so that we can obtain the texture for the teeth. The alignment is done by using a simple triangulation-based image warping, although more advanced techniques [2], [15] may be used to obtain better image quality.

5.3 Face Region Subdivision

We divide the face region into 14 subregions. Fig. 2b shows the subdivision, which is designed manually. At the lower left corner of the figure is the teeth subregion when the mouth is open. The guideline of our subdivision scheme is that we would like the subregions to be small while avoiding expression wrinkles crossing the subregion boundaries. Since we have already aligned all the example images with the standard image, we only need to subdivide the standard image. We create an image mask to store the

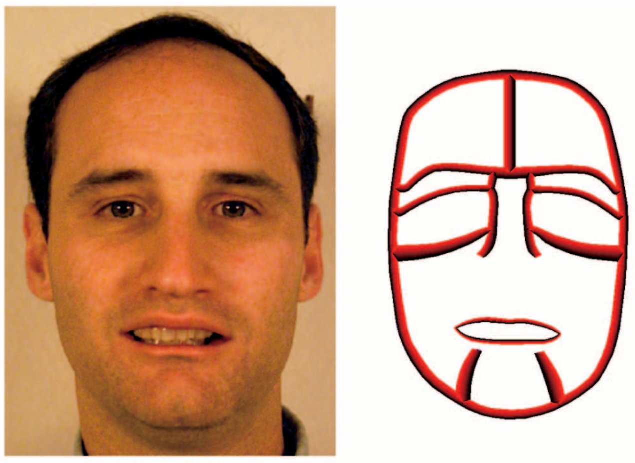

(a) (b)

Fig. 3. (a) The standard image. (b) The weight map for blending along the subregion boundaries.

subdivision information, where the subregion index for each pixel is stored in its color channel.

6 Subregion Expression Synthesis

For each subregion R and example expression E_i, let G_i^R denote the vector of E_i's feature point positions that are within or on the boundary of R. Let G^R denote the new expression's feature point positions that are within or on the boundary of R.

Given G^R, we want to project it into the convex hull of $G_0^R, \dots, G_m^R$. In other words, we want to find the closest point in the convex hull. This can be formulated as an optimization problem:

$$\begin{aligned}&\text{Minimize :} \quad \left(G^R - \sum_{i=0}^{m} c_i G_i^R\right)^T \left(G^R - \sum_{i=0}^{m} c_i G_i^R\right),\\ &\text{Subject to :} \quad \sum_{i=0}^{m} c_i = 1,\ c_i \geq 0 \text{ for } i = 0, 1, \dots, m,\end{aligned} \tag{2}$$

where $c_0, c_1, \dots, c_m$ are the convex combination coefficients for which we must solve. Let

$$\mathcal{G} = (G_0^R, G_1^R, \dots, G_m^R) \tag{3}$$

and

$$C = (c_0, c_1, \dots, c_m)^T. \tag{4}$$

Then, the objective function becomes

$$C^T \mathcal{G}^T \mathcal{G} C - 2 G^{R^T} \mathcal{G} C + G^{R^T} G^R. \tag{5}$$

This is a quadratic programming problem where the objective function is a positive semidefinite quadratic form and the constraints are linear. Since the G_i^R are, in general, linearly independent, the objective function is, in general, positive definite.

There are several ways to solve a quadratic programming problem [17], [33]. In the past decade, a lot of progress has been made on interior-point methods both in theory and in practice [33]. Interior-point methods have become very popular for solving many practical quadratic programming problems. This is the approach that we take. An interior point method works by iterating in the interior of the domain which is constrained by the inequality constraints. At each iteration, it uses an extension of Newton's method to find the next feasible point which is closer to the optimum. Compared to traditional approaches, interior point methods have a faster convergence rate both theoretically and in practice, and they are numerically stable. Even though an interior point method usually does not produce an optimal solution (it yields an interior point), the solution is, in general, very close to the optimum. In our experiments, we find that it works very well for our purpose.

After we obtain the coefficients c_i, we compute the subregion image I^R by compositing the example images together:

$$I^R = \sum_{i=0}^{m} c_i I_i^R. \tag{6}$$

Note that, since the example images have already been aligned, this step is simply pixel-wise color blending.

7 Blending along the Subregion Boundaries

To avoid image discontinuities along the subregion boundaries, we do a fade-in-fade-out blending along the subregion boundaries. In our implementation, we use a weight map to facilitate the blending. Fig. 3b shows the weight map, which is aligned with the standard image (Fig. 3a). The thick red-black curves are the blending regions along the boundary curves. The intensity of the R-channel stores the blending weight. We use the G and B channels to store the indices of the two neighboring subregions, respectively. Given a pixel in the blending region, let r denote the value in the R-channel, and let i_1 and i_2 be the indices of the two subregions. Then, the pixel's blended intensity is

$$I = \frac{r}{255} * I^{i_1} + (1 - \frac{r}{255}) * I^{i_2}. \tag{7}$$

Note that we do not perform blending along some of the boundaries where there is a natural color discontinuity, such as the boundary of the eyes and the outer boundary of the lips.

After blending, we obtain an image which is aligned with the standard image. We then warp the image back so that its feature point positions match the input feature point positions, thus obtaining the final expression image.

8 Teeth

Since the teeth region is special, we use a separate set of examples for the teeth region (see Fig. 8). In our current system, only a small set of examples for the teeth region are used since we are not focusing on speech animations where there are a lot of variations in mouth shapes.

9 Expression Synthesis in 3D

It is straightforward to extend the algorithm to 3D where the feature points are 3D positions and the expressions are 3D meshes with or without texture maps. To compute the subregion blending coefficients, we use (3) in the same manner as before except that G and G_i are $3n$-dimensional vectors. We use the same interior point method to solve the quadratic programming problem. The subregion mesh compositing and blending along subregion boundaries are similar to the 2D case, except that we blend the 3D vertex positions instead of the images.

10 Inferring Feature Point Motions from a Subset

In practice, it is difficult to obtain all the feature points shown in Fig. 2. For example, most of the algorithms for tracking facial features only track a limited number of features along the eyebrows, eyes, mouth, and nose. In the enhanced expression mapping example that we will discuss

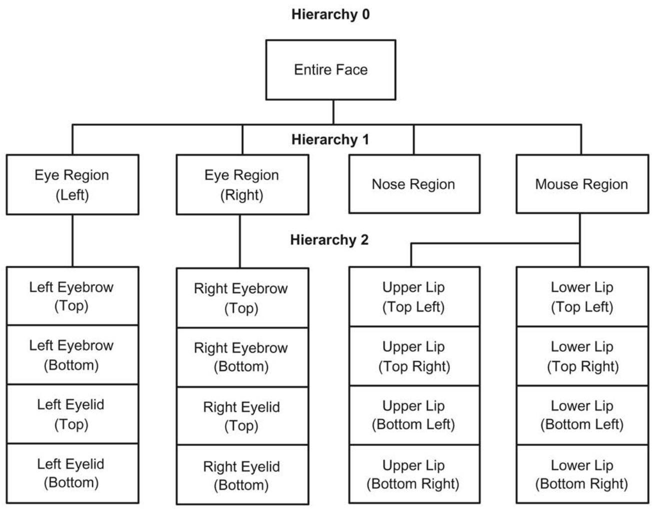


Fig. 4. Feature point sets.

later in the paper, we only extract 40 feature points from the performer. For the expression editing application that will be discussed in Section 13, each time a user moves a feature point, we need to determine the most likely movement for the remaining feature points.

In this section, we explain how to infer the motions for all the feature points from a subset. We take an example-based approach. The basic idea is to learn how the rest of the feature points move from the examples. In order to have fine-grain control, which is particularly important if the motions of only a very small number of feature points are available such as in expression editing, we divide the face feature points into hierarchies and perform hierarchical principal components analysis (PCA) on the example expressions.

There are, in total, 21 feature point sets, with a single feature point set in hierarchy 0, four sets in hierarchy 1, and 16 sets in hierarchy 2 (see Fig. 4 for descriptions of all the feature point sets).

The single feature point set at hierarchy 0 controls the global movement of the entire face. The feature point sets at hierarchy 1 control the local movement of facial feature regions (left eye region, right eye region, nose region, and mouth region). Each feature point set at hierarchy 2 controls details of the face regions, such as eyelid shape, lip line shape, etc. Some facial feature points belong to several sets at different hierarchies, and they are used as bridges between the global and local movement of the face so that we can propagate vertex movements from one hierarchy to another.

For each feature point set, we compute the displacement of all the vertices belonging to this feature set for each example expression. We then perform principal components analysis on the vertex displacement vectors corresponding to the example expressions and generate a lower-dimensional vector space.

10.1 Motion Propagation

In this section, we describe how to use the result of hierarchical principal components analysis to propagate facial motions, so that, from the motions of a subset of the facial feature points, we can infer reasonable movements for the remainder of the feature points. The basic idea is to learn from examples how to estimate the unknown feature point motions. To this end, we use the subspace generated by the principal components, which spans the most "common" feature point motions. Given the motions of a subset of the feature points, we first make a trivial guess about the motions of the remaining feature points (for example, by setting them to zero). We then find the closest point to this approximation in the subspace generated by the principal components. This will provide a better estimate for the motion of the remaining feature points. We iterate this process to improve the approximation. A more detailed description follows.

Let $v_1, v_2, \ldots, v_n$ denote all the feature points on the face. Let δV denote the displacement vector of all the feature points. For any given δV and a feature point set F (the set of indices of the feature points belonging to this feature point set), we use $\delta V(F)$ to denote the subvector of those vertices that belong to F. Let $Proj(\delta V, F)$ denote the projection of $\delta V(F)$ into the subspace spanned by the principal components corresponding to F. In other words, $Proj(\delta V, F)$ is the best approximation of $\delta V(F)$ in the expression subspace. Given δV and $Proj(\delta V, F)$, we say δV is *updated* by $Proj(\delta V, F)$ if, for each vertex that belongs to F, we replace its displacement in δV with its corresponding value in $Proj(\delta V, F)$.

The motion propagation algorithm takes as input the displacement vector for a subset of the feature points, say, $\Delta v_{i_1}, \Delta v_{i_2}, \ldots, \Delta v_{i_k}$. Let T be the set of feature point indices, that is, $T = \{i_1, i_2, \ldots, i_k\}$. The motion propagation algorithm is as follows:

```
**MotionPropagation**
Begin
    Set $\delta V = 0$
    While (stop criterion is not met) Do
        For each $i_k \in T$, set $\delta V(i_k) = \Delta v_{i_k}$
        For all Feature point sets $F$, set
            $hasBeenProcessed(F)$ to be false
        Find the feature point set $F$ in the lowest
            hierarchy such that $F \cap T \neq \emptyset$
        $MotionPropagationFeaturePointSet(F)$
    End
End
```

where the function $MotionPropagationFeaturePointSet$ is defined as follows:

```
**MotionPropagationFeaturePointSet**$(F^*)$
Begin
    Set $h$ to be the hierarchy of $F^*$
    If $hasBeenProcessed(F^*)$ is true, return
    Compute $Proj(\delta V, F^*)$
    Update $\delta V$ with $Proj(\delta V, F^*)$
    Set $hasBeenProcessed(F^*)$ to true
    For each feature set $F$ belonging to hierarchy $h - 1$
        such that $F \cap F^* \neq \emptyset$
            $MotionPropagationFeaturePointSet(F)$
    For each feature set $F$ belonging to hierarchy $h + 1$
        such that $F \cap F^* \neq \emptyset$
            $MotionPropagationFeaturePointSet(F)$
End
```

The algorithm first initializes δV to a zero vector. At the first iteration, it first sets $\delta V(i_k)$ to be equal to the input displacement vector for vertex v_{i_k}. It then finds the feature point set in the lowest hierarchy that intersects with the input feature point set T and invokes $MotionPropagation FeaturePointSet$. The function first uses PCA to infer the motions for the rest of the vertices in this feature point set. It then recursively calls $MotionPropagationFeaturePointSet$ on other feature point sets. To make sure that the function $MotionPropagationFeaturePointSet$ is applied to each feature point set at most once, we maintain a flag named $hasBeenProcessed$. This flag is initialized to "false" for all the feature point sets at the beginning of the iteration. At the end of the first iteration, δV contains the inferred displacement vectors for all the feature points. Note that, for the vertex in T, its inferred displacement vector may be different from the input displacement vector because of the PCA projection. At the second iteration, we reset $\delta V(i_k)$ to be equal to the input displacement vector for all $i_k \in T$. For the remaining feature points that are not in T, we use their displacement vectors resulting from the previous iteration. We then repeat the process. The result is a new δV where the displacement vectors for the vertices in T are closer to the input displacement vectors. This process continues until the stop criterion is satisfied. In our implementation, we use the progress on the resulting displacement vectors in T as the stop criterion. If the average progress on two consecutive iterations is less than a user-specified threshold, the algorithm stops.

11 Enhanced Expression Mapping

As we mentioned earlier, the traditional expression mapping technique has the drawback that the resulting facial expressions may not look convincing due to the lack of expression details. Our technique provides a solution to this problem in the case where we can obtain example images for the new subject. The example images may be acquired offline or designed by an artist. At runtime, we first use the geometric difference vector to obtain the desired geometry for the new subject as in the traditional expression mapping system. Because of the difficulty of face tracking, the number of available feature points is in general much smaller than the number of feature points needed by the synthesis system. So, we use the technique of Section 10 to infer the motions for all the feature points used by the synthesis system. We then apply our synthesis technique to generate the texture image based on the geometry. The end result is more convincing and realistic facial expressions.

12 Motion Compensation

When the performer talks or makes expressions, in addition to the nonrigid skin deformations, there is a global rigid motion of the head. If we directly compute the vertex differences between the expression face and the neutral face, the resulting vertex motions include both the skin deformations and the global head motion. The global head motion is undesirable because it may result in unnatural expression details when input to the expression synthesis system. In this section, we describe techniques to remove the global head motions.

Our approach is first to estimate the motion and then to perform motion compensation in order to undo the global motion. Since the examples we are working with do not contain large out-of-plane rotations, we choose to use a 2D similarity motion model. In our experiments, we find that it is more stable than the 2D affine transformation model. We have also implemented a 3D motion estimation algorithm, which we will describe in the next section. Due to a limited amount of information on the 3D structure and the nonrigid motion of the feature points, the estimated 3D motions are not as accurate as the 2D motions. Therefore, we only use the 3D motions to synthesize the head motion for the target face model.

Techniques for 2D similarity motion estimation can be found in the literature, such as the one in [7], which we briefly describe next. A 2D similarity transformation $T(a, b, t_x, t_y)$ consists of a translation, rotation, and scaling. It is defined as follows:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}. \tag{8}$$

Suppose there are n feature points. Let $G = (x_1, y_1, x_2, y_2, \dots, x_n, y_n)^T$ be the vector of all the feature point positions on the neutral face image and $G' = (x'_1, y'_1, x'_2, y'_2, \dots, x'_n, y'_n)^T$ be the vector of all the feature point positions on an expression image. The goal is to find the quantities a, b, t_x, and t_y that minimize

$$\sum_{i=1}^{n} \left\| T\begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} \right\|^2. \tag{9}$$

We refer the reader to [7] for the solution of this problem.

After motion estimation, we apply the inverse of T to the feature points on the expression face $\begin{pmatrix} x'_i \\ y'_i \end{pmatrix}$ to remove the global motion, and we input the displacement vector

$$T^{-1}\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

to the expression synthesis system.

12.1 3D Head Pose Estimation

Facial animation with a fixed head pose does not look natural. One way to generate natural looking head motion is to map the performer's head motions to the target model. In this section, we describe our technique for estimating 3D head poses from the tracked 2D feature points in the performer's video sequence. We pursue a model-based approach. We use a generic face model, as shown in Fig. 5, as an approximation of the performer's neutral face. First, let us describe how to estimate the head pose under the assumption that there are no facial deformations.

Let $\mathbf{m}_i = (u_i, v_i)^T$ be the feature points on an expressive face image. Let $\mathbf{p}_i = (x_i, y_i, z_i)^T$ be the corresponding 3D points on the generic face model (Fig. 5). Let $\tilde{\mathbf{m}}_i = (u_i, v_i, 1)^T$ denote the homogeneous coordinates of $\mathbf{m}_i$, and $\tilde{\mathbf{p}}_i$ denote the homogeneous coordinates of $\mathbf{p}_i$. Assuming a

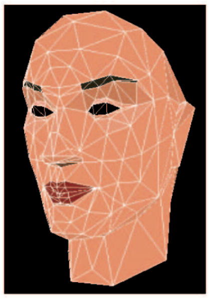


Fig. 5. The generic face model used for 3D pose estimation.

pinhole camera model, the 3D point $\mathbf{p}_i$ and its 2D image point $\mathbf{m}_i$ are related by

$$\lambda_i \tilde{\mathbf{m}}_i = \mathbf{A}\mathbf{P}\Omega\tilde{\mathbf{p}}_i, \tag{10}$$

where λ_i is a scale, and $\mathbf{A}$, $\mathbf{P}$, and Ω are given by

$$\mathbf{A} = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \Omega = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}.$$

The elements of matrix $\mathbf{A}$ are the intrinsic parameters of the camera. Matrix $\mathbf{P}$ is the perspective projection matrix. Matrix Ω is the 3D head pose (with rotation $\mathbf{R}$ and translation $\mathbf{t}$).

The intrinsic parameters can be obtained by using a simple camera calibration procedure [35]. For each point, (10) contains two constraints for Ω. We use a technique described in [10] to solve for Ω.

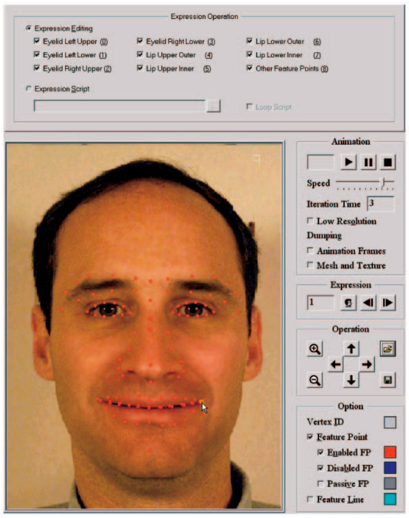
When there are facial deformations, the motion of each point contains both the rigid transformation and the local deformation. In our experiment, we select a subset of the feature points which do not have large local deformations including the five points on the boundary between the forehead and the hair, and the two eye corner points which are close to the nose bridge (see Fig. 16). Only these points are used in (10) to solve for the head pose Ω.

13 Expression Editing

Another interesting application of our technique is interactive expression editing. A common approach for designing facial expressions is to allow a user to modify control point positions or muscle forces interactively. The images are then warped accordingly. Our technique can be used to enhance such systems to generate expression details interactively.

We have developed a system that allows a user to drag a facial feature point, and the system interactively displays the resulting image with expression details. Fig. 6 is a snapshot of the expression editing interface, where the red dots are the feature points upon which the user can click and drag using the mouse.

The first stage of the system is a geometry generator. When the user drags a feature point, the geometry generator infers the "most likely" positions for all the feature points by using the algorithm described in Section 10. For example, if a user



[✓ Expression Editing]
[✓ Eyelid Left Upper (0)] [✓ Eyelid Right Lower (3)] [✓ Lip Lower Outer (6)]
[✓ Eyelid Left Lower (1)] [✓ Lip Upper Outer (4)] [✓ Lip Lower Inner (7)]
[✓ Eyelid Right Upper (2)] [✓ Lip Upper Inner (5)] [✓ Other Feature Points (8)]
Expression Script
Loop Script
Speed
Iteration Time 3
Low Resolution
Dumping
Animation Frames
Mesh and Texture
Expression 1
Vertex ID ⬜
[✓ Feature Point]
[✓ Enabled FP] 🟥
[✓ Disabled FP] 🟦
Passive FP ⬛
Feature Line 🟦

Fig. 6. The expression editing interface. The red dots are the feature points upon which a user can click and drag.

drags the feature point on the tip of the nose, the entire nose region will move instead of just this single point.

We typically use 30-40 example expressions for feature point inference in both the expression editing and expression mapping applications.

14 Results

We now show some experimental results for two faces: a male subject and a female subject. For each subject, we capture about 30-40 images of whatever expressions they can make. We then select the example images (see Figs. 7 and 9) based on the intuitive criteria that the example expressions should be sufficiently different from each other while covering all the expressions in the captured images. The rest of the images are used as ground truth for validating our system. For motion propagation (Section 10), we use all the captured images (30-40 images) to perform hierarchical PCA. For each feature point set, we typically keep 5-10 principal components.

Fig. 7 shows the selected example images for the male subject. The teeth examples are shown in Fig. 8. Fig. 10 is a side-by-side comparison, where the images on the left column are ground truth while the images on the right are the synthesized results. Note that, although each of the expressions in Fig. 10 is different from the expressions in the examples, the results from our system closely match the ground truth images. There is slight blurriness in the synthesized images because of the pixel misalignment resulting from the image warping process.

Fig. 9 shows the selected example images of the female subject. Fig. 11 is the side-by-side comparison for the female subject where the ground truth images are on the left while the synthesized images are on the right. Again, the synthesized results match very well with the ground truth images.

To test the performance of our motion propagation algorithm, we chose 25 different expressions from the male

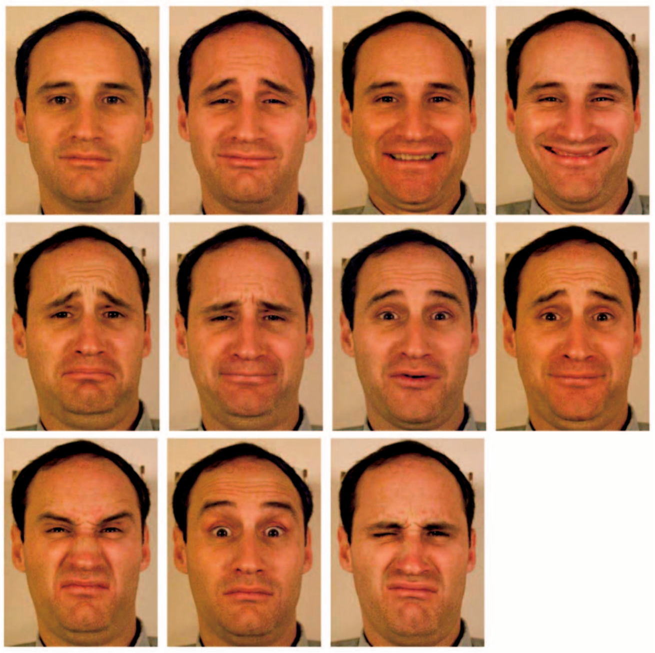


Fig. 7. The example images of the male subject.

actor and selected only 17 feature points from the original 134 marked points as the input of the motion propagation algorithm to infer the motions for all of the 134 feature points (which are required by the expression synthesis system). Fig. 12 shows the convergence of the algorithm. The horizontal axis indicates the number of iterations. The vertical axis indicates the average error in pixels for the 17 feature points and all 25 expressions. We can see that the error decreases quickly, closely approaching its optimum after 50 iterations.

Fig. 13 compares the expression synthesis results with only 17 points (right) and the results with all the 134 points (left). The results are very close. The main difference is in the mouth region where we do not have enough examples to cover the space of all possible mouth shapes.

14.1 Results of Enhanced Expression Mapping

In this section, we show the results of expression mapping enhanced with our facial expression synthesis system. Fig. 14 shows examples of mapping the female subject's

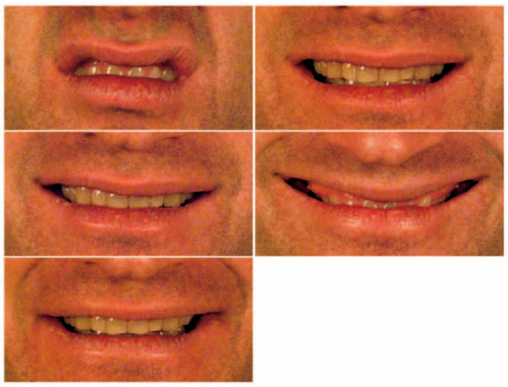


Fig. 8. The example images of the male subject's teeth.

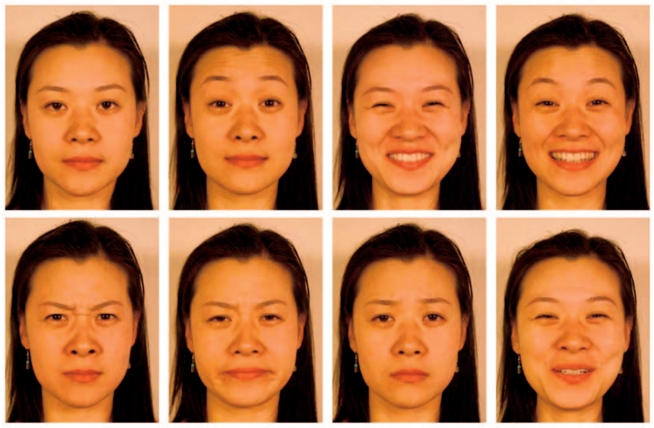


Fig. 9. The example images of the female subject.

expressions to the male subject. The female's expressions are the real data. The male's images on the right result from the enhanced expression mapping. We can see that the synthesized images have natural looking expression details.

To test the frame-to-frame coherence of our expression synthesis system, we have experimented with both synthetic image sequences and live image sequences. In the accompanying video, we show an expression sequence where the geometry of each frame is a linear interpolation of a few key expressions. For each frame, we independently synthesize the expression image from the geometry of that frame. We can see that the resulting expression sequence is very smooth from frame to frame.

We have captured a few live sequences for the male subject. For each live sequence, we manually extract about 40 feature points in each of the frames. For each frame, we infer the positions for the remaining feature points and then use our expression synthesis system to produce the expression image. The accompanying video (which can be found at http://www.computer.org/tvcg/archives.htm) shows both the real sequences and the synthesis results.

The accompanying video also shows the results of mapping the live sequences of the male subject to the female subject. Again, 40 feature points are used.

We have captured a 3D face model of the male subject by using a laser scanner. We then use the feature point motions of the male subject to drive the vertex movement of the 3D mesh. This is done by using a simple triangulation-based interpolation. For each frame, we use the synthesized expression image as the texture map for the 3D mesh. Since there are almost no head motions in the captured video, we manually add head rotations through simple keyframing. The accompanying video shows the results.

In the final example of expression mapping, we map a live sequence of a different male subject's expression sequence to the first male actor. This sequence has relatively large head pose motions, and its feature points are tracked automatically by using a correlation-based tracking technique [1]. Due to the difficulty of automatic tracking, we tracked only 27 interior feature points. Fig. 16 shows all the tracked points. The accompanying video shows the live sequence together with the tracked points. We can see that the tracking result is very noisy, so we first smooth each feature point trajectory by using a Gaussian kernel. Because

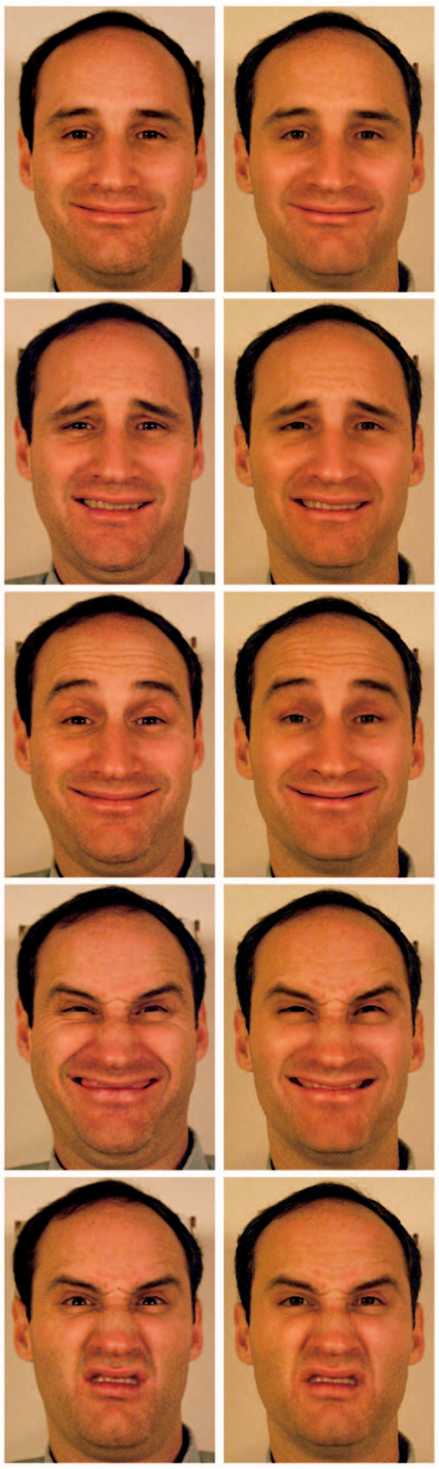


Fig. 10. Side-by-side comparison with ground truth for the male subject. In each image pair, the left image is the ground truth and the right image is the synthesis result.

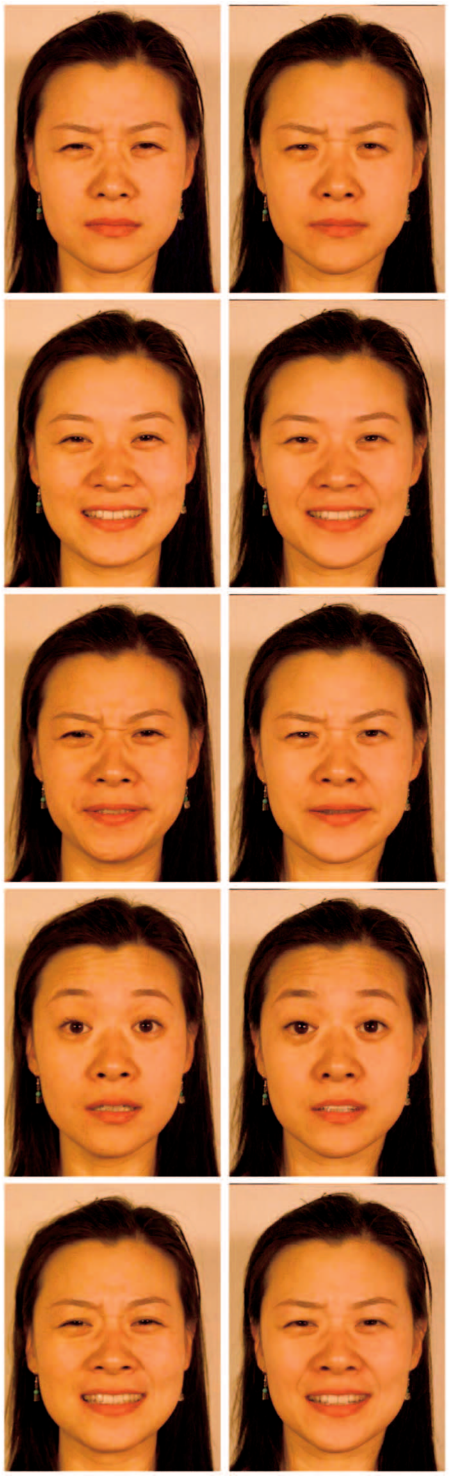


Fig. 11. Side-by-side comparison with the ground truth of the female subject. In each image pair, the left image is the ground truth and the right image is the synthesis result.

of the large head pose changes, we must apply the motion compensation algorithm to undo the head motion before performing expression mapping. After motion compensation, we first apply traditional expression mapping (directly mapping the vertex displacement) to map his motion to the first male actor's face (Fig. 7). From the result, which is shown in the accompanying video, we can see that there are many artifacts due to the limited number of feature points and the inaccuracy of tracking. We then apply our motion propagation algorithm to infer the motions for all 134 feature points and input them to our expression synthesis system. The accompanying video shows the synthesized result, which has photorealistic expression details. Finally, we estimate the 3D head poses from the input sequence and apply both the synthesized expressions and the 3D head

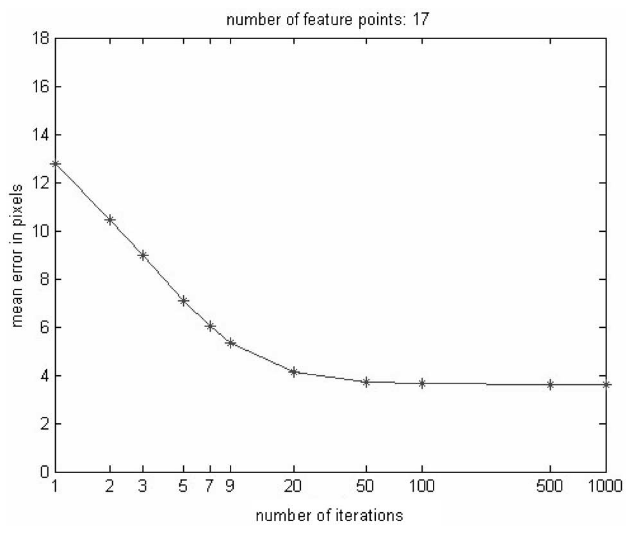


Fig. 12. The convergence of the motion propagation algorithm with only 17 points as the input while the total number of feature points is 134. The horizontal axis indicates the number of iterations of the motion propagation algorithm. The vertical axis indicates the average error in pixels. The image size is $750 \times 1,000$ pixels.

poses to the male actor's 3D model. The accompanying video shows the final result.

14.2 Results of Expression Editing

Fig. 15 shows some of the expressions generated by our expression editing system. Note that each of these expressions has a different geometry than any of the example images. Our system is able to produce photorealistic and convincing facial expressions.

We have also tested our system with 3D expression synthesis. We asked an artist to create a set of 3D facial expressions as shown in Fig. 17. Fig. 18 shows some of the synthesized expressions in different poses.

Finally, the accompanying video shows expression editing in action. The sizes of the images used in our experiments are 600×800 pixels. Our current system achieves a frame rate of two-four frames per second on a 2GHz PC. Because the frame rate is not high enough, we do not perform synthesis until the mouse stops moving. When the mouse stops moving, we sample five frames between the previous mouse position and the current mouse position, and synthesize the expression images for each frame, and display them on the large window on the left. At the same time, we update the image in the small window. The main computation cost is the image compositing. Currently, the image compositing is done in software and, for every pixel, we perform the compositing operation for all the example images, even though some of the example images have coefficients close to zero. One way to increase the frame rate is not to composite those example images whose coefficients are close to zero. Another way is to use hardware acceleration. We plan to explore both approaches toward improving the performance.

15 Limitations

A limitation of our system is the lack of extrapolation. For each subregion, the texture images that our system can

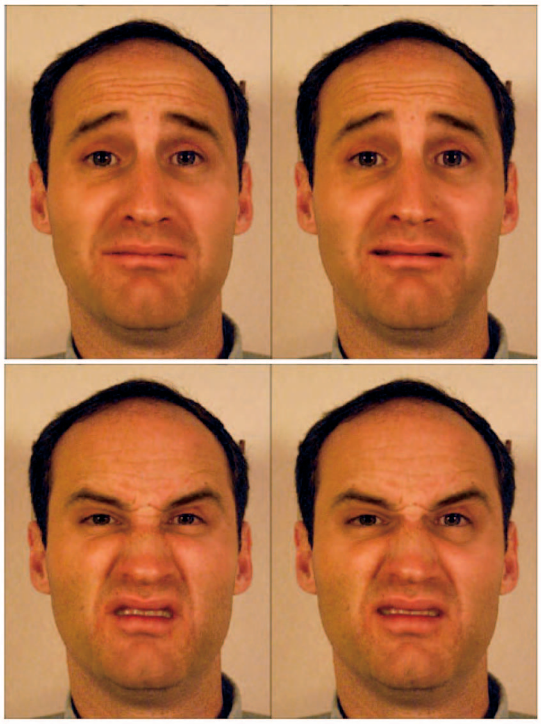


Fig. 13. The left image of each pair shows synthesized expressions resulting from all the feature points input directly to the expression synthesis system. The right image of each pair shows synthesized expressions resulting from only 17 feature points input to the motion propagation algorithm in order to infer the motions of all the feature points, which are then input to the expression synthesis system.

potentially generate are limited to the convex hull of the example images. If the example expressions are not very expressive, the generated expressions will be commensurately unexpressive. For example, in the data we captured, the expressions of the female subject are not as expressive as those of the male subject. As shown in the accompanying video, when we map the male subject's video sequence to the female subject, the resulting video for the female is less expressive than that of the male. Extrapolation in the texture space is a difficult problem. One possibility is to estimate a normal map as in [30] and extrapolate the normal.

Sometimes the artifacts due to image blending and pixel misalignment make the animation look unnatural. For example, if we zoom in on the eyebrow area of the female subject's morphing sequence, it appears that the pixels are crawling and the skin movement has a look of stretching rubber. One possibility is to refine the pixel alignment through image matching such as by applying the optical flow technique. We can improve the image blending by using the band-pass decomposition technique as in [12].

Another limitation is that our system does not handle out-of-plane head rotations (i.e., when the head rotates from one side to the other) because we use only a 2D motion model. With a 2D motion model, the out-of-plane rotations are partially accounted for by the scaling transformations. The unaccounted motions become local deformations resulting in unnatural expressions. On the other hand, the 3D head pose estimation technique described in Section 12.1 is not accurate enough for motion compensation. There are more sophisticated techniques in computer vision for more accurate head

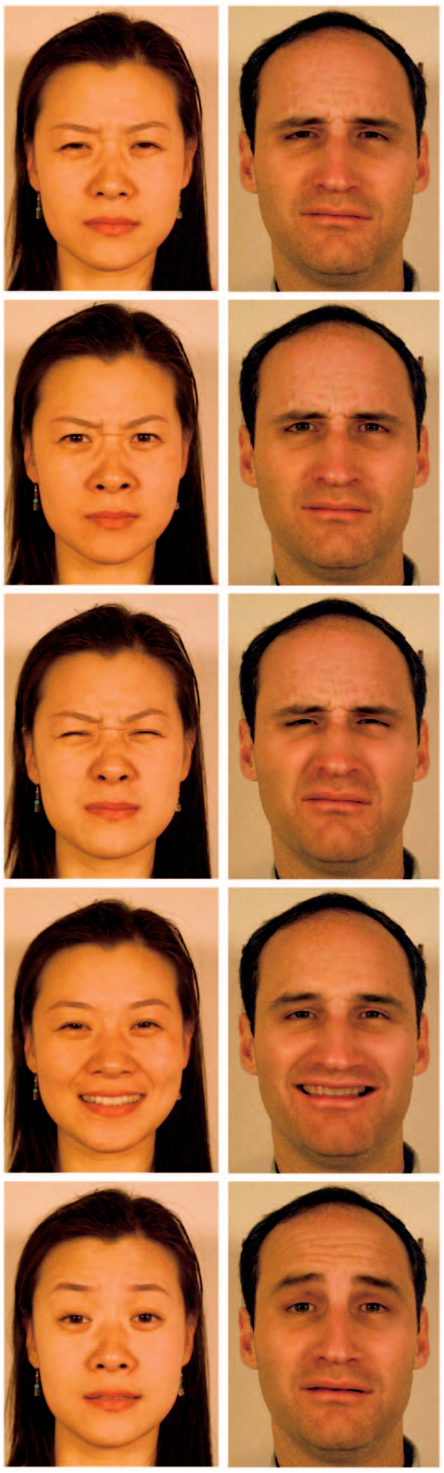


Fig. 14. Results of enhanced expression mapping. The expressions of the female subject are mapped to the male subject.

motion estimation. We would like to develop such a system and integrate it with our expression synthesis system.

In order to generate speech animations, we need a lot more example images of mouth shapes. It would be tedious to select example images of mouth shapes manually, because the amount of captured data will likely be quite large compared to the expression data. One solution is to use the technique presented in [9] to select example images automatically.

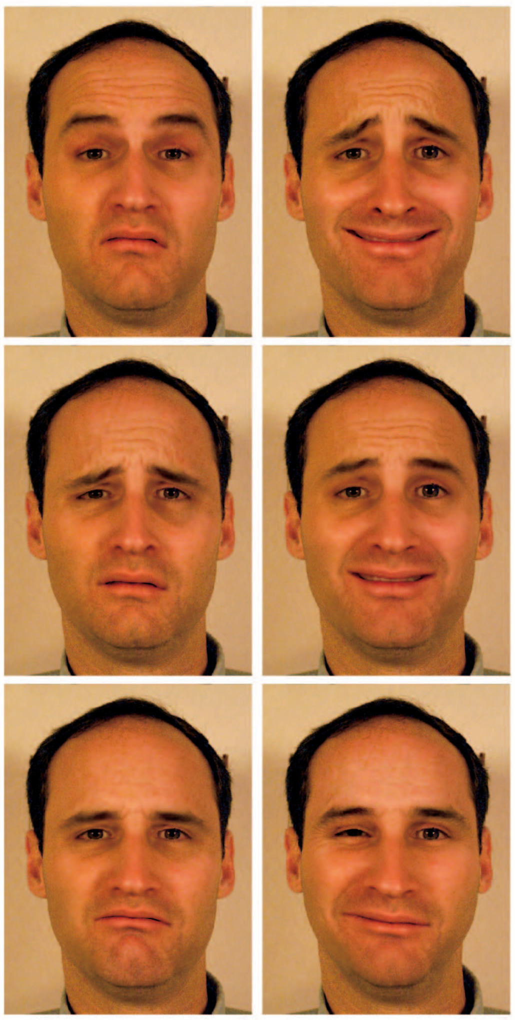


Fig. 15. Expressions generated by the expression editing system.

16 Conclusion

We have presented a geometry-driven facial expression synthesis system. Our contributions include: 1) a framework for synthesizing facial expression textures from facial feature point motions, and 2) a motion propagation algorithm to infer facial feature point motions from a small subset of tracked points. The combination of these two techniques allows us to enhance traditional expression mapping to generate facial expression details. We also demonstrated expression editing, where the user can interactively manipulate the geometric positions of the feature points and see the resulting realistic looking facial expressions. We showed that the technique works for both 2D and 3D face models.

Acknowledgments

The authors thank Ying Song, who implemented the geometric deformation system for the scanned 3D head model. They are grateful to Yangzhou Du for implementing the motion estimation algorithms and generating some of the expression mapping results. They would also like to thank Bo Zhang for helping with the video editing, Jianghong Li and Stephen Dahl for helping with the data

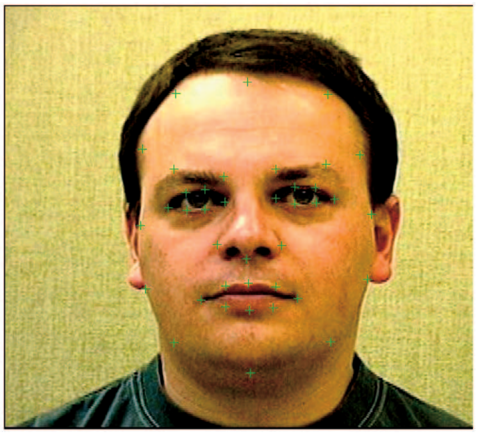


Fig. 16. A snapshot from the live sequence of a different male subject, where the feature points shown in green are tracked automatically.

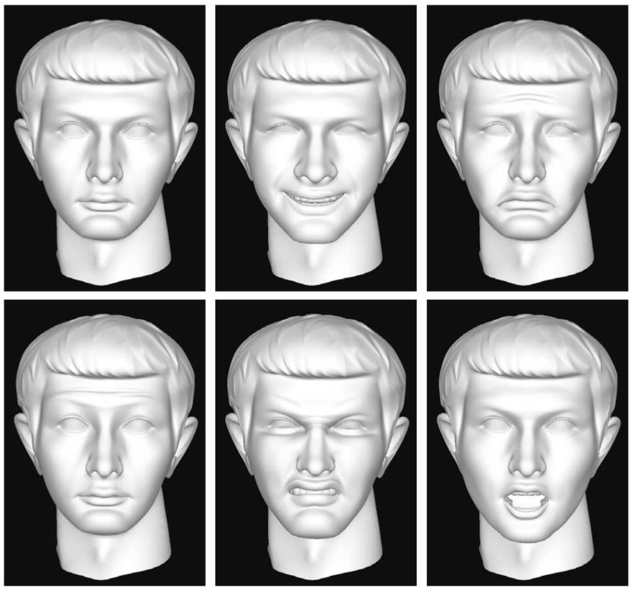


Fig. 17. The examples used for 3D expression synthesis.

collection, and Steve Lin for proofreading the paper. They appreciate the helpful suggestions for improving the paper that were provided by the anonymous referees. Many thanks to Professor Guoliang Chen for his support during the course of this work.

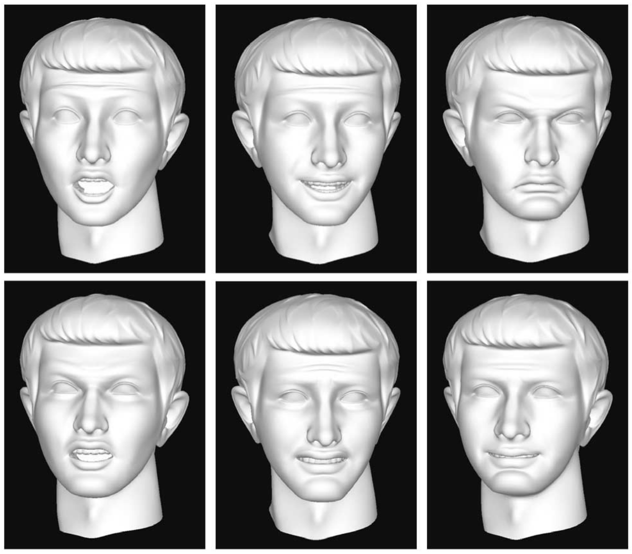


Fig. 18. The results of 3D expression synthesis.

REFERENCES

[1] D. Ballard and C. Brown, *Computer Vision.* Englewood Cliffs, N.J.: Prentice-Hall, 1982.
[2] T. Beier and S. Neely, "Feature-Based Image Metamorphosis," *Computer Graphics,* pp. 35-42, July 1992.
[3] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating Faces in Images and Video," *Proc. Eurographics Conf.,* 2003.
[4] M. Brand, "Voice Puppetry," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 22-28, Aug. 1999.
[5] C. Bregler, M. Covell, and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," *Computer Graphics,* pp. 353-360, Aug. 1997.
[6] S.E. Brennan, "Caricature Generator," MS Visual Studies, Dept. of Architecture, Massachusetts Inst. of Technology, Cambridge, Mass., Sept. 1982.
[7] T.F. Cootes and C.J. Taylor, "Statistical Models of Appearance for Computer Vision," http://www.isbe.man.ac.uk/bim/Models/app_models.pdf, 2001.
[8] Y. Du and X. Lin, "Realistic Mouth Synthesis Based on Shape Appearance Dependence Mapping," *Pattern Recognition Letters,* vol. 23, no. 14, pp. 1875-1885, 2002.
[9] T. Ezzat, G. Geiger, and T. Poggio, "Trainable Videorealistic Speech Animation," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 388-398, Aug. 2002.
[10] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.
[11] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making Faces," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 55-66, July 1998.
[12] P. Joshi, W.C. Tien, M. Desbrun, and F. Pighin, "Learning Controls for Blend Shape Based Realistic Facial Animation," *Proc. Symp. Computer Animation (SCA '03),* pp. 187-192, July 2003.
[13] Y. Lee, D. Terzopoulos, and K. Waters, "Realistic Modeling for Facial Animation," *Computer Graphics,* pp. 55-62, Aug. 1995.
[14] S.Z. Li and L. Gu, "Real-Time Multi-View Face Detection, Tracking, Pose Estimation, Alignment, and Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Demo Summary,* 2001.
[15] P. Litwinowicz and L. Williams, "Animating Images with Drawings," *Computer Graphics,* pp. 235-242, Aug. 1990.
[16] Z. Liu, Y. Shan, and Z. Zhang, "Expressive Expression Mapping with Ratio Images," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 271-276, Aug. 2001.
[17] D.G. Luenberger, *Linear and Nonlinear Programming.* Addison-Wesley, 1984.
[18] N. Magneneat-Thalmann, N.E. Primeau, and D. Thalmann, "Abstract Muscle Actions Procedures for Human Face Animation," *The Visual Computer,* vol. 3, no. 5, pp. 290-297, 1988.
[19] J.-Y. Noh and U. Neumann, "Expression Cloning," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 277-288, Aug. 2001.
[20] F.I. Parke, "Computer Generated Animation of Faces," *Proc. ACM Ann. Conf.,* Aug. 1972.
[21] F.I. Parke and K. Waters, *Computer Facial Animation.* Wellesley, Mass.: AK Peters, 1996.
[22] K. Perlin and A. Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 205-216, Aug. 1996.
[23] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin, "Synthesizing Realistic Facial Expressions from Photographs," *Computer Graphics, Proc. Ann. Conf. Series,* pp. 75-84, July 1998.
[24] F. Pighin, R. Szeliski, and D.H. Salesin, "Resynthesizing Facial Animation through 3D Model-Based Tracking," *Proc. Int'l Conf. Computer Vision (ICCV '99),* 1999.
[25] S. Platt and N. Badler, "Animating Facial Expression," *Computer Graphics,* vol. 15, no. 3, pp. 245-252, 1981.
[26] H. Pyun, Y. Kim, W. Chae, H.W. Kang, and S.Y. Shin, "An Example-Based Approach for Facial Expression Cloning," *Proc. Symp. Computer Animation (SCA '03),* pp. 167-176, July 2003.

[27] S.M. Seitz and C.R. Dyer, "View Morphing," *Computer Graphics,* pp. 21-30, Aug. 1996.

[28] D. Terzopoulos and K. Waters, "Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 6, pp. 569-579, June 1993.

[29] S. Toelg and T. Poggio, "Towards an Example-Based Image Compression Architecture for Video-Conferencing," Technical Report 1494, MIT, 1994.

[30] P.-H. Tu, I.-C. Lin, J.-S. Yeh, R.-H. Liang, and M. Ouhyoung, "Expression Detail Mapping for Realistic Facial Animation," *Proc. Eighth Int'l Conf. CAD/Graphics,* pp. 20-25, Oct. 2003.

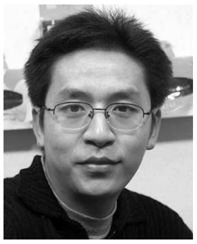
[31] K. Waters, "A Muscle Model for Animating Three-Dimensional Facial Expression," *Computer Graphics,* vol. 22, no. 4, pp. 17-24, 1987.

[32] L. Williams, "Performance-Driven Facial Animation," *Computer Graphics,* pp. 235-242, Aug. 1990.

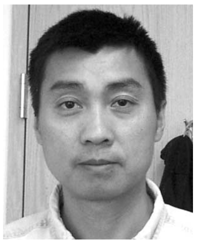
[33] Y. Ye, *Interior Point Algorithms: Theory and Analysis.* John Wiley, 1997.

[34] Q. Zhang, Z. Liu, B. Guo, and H. Shum, "Geometry-Driven Photorealistic Facial Expression Synthesis," *Proc. Symp. Computer Animation (SCA '03),* pp. 177-186, July 2003.

[35] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," *Proc. Int'l Conf. Computer Vision (ICCV '99),* pp. 666-673, 1999.



Qingshan Zhang received the PhD degree in computer science from the University of Science and Technology of China in 2003. He worked on face modeling and animation in Microsoft Research Asia from 2001 to 2003. He is now a researcher at the Research and Innovation Center of Alcatel Shanghai Bell Ltd. and his research topics are network and telecommunication technologies.



Zicheng Liu received the PhD degree in computer science from Princeton University, the MS degree in operational research from the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, China, and the BS degree in mathematics from Huazhong Normal University, Wuhan, China. He is currently a researcher at Microsoft Research. Before joining Microsoft, he worked as a member of technical staff at Silicon Graphics focusing on trimmed NURBS tessellation for CAD model visualization. His research interests include 3D face modeling and facial animation, linked figure animation, multisensory speech enhancement, and multimedia signal processing. He was the cochair of the IEEE International Workshop on Multimedia Technologies in E-Learning and Collaboration in 2003. He is a senior member of the IEEE.

Baining Guo received the PhD and MS degrees from Cornell University and the BS degree from Beijing University. He is a senior researcher and the research manager of the graphics group at Microsoft Research Asia (formerly Microsoft Research China). Before joining Microsoft, Baining was a senior staff researcher in Microcomputer Research Labs at Intel Corporation in Santa Clara, California, where he worked on graphics architecture. He holds more than 30 granted and pending US patents. He is a senior member of the IEEE.

Demetri Terzopoulos received the BEng and MEng degrees in electrical engineering from McGill University, Montreal, Canada, in 1978 and 1980, respectively, and the PhD degree in EECS (artificial intelligence) from the Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts, in 1984. He holds the Lucy and Henry Moses Professorship in Science at New York University (NYU) and is a professor of computer science and mathematics at NYU's Courant Institute. Terzopoulos has been a Killam Fellow of the Canada Council for the Arts, a Steacie Fellow of the Natural Sciences and Engineering Research Council (NSERC) of Canada, an AI/Robotics Fellow of the Canadian Institute for Advanced Research, and a senior visiting fellow at UCLA's Institute for Pure and Applied Mathematics. He is a member of the European Academy of Sciences and Sigma Xi. His published work comprises hundreds of research papers and several volumes, primarily in computer graphics and computer vision, as well as in medical imaging, computer-aided design, and artificial life/intelligence. His research has received numerous honors, including computer graphics awards from Ars Electronica, NICOGRAPH, Computers and Graphics, and the International Digital Media Foundation, and computer vision awards from the IEEE, the American Association for Artificial Intelligence, the International Medical Informatics Association, and the Canadian Image Processing & Pattern Recognition Society. He has been a conference or program chair of IEEE CVPR, Pacific Graphics, and the SIGGRAPH/EG Symposium on Computer Animation. He has been a founding editorial board member of journals spanning computer vision, computer graphics, medical imaging, and applied mathematics. He is a fellow of the IEEE.

Heung-Yeung Shum received the PhD degree in robotics from the School of Computer Science, Carnegie Mellon University, in 1996. He worked as a researcher for three years in the vision technology group at Microsoft Research Redmond. In 1999, he moved to Microsoft Research Asia where he is currently the managing director. His research interests include computer vision, computer graphics, human computer interaction, pattern recognition, statistical learning, and robotics. He is on the editorial boards for *IEEE Transactions on Pattern Analysis and Machine Intelligence* (PAMI), *International Journal of Computer Vision* (IJCV), and *Graphical Models*. He is the general cochair of the 10th International Conference on Computer Vision (ICCV 2005 Beijing). He is a senior member of the IEEE.