Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Geometry-Driven Photorealistic Facial Expression Synthesis

Qingshan Zhang,[1] Zicheng Liu[2], Baining Guo[1] and Harry Shum[1]

[1] Microsoft Research Asia, Beijing, China
[2] Microsoft Research, Redmond, WA

Abstract

*Expression mapping (also called performance driven animation) has been a popular method to generate facial animations. One shortcoming of this method is that it does not generate expression details such as the wrinkles due to the skin deformation. In this paper, we provide a solution to this problem. We have developed a geometry-driven facial expression synthesis system. Given the feature point positions (*geometry*) of a facial expression, our system automatically synthesizes the corresponding expression image which has photorealistic and natural looking expression details. Since 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 by using an example-based approach. Another application of our system is on expression editing where the user drags the feature points while the system interactively generates facial expressions with skin deformation details.*

1. Introduction

Realistic facial expression synthesis has been one of most interesting yet difficult problems in computer graphics. There has been a lot of research in this area, and the reader is referred to the book [16] by Parke and Waters for an excellent survey.

Expression mapping (also called performance-driven animation) [5, 12, 24, 16] has been a popular method to generate facial animations. It uses a performer's feature point motions to drive the feature point motions of a different person's face. One shortcoming of this method is that it does not produce expression details such as wrinkles caused by skin deformation. The technique proposed by Liu et. al. [13] 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 are obtained once offline. We call our system a geometry-driven facial expression synthesis system. Given the feature point positions (*geometry*) of a facial expression, our system automatically synthesizes the corresponding expression image which has 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 through an example-based approach. Another application of our system is on expression editing where the user drags the feature points while the system interactively generates facial expressions with skin deformation details.

The remainder of the paper is organized as follows. We review the related work in Section 2. From Section 3 through Section 9, we describe the algorithm in both 2D and 3D. In Section 3, we describe the feature point inference. The enhanced expression mapping is described in Section 11. In Section12, we describe the expression editing application. The results are shown in Section 13. Finally we conclude the paper and discuss future directions in Section 14.

2. Related work

There has been a lot of work on facial animation [1, 23, 21, 10, 15, 17, 7, 2, 20, 4, 18]. It is virtually impossible to enumerate all of them here. In the following, we will discuss a few that we think are mostly related to our work.

There has been a lot of success on speech driven facial animation [4, 3, 8]. Speech driven facial animation systems are mainly concerned about the mouth region, while our method is mainly for facial expressions. One interesting analogy is that speech driven animation systems use audio signals to derive mouth images, and our system uses feature point motions to derive the facial images. It would be interesting to combine these two techniques together to generate speech-driven facial animations with expressions.

Toelg and Poggio [22] 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. Choe and Ko [6] projected facial expressions to their manually generated muscle actuation basis. These muscle basis coefficients can then be used to drive a different face model.

Guenter et. al. [9] developed a system to use digital cameras to capture 3D facial animations. Noh and Neumann [26] developed the expression cloning technique to map the geometric motions of one person's expression to a different person.

One effective approach to generate photorealistic facial expressions with details is the morph-based approach [2, 20, 4, 18]. In particular, Pighin et al [18] used the convex combinations of the geometries and textures of the example face models to generate photorealistic facial expressions. They also provided a set of easy-to-use tools and interfaces to allow a user to interactively design facial expressions. Their system was mainly designed for offline authoring purpose and it requires a user to manually specify blending weights to obtain a desired expression. Our synthesis algorithm differentiates from the work of Pighin et. al. [18] in that our method is completely automatic. Furthermore, we have developed a technique to infer the feature point motions from a subset. By combining these two techniques together, we can enhance the traditional expression mapping system with expression details. In another paper, Pighin et. al. [19] used the expression morphing model to reconstruct the 3D expressions from a video sequence. Their system did not try to map the facial expressions to a different face model.

Liu et. al. [13] proposed a technique, called expression ratio image, to map one person's expression details to a different person's face. Given the feature point motions of an expression, their method requires an additional input of a different person's image with the same expression. In other words, their method requires an image of someone for every different expression. In contrast, our method can generate arbitrary number of expressions from a small set of example images. For the situations where no examples are available for the target face, their method is more useful. For the situations where we are given the feature point positions of an expression but no expression ratio images are available for this geometry, our method is more useful. For example, in expression editing applications, when a user manipulates the face feature points, it is unlikely that he/she can find a different person's expression image 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 the head pose changes, the ratio images may be difficult to create.

3. Geometry-driven expression synthesis

Given the feature point positions of a facial expression, to compute the corresponding expression image, one possibility would be to use some mechanism such as physical simulation [10] to figure out the geometric deformations for each point on the face, and then render the resulting surface. The problem is that it is difficult to model the detailed skin deformations such as the expression wrinkles, and it is also difficult to render a face model so that it looks photorealistic. We instead take an example-based approach.

Given a set of example expressions, Pighin et al. [18] demonstrated that one can generate photorealistic facial expressions through convex combination. Let $E_i = (G_i, I_i), i = 0, ..., m$, be the 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 $H(E_0, E_1, ..., E_m)$ be the set of all possible convex combinations of these examples. Then

$$H(E_0, E_1, ..., E_m) = \{(\sum_{i=0}^{m} c_i G_i, \sum_{i=0}^{m} c_i I_i) | \sum_{i=0}^{m} c_i = 1, c_i \geq 0, i = 0, ..., m\} \quad (1)$$

Pighin et al. [18] also developed a set of tools so that a user can use it to interactively specify the coefficients c_i to generate the desired expressions.

Notice that each expression in the space $H(E_0, E_1, ..., E_m)$ 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 to use the geometric component to infer the texture component.

Given the geometric component G, we can project G to the convex hull spanned by $G_0, ..., G_m$, and then use the resulting coefficients to composite the example images and obtain the desired texture image.

One problem with this approach is that the space of $H(E_0, E_1, ..., E_m)$ is very limited. A person can have expression wrinkles in different face regions, and the combinatorics is very high. So we subdivide the face into a number of subregions. For each subregion, we use the geometry which is associated with this subregion to compute the subregion texture image. We then seamlessly blend these subregion images to produce the final expression image.

One potential alternative to the convex combination is to simply use the linear space without adding constraints on the coefficients c_i's. The problem is that the coefficients resulted from the linear space approximation of the geometries may contain negative coefficients as well as coefficients which are larger than 1. It causes artifacts in the composited images.

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

4. System overview

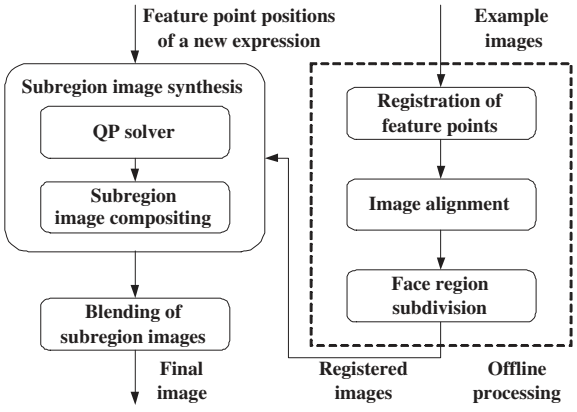
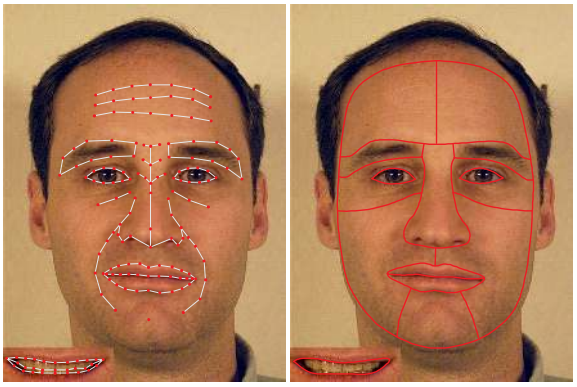


Figure 1: *An overview of the geometry-driven expression synthesis system.*

Figure 1 is an overview of our system. It consists of an offline processing unit and a run time unit. The example images are processed offline only once. At run time, 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



(a) (b)

Figure 2: *(a) Feature points. (b) Face region subdivision*

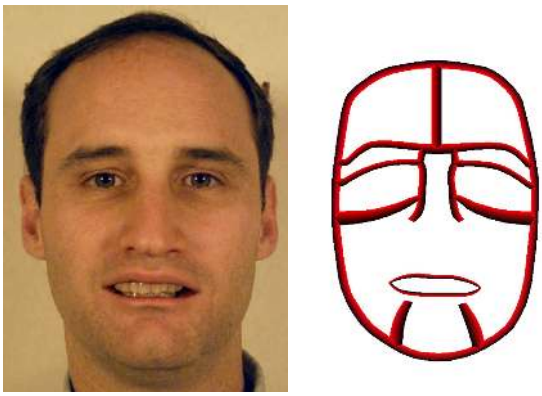
Figure 2(a) shows the feature points that we use in our system. At the bottom left corner are the feature points of the teeth area when the mouth is open. There are 134 feature points in total. Given a face image, it is possible to automatically compute face features [11]. Since the number of example images is very small in our system (10 to 15 examples per person). we choose to manually mark the feature points of the example images.

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 Figure 3(a). 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, 12] may be used to obtain better image quality.

5.3. Face region subdivision

We divide the face region into 14 subregions. Figure 2(b) shows the subdivision. At the bottom left corner is the subregion of the teeth 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 subdivision information where for each pixel, its subregion index is stored in its color channel.



(a) (b)

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

6. Subregion expression synthesis

Let n denote the number of feature points. For each example expression E_i, We use G_i to denote the $2n$ dimensional vector which consists of all the feature point positions. Let G be the feature point positions of a new expression. For each subregion R, we use G_i^R to denote the feature points of E_i which are in or at the boundary of R. Similarly we use G^R to denote the feature points of G associated with R.

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

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

Denote

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

and

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

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

There are multiple ways to solve a quadratic programming problem [14, 25]. In the past decade, a lot of progress have been made on the interior-point methods both in theory and in practice [25]. Interior-point methods have become very popular for solving many practical quadratic programming problems. This is the approach that we choose to use. 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 the traditional approaches, interior point methods have faster convergence rate both theoretically and in practice, and they are numerically stable. Even though an interior point method usually does not produce the optimal solution (since it is 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.

6.1. Subregion image compositing

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

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

Notice 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 the image discontinuity 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. Figure 3(b) shows the weight map, which is aligned with the standard image (Figure 3(a)). 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 G and B channels to store the indexes of the two neighboring subregions, respectively . Given a pixel in the blending region, let r denote the value of R-channel, and let i_1 and i_2 be the indexes of the two subregions. Then its blended intensity is

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

Notice 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 obtain the final expression image.

8. Teeth

Since the teeth region is quite orthogonal to the other regions of the face, we use a separate set of examples for the teeth region. In our current system, only a small set of examples for the teeth region are used since we are not focusing on the speech animations where there are a lot of variations on mouth shapes.

9. Expression synthesis in 3D

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 sub-region blending coefficients, we use equation 3 in the same way 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 sub-region mesh compositing and blending along sub-region 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 in Figure 2. For example, most of the algorithms to track face features only track a limited number of features along the eye brows, eyes, mouths, and noses. In the enhanced expression mapping example which we will discuss later in the paper, we only extract 40 feature points from the performer. For the application of expression editing which will be discussed in Section 12, each time when a user moves a feature point, we need to figure out what is the mostly likely movement for the rest of the feature points. In this section, we describe 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 a fine-grain control which is particularly important if only the motions of 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 component analysis on the example expressions.

At hierarchy 0, we have a single feature point set which controls the global movement of the entire face. There are four feature point sets at hierarchy 1 each controlling 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. There are 16 feature point sets at hierarchy 2. Some facial feature points belong to several sets at different hierarchies, and they are used as bridges between 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 component 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 hierarchical principal component analysis result to propagate vertex motions so that from the movement of a subset of feature points, we can infer the most reasonable movement for the rest of the feature points. The basic idea is to learn from the examples how the rest of the feature points move when a subset of the vertices move.

Let v_1, v_2,..., 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 indexes of the feature points belonging to this feature point set), we use $\delta V(F)$ to denote the sub-vector 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 $Proj(\delta V, F)$.

Let us first describe how to infer the motions of all the feature points from a single vertex motion. Assume vertex v_i has a motion and we obtain a vector δV where δv_i is equal to the displacement for vertex v_i while the rest of the vertex displacement are 0. To propagate the vertex motion, we first find the feature point set, F^*, which has the lowest hierarchy among all the feature point sets containing v_i. The algorithm proceeds as follows where for each feature point set F, we use the flag $hasBeenProcessed(F)$ to denote whether F has been processed or not. Initially we set $hasBeenProcessed(F)$ to be false for all the F.

$MotionPropagation(F^*)$

Begin

Set h to be the hierarchy of F^*.

If $hasBeenProcessed(F^*)$ is true, return.

Compute $Proj(\delta V, F^*)$.

Update δV with $Proj(\delta V, F^*)$.

Set $hasBeenProcessed(F^*)$ to be true.

For each feature set F belonging to hierarchy $h-1$ such that $F \cap F^* \neq \emptyset$

$MotionPropagation(F)$

For each feature set F belonging to hierarchy h+1 such that $F \cap F^* \neq \emptyset$

$MotionPropagation(F)$

End

Similarly, we can infer the motions of all the feature points from a subset. Let us assume a subset of the feature points: $v_{i_1}, v_{i_2}, ..., v_{i_k}$ have motions. We set the vector δV so that δv_{i_j} is equal to the displacement vector for vertex v_{i_j} for $j = 1, ..., k$. For each vertex v_{i_j}, we find the feature point set, F^j, which has the lowest hierarchy among all the feature point sets containing v_{i_j}, and run $MotionPropagation(F^j)$ (notice that now δV contains the displacement for all $v_{i_j}, j = 1, ..., k$).

11. Enhanced expression mapping

Expression mapping technique (also called performance-driven animation) [5, 12, 24, 16] is a simple and widely used technique for facial animations. It works by computing the difference vector of the feature point positions between the neutral face and the expression face of a performer, and then adding the difference vector to the new character's face geometry. One main drawback is 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 situation where we can obtain the example images for the new character. The example images may be obtained offline through capturing or designed by an artist. At the run time, we first use the geometric difference vector to obtain the desired geometry for the new character 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 3 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 final results are more convincing and realistic facial expressions.

12. Expression editing

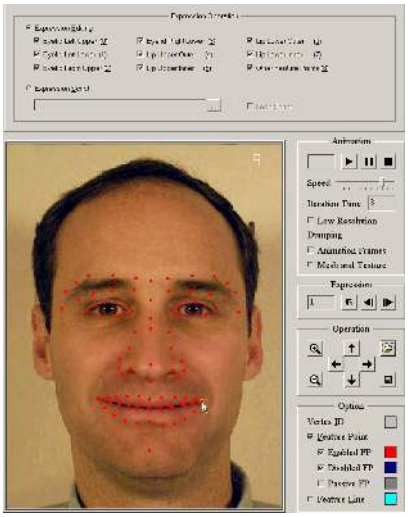


Figure 4: *The expression editing interface. The red dots are the feature points which a user can click on and drag.*

Another interesting application of our technique is on interactive expression editing. One common approach to designing facial expressions is to allow a user to interactively modify control point positions or muscle forces. 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 face feature point, and the system interactively displays the resulting image with expression details. Figure 4 is a snapshot of the expression editing interface where the red dots are the feature points which the user can click on and drag.

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 3. For example, if a user drags the feature point on the top of the nose, the entire nose region will move instead of just this single point.

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

13. Results

We show some experimental results for two faces: a male and a female. For each person, we capture about 30-40 images with whatever expressions they can make. We then select the example images, and

the use the rest of the images as the ground truth to test our system.

Figure 5 shows the example images for the male. The teeth examples are shown in Figure 6. Figure 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. We would like to point out that each of the expressions in Figure 10 is different from the expressions in the examples. But the results from our system closely match the ground truth images. There is a small blurriness in the synthesized images because of the pixel misalignment resulted from the image warping process.

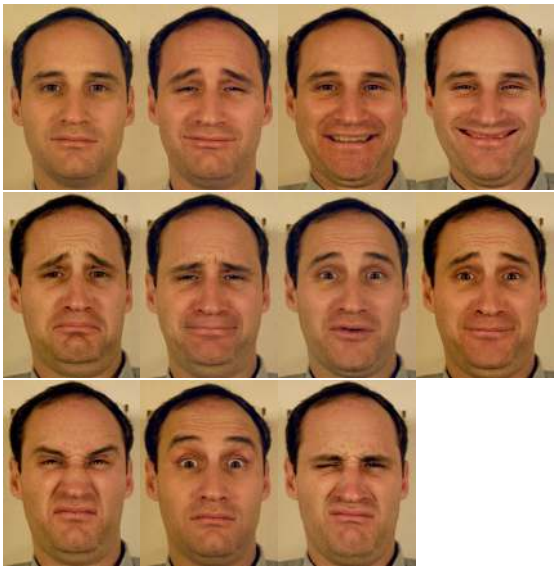


Figure 5: *The example images of the male.*

Figure 6: *The example images of the male's teeth.*

Figure 7 shows the example images of the female. Figure 11 is the side-by-side comparison for the female where the ground truth images are on the left while

Figure 7: *The example images of the female.*

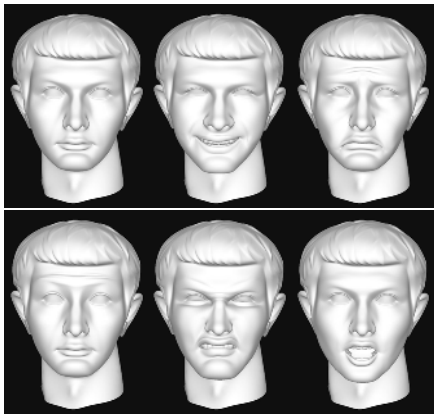


Figure 8: *The examples used for the 3D expression synthesis.*

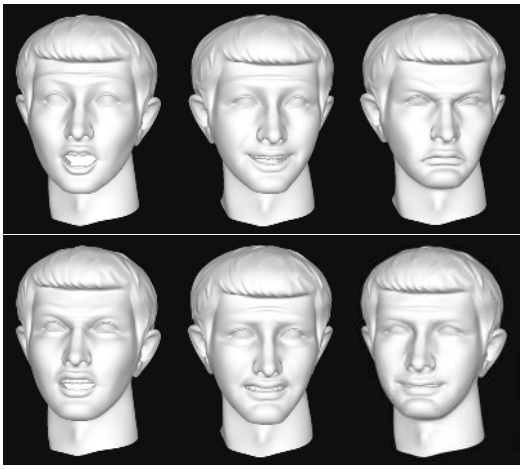


Figure 9: *The results of the 3D expression synthesis.*

the synthesized images are on the right. Again, the synthesized results match very well with the ground truth images.

Next we show the results of the expression mapping enhanced with our facial expression synthesis system. Figure 12 shows some of the results of mapping the female's expressions to the male. The female's expressions are the real data. The images on the right are the results of the enhanced expression mapping. We can see that the synthesized images have the natural looking expression details. For clarification purpose, we would like to point out that to map the female's expressions to the male, we do not need any example expressions from the female. We only need the feature points of the female's expressions. This is very different from the expression mapping in [18] which needs example expressions for both actors and requires the correspondence between the two sets of example expressions.

Figure 13 shows some of the expressions generated by our expression editing system. Notice that each of these expressions has a different geometry than the example images. Our system is able to produce photorealistic and convincing facial expressions.

To test the frame-to-frame coherence of our expression synthesis system, we have experimented with both artificial sequences and live 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. For each live sequence, we manually extract about 40 feature points for all the frames. For each frame, we infer the positions for the rest of the feature points and then use our expression synthesis system to produce the expression image. The accompanying video shows both the real sequences and the synthesis results.

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

We have captured a 3D face model of the male by using a laser scanner. We then use the feature point motions of the male 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. We add the head rotations by linear interpolation. The accompanying video shows the results.

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 Figure 8. Figure 9 shows some of the synthesized expressions in different poses.

Finally, the accompanying video shows the expression editing in action. The sizes of the images used in our experiments are 600x800. Our current system achieves 2-4 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 5 frames between the previous mouse stop and the current mouse stop, 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 0. One way to increase the frame rate is to not composite those example images whose coefficients are close to 0. Another way is to use hardware acceleration. We are planning on exploring both approaches to improve the performance.

14. Conclusion and future work

We have presented a geometry-driven facial expression synthesis system, and a feature point inference technique. The combination of these two techniques allows us to enhance the traditional expression mapping to generate facial expression details. This is the first expression mapping system which is capable of generating expression details while only requires the feature point motions from the performer. We have also demonstrated the expression editing application where the user, while manipulating the geometric positions of the feature points, can see the resulting realistic looking facial expressions interactively.

In the future, we are planning on improving the computation speed by accelerating the image compositing module. Another area that we would like to improve on is the image alignment so that the resulting images are sharper. Potential solutions include optical flow techniques and better image warping algorithms.

To generate expressions with various poses, we currently need to use 3D face models. Another possibility is to extend our technique to synthesize expressions with various poses from examples. We could potentially take as input the pose parameters as well as the feature point motions, and synthesize the corresponding expression from the examples. Another area we

would like to work on is to handle lip motions during speech. One potential approach is to combine our technique with the technique presented in [8]. One of our final goals is to be able to take the minimum information, such as the feature points, poses, and phonemes, of the performer and automatically synthesize the photorealistic facial animations for the target character.

Acknowledgements

We owe a debt to Ying Song who implemented the geometric deformation system for the scanned 3D head model. We would like to thank Bo Zhang for helping with the video editing. We would like to thank Hong Jiang and Stephen Dahl for helping with the data collection.

References

1. N. Badler and S. Platt. Animating facial expressions. In *Computer Graphics*, pages 245–252. Siggraph, August 1981.
2. T. Beier and S. Neely. Feature-based image metamorphosis. In *Computer Graphics*, pages 35–42. Siggraph, July 1992.
3. M. Brand. Voice puppetry. In *Computer Graphics, Annual Conference Series*, pages 22–28. Siggraph, August 1999.
4. C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Computer Graphics*, pages 353–360. Siggraph, August 1997.
5. S. E. Brennan. *Caricature Generator*. M.S. Visual Studies, Dept of Architecture, Massachusetts Institute of Technology, Cambridge, MA., Sept. 1982.
6. B. Choe and H. seok Ko. Analysis and synthesis of facial expression with hand-generated muscle actuation basis. In *Proceedings of Computer Animation*, 2001.
7. P. Ekman and W. V. Friesen. *Manual for the Facial Action Coding System*. Consulting Psychologists Press, Inc., 1978.
8. T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. In *Computer Graphics, Annual Conference Series*, pages 388–398. Siggraph, August 2002.
9. B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Computer Graphics, Annual Conference Series*, pages 55–66. Siggraph, July 1998.
10. Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Computer Graphics*, pages 55–62. Siggraph, August 1995.
11. S. Z. Li and L. Gu. Real-time multi-view face detection, tracking, pose estimation, alignment, and recognition. In *IEEE Conf. on Computer Visioin and Pattern Recognition Demo Summary*, 2001.
12. P. Litwinowicz and L. Williams. Animating images with drawings. In *Computer Graphics*, pages 235–242. Siggraph, August 1990.
13. Z. Liu, Y. Shan, and Z. Zhang. Expressive expression mapping with ratio images. In *Computer Graphics, Annual Conference Series*, pages 271–276. Siggraph, August 2001.
14. D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1984.
15. N. Magneneat-Thalmann, N. E. Primeau, and D. Thalmann. Abstract muscle actions procedures for human face animation. *Visual Computer*, 3(5):290–297, 1988.
16. F. I. Parke and K. Waters. *Computer Facial Animation*. AKPeters, Wellesley, Massachusetts, 1996.
17. K. Perlin and A. Goldberg. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics, Annual Conference Series*, pages 205–216. Siggraph, August 1996.
18. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *Computer Graphics, Annual Conference Series*, pages 75–84. Siggraph, July 1998.
19. F. Pighin, R. Szeliski, and D. H. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *International Conference on Computer Vision (ICCV'99)*, 1999.
20. S. M. Seize and C. R. Dyer. View morphing. In *Computer Graphics*, pages 21–30. Siggraph, August 1996.
21. D. Terzopoulos and K. Waters. Physically-based facial modeling and animation. *Journal of Visualization and Computer Animation*, 1(4):73–80, March 1990.
22. S. Toelg and T. Poggio. Towards an example-based image compression architecture for video-conferencing. In *MIT Technical Report No. 1494*, 1994.
23. K. Waters. A muscle model for animating three-dimensional facial expression. *Computer Graphics*, 22(4):17–24, 1987.
24. L. Williams. Performace-driven facial animation. In *Computer Graphics*, pages 235–242. Siggraph, August 1990.
25. Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley, 1997.
26. J. yong Noh and U. Neumann. Expression cloning. In *Computer Graphics, Annual Conference Series*, pages 277–288. Siggraph, August 2001.

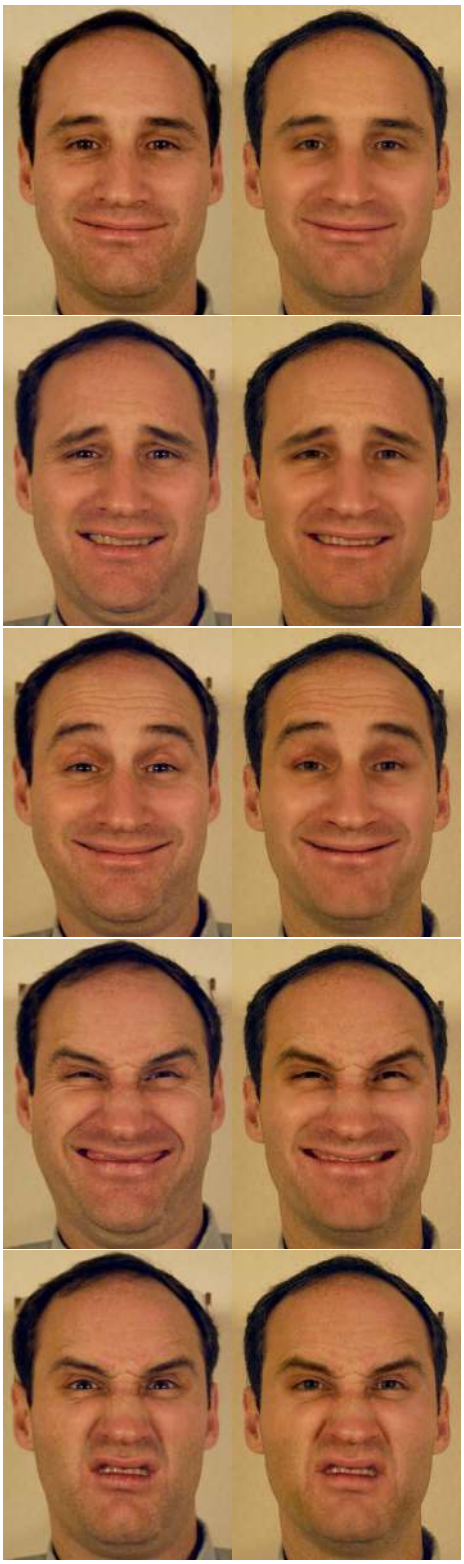


Figure 10: *Side-by-side comparison with ground truth for the male. The left column are the ground truth. The right are synthesis results.*

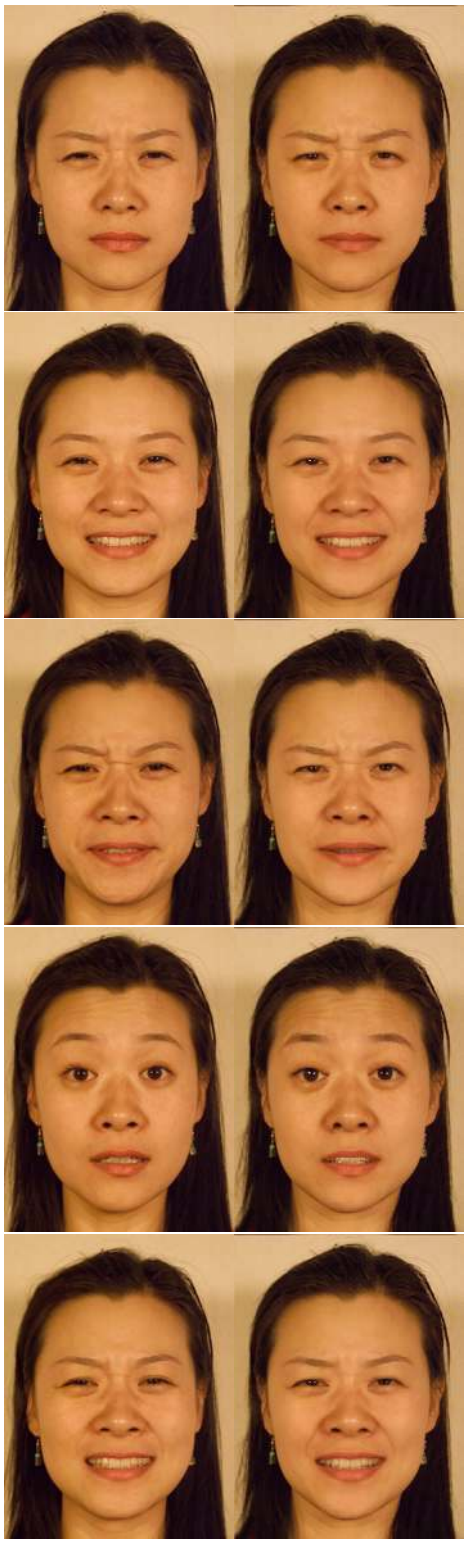


Figure 11: *Side-by-side comparison with the ground truth of the female. The left column are the ground truth. The right are synthesis results.*

Figure 12: *Results of the enhanced expression mapping. The expressions of the female are mapped to the male.*

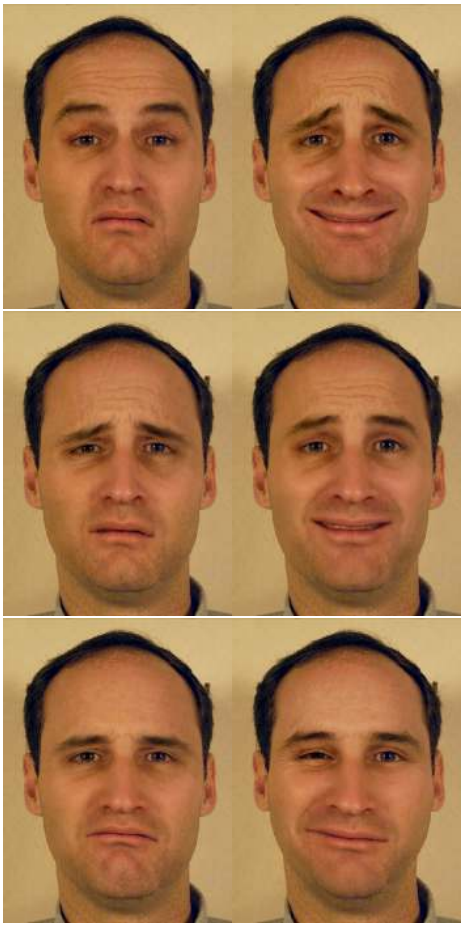


Figure 13: *Expressions generated by the expression editing system.*