A Method for Animating a Customized Face

Kaicheng Zhang, Zhiyong Huang and Tat-Seng Chua

Department of Computer Science
School of Computing
National University of Singapore
Singapore 117543
{zhangkai, huangzy, chuats}@comp.nus.edu.sg

Abstract

Animation of a customized human face is to develop techniques to have a new face model to speak the same sentences with the similar expression as the face in an existing facial animation sequence. In this paper, we propose a rapid modeling method. In this method, two orthogonal photos of a human face are used as input. Based on these two photos, a generic animation-ready face model will be adapted for both the geometry and muscle model underneath. We formulate it as a non-linear optimization problem and apply the Levenberg-Marquardt method for the solution. The result is a new animation-ready model. Applying the previous motion specification to this model, we can animate the customized face. We have implemented the method. The results are good.

Keywords: *Face modeling and animation, Motion reuse, Reconstruction, Non-linear optimization*

1. Introduction

Modeling and animation of the human face is interesting and challenging. There is no landscape that we know as well as the human face. The human facial features are the most intimately scrutinized features in existence. Every detail of the nose, eyes, and mouth, regularity in proportion, variation from one individual to the next, are matters about which we are all authorities. We can recognize a face from vast universe of similar faces and are able to detect very subtle changes in facial expression.

Modeling and animation of the human face have important application in many areas. Essentially, the face is the part of the body we use to recognize individuals. Our faces play an important role in our intellectual communication. In recent years, there has been considerable number of work and research done on the topic. Modeling and animation of realistic human face for an individual has wide applications in games industry, film industry, teleconferencing, and virtual reality.

Animation of a customized human face is to develop techniques to have a new face model to speak the same sentences with the similar expression as the face in an existing facial animation sequence. A more general area is motion reuse, where the goal is to adjust the existing animation to a new model, e.g., applying a dance sequence of a tall man to a young boy.

In this paper, we propose a rapid modeling method for animation of a customized human face. In this method, two orthogonal photos of a human face are used as input. Based on these two photos, a set of feature points are manually selected, and the generic animation-ready face model will be adapted for both the geometry and muscle model underneath. We formulate it as a non-linear optimization problem and apply the Levenberg-Marquardt method for the solution. We can get the result instantly. The result is a new animation-ready model. Applying the previous motion specification to this model, we can animate the customized face. We have implemented the method. The results are good.

2. Related Work and the *SimpleFace* Model

Much work has already been done in the area of facial reconstruction [Park75, Plat81, Wate87, Kuri91, Kalr94, Lee95, Essa96, Lee97, Guen98, Pigh98, Golt99, Noh01]. An excellent summary of the tools and techniques used in facial reconstruction and animation can be found in [Park96]. However, these research works emphasize photo realistic quality that requires extensive input, a large number of sample points, and intensive user interaction. Except for [Lee95 and Lee97], most of the resulting face models consist of only the facemask, i.e., the face geometry and texture information, not animation-ready. Our method is similar to [Lee97] in the general framework. However, there are three differences: (1) we formulate the reconstruction as a non-linear optimization problem and solve it using the Levenberg-Marquardt method while [Lee97] formulates as structured snake and solves it by the Dirichlet Free Form Deformation (DFFD), (2) we can adapted the muscle models and the facemask together using the same way while [Lee97] does not discuss it, and (3) our method starts from the *Simple2.0*, a generic animation-ready face model free available on Internet [Wate98].

The SimpleFace is based on skin-muscle model in polygonal representation, implemented in C with OpenGL. In Figure 1, the hierarchy of the model is illustrated.

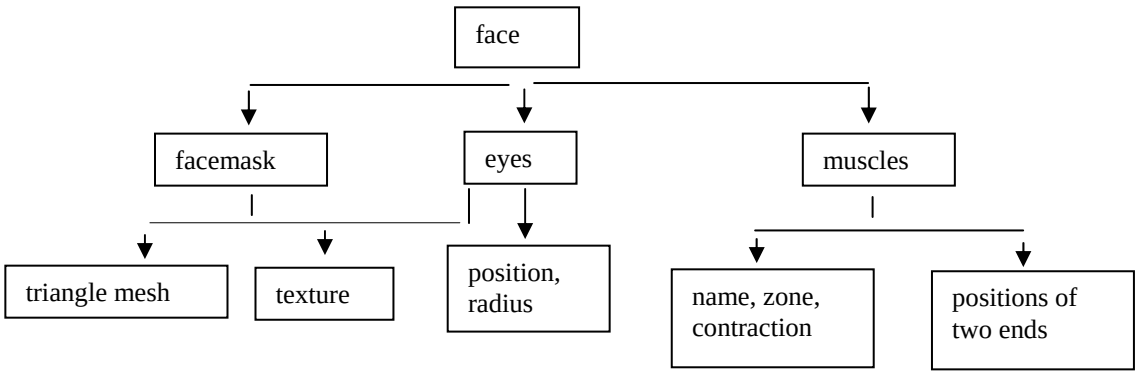

Figure 1. The hierarchy of *SimpleFace*

The facial model has three main parts: the facemask, muscles and the eyes. The facemask describes the geometrical feature of the face. It is a triangular mesh with the face texture. There are 256 vertices in half of the face. The eyes are modeled as two half spheres. They have parameters as position, radius, with an eye texture image. The muscles represent the real muscles lying under the facemask. There are 18 muscles modeled. Each muscle has two end vertices, and some internal parameters: name, zone of influence, and contraction. Figure 2 shows the rendering of the model in wireframe, where the muscle is shown as the red color segment and the active one in yellow.

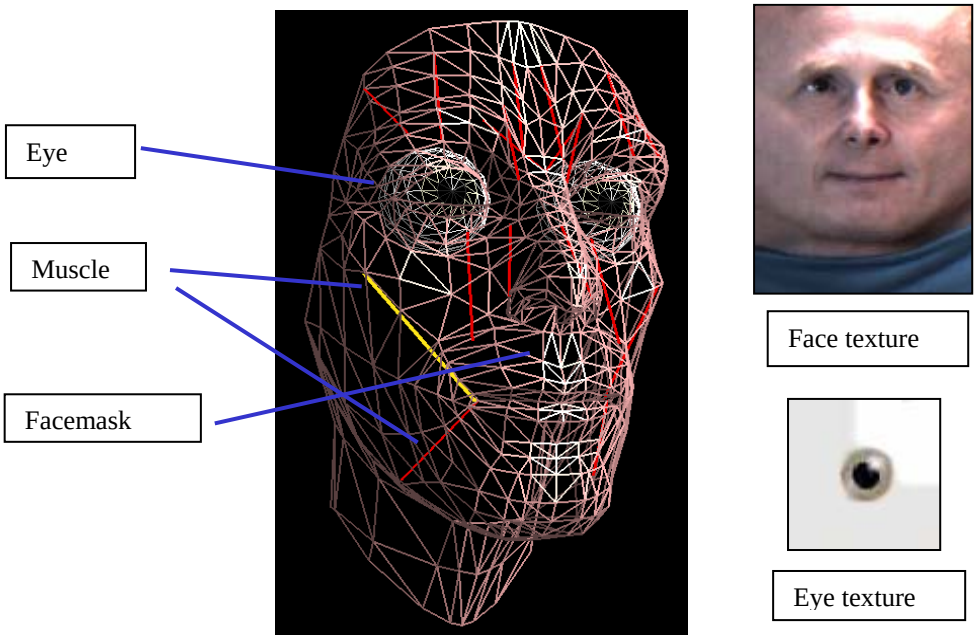

Figure 2. The rendering of the *SimpleFace*

Animation is achieved as follows for the *SimpleFace*. The key factor is the muscle model. Each muscle can take effect on the vertices in the facemask within its influence zone. When it contracts or extended, it will move the influenced vertices of the facemask. Waters proposed three basic types of muscle model: linear muscles, sphincter muscles, and sheet muscles. Only the linear muscles are implemented.

As shown in Figure 3, each linear muscle has two end vertices – v_1 for the head, and v_2 for the tail. The zones of influences of the muscle is defined by two radius R_s and R_f, and an angle a_1. A contraction value k represents the contraction state of the muscle, measuring of how hard a muscle is pulling or pushing the vertex in its influence zones.

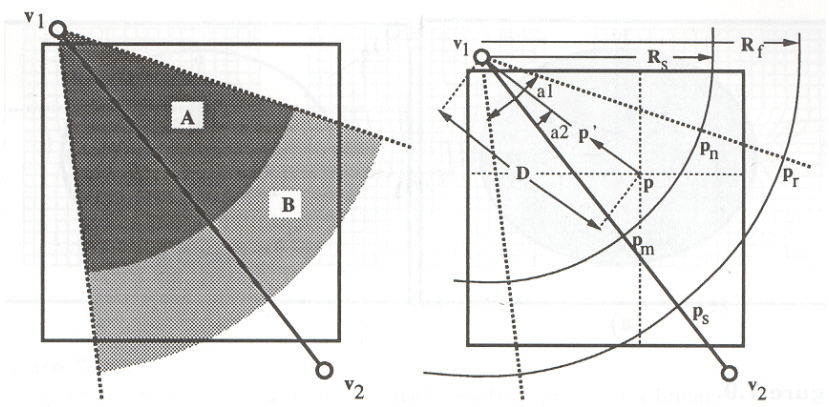

Figure 3. The linear muscle zones of influences

For each vertex p in the influence zones of the muscle, the new position p' under the effect of the muscle is given by

$$p' = p + a\,k\,r\,\frac{pv_1}{\| pv_1 \|}, \tag{1}$$

where $a = \cos(a2)$, k is the contraction value of the muscle, and

$$r = \begin{cases} \cos\dfrac{1-D}{R_s}; & \text{for p in zone A.} \\ \cos\dfrac{D-R_s}{R_f - R_s}; & \text{for p in zone B.} \end{cases} \tag{2}$$

$$D = \| v_1 - p \|,$$

By observing Equation (1), under the effect of the muscle, p always moves to new location p' along the direction pv_1 by a distance of a k r, which are both controlled by the contraction value k, distance and orientation of p with respect to the muscle.

As the muscle always pull (or push) the vertex in its influence zones towards (or against) the face vertex, it is thus called a linear muscle. By having ability to control the change of face mesh using these muscles, we can generate different expressions. A facial expression can be defined by a set of muscle contraction values. If two different persons have exactly the same facial expression, each pair of the corresponding muscles on the two faces have the same contraction value.

3 Our Work

The problem we address is a rapid face modeling for animation of a customized face. Our approach is to adapt the *SimpleFace* from two orthogonal photos. The input consists of (1) the *SimpleFace*, and (2) two orthogonal photos of a human face. By applying our method, a customized animation-ready face model can be adapted from the *SimpleFace*.

There are three assumptions for our implementation: (1) the two input images are orthogonal projection (with scaling) in *–Z* (front) and *–X* (left) directions, respectively, (2) some details of the head are omitted, including ears, teeth, tongue, and hair, and (3) at least the coordinates of the two vertices in the customized face are known.

The customized facial model will have the same structure as the generic model, and what to be changed are: (1) coordinates of each vertex in vertex set *V*, (2) coordinates of head and tail vertices in muscle set *M*, (3) face texture, eye texture, and (4) eye radius and positions.

The major problem here is to deform the model. That is, for a vertex in generic model $v_0(x_0, y_0, z_0)$, find its corresponding coordinate in the customized model $v(x, y, z)$, so that the projected image of the new model will be matched to the input photos. The information we can have is the projected 2D image coordinate of v in the front (portrait) and side (profile) image (x_1, y_1) and (z_2, y_2). This information can be obtained by manual labeling the projection of point v in the two images. Intuitively, if we know the two projection, we can immediately get back $v(x, y, z)$ using (x_1, y_1) and (z_2, y_2). However, there are as many as *256* vertices in a half face, it is tedious and imprecise if we manually label them one by one. The solution is to select a group of feature points among all the vertices, label them and get their

customized 3D coordinates, then determine the coordinates of the rest of the vertices using these feature points. Now, we describe it in detail.

3.1 Model Deformation

The first step is the global scaling. Since human faces vary in the ratio of height (*y*-direction) over width (*x*-direction), but the depth (*z*-direction) does not vary much, before we deform each vertex locally, we can do a global scaling to justify the ratio on height over width on the whole model. Suppose the height and width of the face in generic model is h_0 and w_0, in the front image we can find the new height and width h and w. Then for each vertex $v_0(x_0, y_0, z_0)$ in face mask and muscle, its scaled coordinates is given by Equation (3)

$$\begin{cases} x = x_0, \\ y = \left(\dfrac{h}{w} \Big/ \dfrac{h_0}{w_0} \right) y_0, \\ z = z_0. \end{cases} \tag{3}$$

The scaled model will be used for pre-determining the coordinates of two vertices in the customized face, and recovering the projection. In the following parts, we use "generic model" to refer the scaled model, rather than the original *SimpleFace*.

The second step is to adapt the feature vertices using the input front and side view images. It is illustrated in Figure 4.

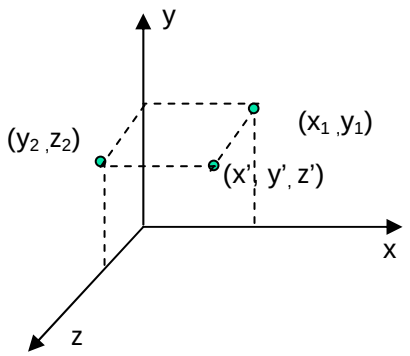

Figure 4. Using the front and side view images to adapt the feature points

To get the 3D coordinates from its projected 2D coordinates in the two front and side images, we need to recover the front and side projections. Since the front and side projections are orthogonal, as shown in Figure 4, a vertex P'(x', y', z') and its' two 2D coordinates (x_1, y_1) and (z_2, y_2) will have the relation as in Equation 4.

$$\begin{aligned} x_1 &= c_1 x' + t_{x1}, \\ y_1 &= c_1 y' + t_{y1}, \\ z_2 &= c_2 z' + t_{z2}, \\ y_2 &= c_2 y' + t_{y2}, \end{aligned} \tag{4}$$

where c_1 is the scaling factor for the front image, t_{x1} and t_{y1} are translation along *x* and *y* direction; c_2 is the scaling factor for the side image, and t_{z2} and t_{y2} are translation along *z* and *y* direction. To solve for these 6 unknowns, we need at least 2 points in the customized model with known coordinates, which could give us 8 equations. This is why we make the third assumption when we define the problem at the beginning of Section 2. For simplicity, we can just choose 2 points and assume that they have the same coordinates as in the generic model.

3.2 Feature Point Selection and Model Fitting for Facemask and Muscle Model

Once we have recover the projection, for a point *P*, we can label it in the front and side image, thus obtaining (x_1, y_1) and (z_2, y_2), then from Equation 4 we can compute the coordinate of P(x', y', z') by Equation 5:

$$\begin{cases} x' = \dfrac{x_1 - t_{x1}}{c_1}, \\ y' = \left(\dfrac{y_1 - t_{y1}}{c_1} + \dfrac{y_2 - t_{z2}}{c_2}\right) \Big/ 2, \\ z' = \dfrac{z_2 - t_{z2}}{c_2}. \end{cases} \tag{5}$$

To deform all the vertices in the whole facemask, one naïve way is to manually label the projected points in front and side image and use Equation 5 to get the deformed 3D coordinate for every vertex. But apparently there are two problems with this method: (1) there are 256 vertices in the half facemask, too many to be labeled, and (2) more importantly, not all vertices are easy to locate. Some of the vertices appear to be able to project to different locations in the image, even invisible in the images.

Thus we come out with the idea of selecting only a group of vertices from vertex set as "feature points". These feature points should:

(1) catch most features of the individual face,

(2) be easy to locate and label on the image, and

(3) focus on most crucial parts of the face: lips, nose, eyes.

Figure 5 shows the 22 feature points we choose among all the 256 vertices of the half facemask.

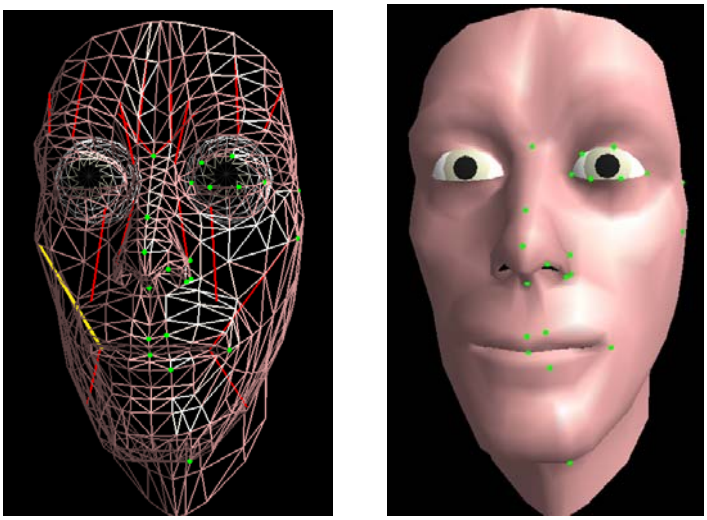

Figure 5. Feature points highlighted in green color

We label these feature points to get the real coordinates of them in the customized model. The coordinates of rest can then be approximated using a fitting method. We first define "vector displacement". The vector displacement from vector u to vector v can be defined by

$$\begin{aligned} d(u,v) &= \frac{\| \tilde{u} - \tilde{v} \|}{\| u \|} \\ \tilde{u} &= u / \| u \| \\ \tilde{v} &= v / \| v \| \end{aligned} \tag{6}$$

where we denote the normalized vector of u as $\tilde{u}$, and normalized vector of v as $\tilde{v}$.

Now for a feature point Q_0 in generic model, we already know its customized position Q. We want to find for a non-feature point P_0, its customized position P. We now let

$$\begin{aligned} u &= \overrightarrow{P_0 Q_0}, \\ v &= \overrightarrow{PQ}, \end{aligned} \tag{7}$$

as shown in Figure 6. Now, $d(u,v)$ becomes the "error" of P in the deformed model. It has the following properties: (1) when orientation change increases, d increases, and (2) when Q_0 is nearer to P_0 in the generic model, d is bigger.

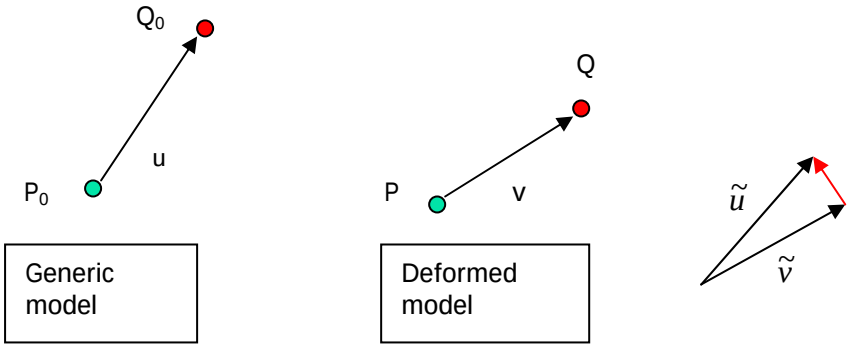

Figure 6. Determining P from feature point Q

Intuitively, in the deformed model, *Q* will force *P* to have the same orientation as it is in the generic model: the nearer to Q_0 in the generic model, the stronger influence will be on *P* in the new model.

Now instead of using a single *Q*, we use all the feature points we selected (Figure 7).

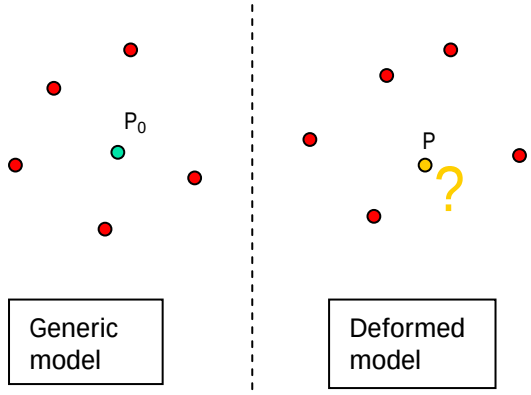

Figure 7. Determine a non-feature (green) point using feature points (red)

We can sum all the vector displacements obtained from *P* to each point in the generic and deformed model and try to minimize it. Assuming we have a set of feature points *Feature* with k elements,

$$\sum_{Q_{0,i},Q_{1,i}\in Featuret} d(P_0Q_{0,i},PQ_{1,i}) = f_1(x_p,y_p,z_p) + f_2(x_p,y_p,z_p) + \cdots + f_k(x_p,y_p,z_p) \tag{8}$$

As we can see in Equation 8, each term of the sum is a vector displacement described in Equation 6 and 7, in each term the only unknowns are the coordinates of *P* in the deformed model: x_p, y_p and z_p, and each term is positive. Therefore, we can view the sum as *k* functions about x_p, y_p and z_p. Minimizing the sum becomes a non-linear least square problem, which can be solved using Levenberg-Marquardt method [Chen80, Pres93], to find the most fitting (x_p, y_p, z_p) which minimize the sum. Applying this method to each non-feature point, we can have all the deformed coordinates of the vertices in the facemask.

Now, we discuss the adaptation for the muscle model. For the vertices in the muscles, we apply the similar technique. This time for each muscle end vertex *P* in the generic model, we make use of the vertex set *VertexSet* of the facemask we have just deformed to play the role as feature points, i.e. we are going to minimize

$$\sum_{Q_{0,i},Q_{1,i}\in VertexSet} d(P_0Q_{0,i},PQ_{1,i}) = f_1(x_p,y_p,z_p) + f_2(x_p,y_p,z_p) + \cdots + f_k(x_p,y_p,z_p) \tag{9}$$

using the Levenberg-Marquardt method again. We can obtain the best-fit muscle end vertices in the deformed facial model. Thus, we adapt the muscle model and facemask together in the same way.

3.3 Texturing Mapping

After deforming the generic facemask into our customized one, we can apply texturing mapping for photo-realistic rendering. For each vertex in the facemask, we can use the front and side projections we have used before and project the facemask back to the 2D images:

$$\begin{cases} x_1 = c_1 x' + t_{x1} \\ y_1 = c_1 y' + t_{y1} \\ z_2 = c_2 z' + t_{z2} \\ y_2 = c_2 y' + t_{z2} \end{cases} \tag{10}$$

Then (x_1, y_1) (z_2, y_2) are used as the texture coordinates of the vertices in the front and side images. Each polygon can be rendered using either the front part of the front image or part of the side image. In our implementation, we render those polygon lies within the outer corners of two eyes using the front images, and the rest with side images.

3.4 Other Refinements

There are also some refinements we need to do in customizing the generic model, and most of them are done manually in the current implementation.

First, the input images sometimes cannot be directly used as texture images. We need to adjust the brightness of the two images such that when we use both of them on the face there will not be much difference of the brightness. It is necessary for using both images in texture mapping. We also need to remove some unwanted edge and noise in the images.

Second, we also have to produce the texture image for the eyes. This image is simply extracted from the front image by using some image-processing tool. The radius and position of the eyes is also adjusted manually.

4. Results

Figure 8 shows the front and side images. Figure 9 shows the textured view of the generic model and the customized model.

Figure 8. Input face images, front (portrait) and side (profile) views.

Feature points highlighted in green color

Figure 9. The generic face and the result of the customized face

The customized model preserves the animation capability of the generic model, i.e., animation-ready. In the generic model, there is a set of expression defined using muscle contraction values. Now we apply the same values in the customized model, and we get the same expressions on the customized model (Figure 10). A video demo can be found in http://www.comp.nus.edu.sg/~zhangkai/urop/demo.avi.

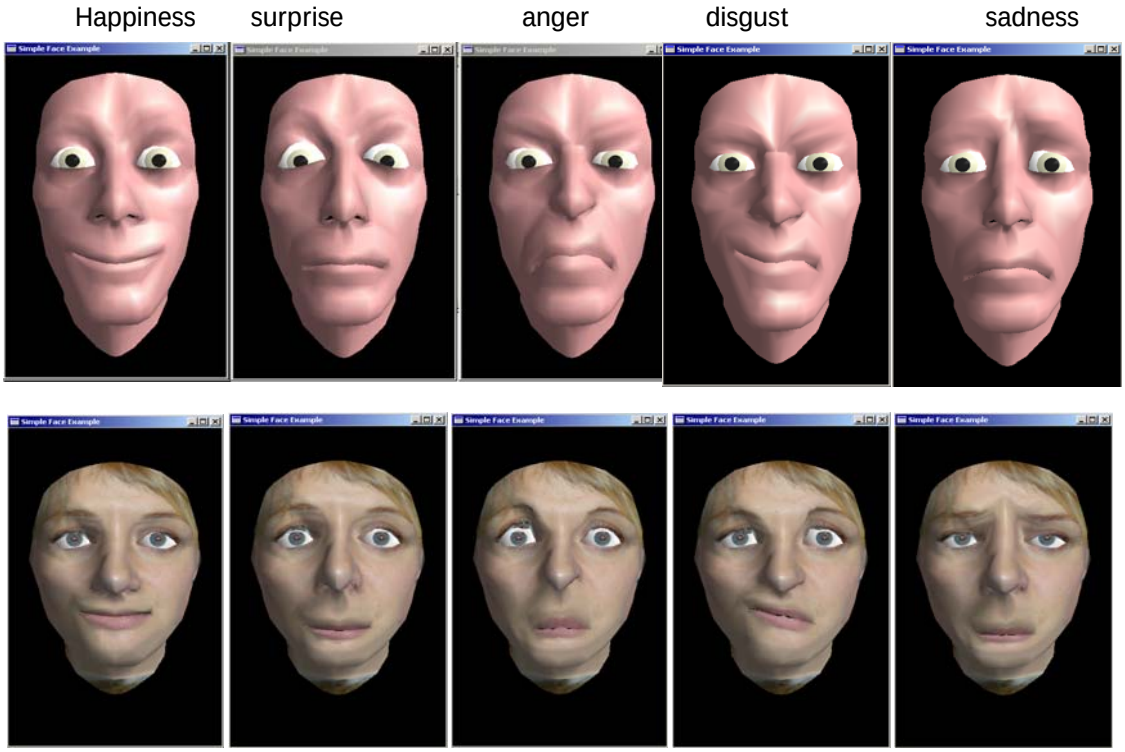

Figure 10. Different expressions in generic and customized model

5. Conclusion and Future Work

In this paper, we propose and implemented a method for animating a customized face. A generic face model, which can be animated, is customized using two photos of a new face. The major techniques we employed are: model deformation of facemask and muscles, texture mapping, and some refinements. In order to deform the model, we need to recover the projection of the two photos, and then compute the positions of feature points using the projections. The positions of the rest of the points are determined using the vector displacement.

There are several possible improvements for the future work: for easier use, we can apply more sophisticated image-processing techniques to detect the feature points automatically such as [Lin00]. Another way is to have a better generic face model: we can add more types of muscles such as the sphincter and sheet muscles defined in [Wate87].

6. References

[Akim93] Akimoto, T., Seunaga, Y., and Wallace, R. S., *Automatic Creation of 3D Facial Models*, IEEE Computer Graphics and Applications, Vol. 13, No. 5, 1993, pp. 16-22.

[Chen80] Cheney W. and Kincaid D., *Numerical Mathematics and Computing*, Rooks-Cole, Montery, CA, 1980.

[Essa96] Essa, I., Basu, S., Darrell, T., and Pentland, A., *Modeling, Tracking and Interactive Animation of Faces and Heads Using Input from Video*, Computer Animation 1996, pp. 68-79.

[Guen98] Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F., *Making Faces*, SIGGRAPH 1998, pp. 55-66.

[Gotl99] Gotla M. and Huang Z., *A Minimalist Approach to Facial Reconstruction*, MMM 1999, pp. 377-388.

[Kalr94] Kalra, P. and Magnenat Thalmann, N., *Modeling of Vascular Expressions in Facial Animation*, Computer Animation 1994, pp. 50-58.

[Kuri91] Kurihara, T. and Arai, K. A., *Transformation Method for Modeling and Animation of Human Face from Photographs*, Computer Animation 1991, pp. 45-58.

[Lee97] Lee S. W., Karlra P., and Magnenat Thalmann N., *Model Based Face Reconstruction For Animation*, MMM 1997, pp. 323-338.

[Lee95] Lee, Y., Terzopoulos, D., and Waters, K., *Realistic Modeling for Facial Animation*, SIGGRAPH 1995, pp. 55-62.

[Lin00] Lin I. C., Huang C. F., Wu J. C., and Ouhyoung M., *A Low Bit-rate Web-enabled Synthetic Head with Speech-driven Facial Animation*, EG CAS 2000, pp. 29-40.

[Noh01] Noh J. Y., Neumann U., *Expression Cloning*, SIGGRAPH 2001, pp. 277-288.

[Park75] Parke, F. I., *A Model for Human Faces that allows Speech Synchronized Animation*, Computer and Graphics, Pergamon Press, Vol. 1, No. 1, 1975, pp. 1-4.

[Park96] Parke, F. I. and Waters, K., *Computer Facial Animation*, AK Peters, Wellesley, Massachusetts, 1996.

[Pigh98] Pighin, F., Hecker, J., Lischinski, D., Szeliski, R. and Salesin, D., *Synthesizing Facial Expressions from Photographs*, SIGGRAPH 1998, pp. 75-84.

[Plat81] Platt, S. and Badler, N., *Animating Facial Expressions*, SIGGRAPH 1991, pp. 245-252.

[Pres93] Press H. W., Flannery P. B, Teukolsky A. S., and Vetterling T. W., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press, 1993.

[Wate87] Waters, K., *A Muscle Model for Animating Three-Dimensional Facial Expression.* SIGGRAPH 1987, pp. 17-24.

[Wate98] Waters, K., SimpleFace Version 2.0 for OpenGL for Windows95/98 & NT. *http://www.crl.research.digital.com/publications/books/waters/Appendix1/opengl/OpenGLW95NT.html*, 1998.