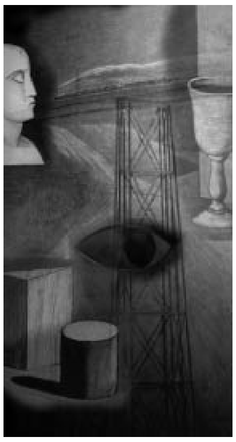
Performance-driven muscle-based facial animation



By Byoungwon Choe, Hanook Lee and Hyeong-Seok Ko*

We describe a system to synthesize facial expressions by editing captured performances. For this purpose, we use the actuation of expression muscles to control facial expressions. We note that there have been numerous algorithms already developed for editing gross body motion. While the joint angle has direct effect on the configuration of the gross body, the muscle actuation has to go through a complicated mechanism to produce facial expressions. Therefore, we devote a significant part of this paper to establishing the relationship between muscle actuation and facial surface deformation. We model the skin surface using the finite element method to simulate the deformation caused by expression muscles. Then, we implement the inverse relationship, muscle actuation parameter estimation, to find the muscle actuation values from the trajectories of the markers on the performer's face. Once the forward and inverse relationships are established, retargeting or editing a performance becomes an easy job. We apply the original performance data to different facial models with equivalent muscle structures, to produce similar expressions. We also produce novel expressions by deforming the original data curves of muscle actuation to satisfy the key-frame constraints imposed by animators. Copyright © 2001 John Wiley & Sons, Ltd.

Received: 13 June 2000; Revised: 4 January 2001

KEY WORDS: performance-driven facial animation; facial expression capture; physically based facial modeling; muscle-based facial model; facial expression editing

Introduction

Synthesizing realistic facial expression remains one of the elusive goals in computer animation. Unlike other parts of human body, the skeletal structure of the face has quite limited degrees of freedom. Nevertheless, the face is equipped with a rich collection of expression muscles to utter speech or generate expressions that reflect the mental state of the person. Considering that facial expressions are the result of very delicate movements, people are surprisingly skilled at recognizing and interpreting them. Therefore, animating the face requires different level of accuracy and realism compared to other parts of the human body.

Performance-driven facial animation[1–4] has been a promising approach for capturing the subtle and intricate movements in facial expressions. While the original performance could be a good source for synthesizing facial expressions, the raw data may contain artifacts, or should be altered to meet the animator's needs. This article is about editing facial expressions captured from live performances.

We choose *muscle actuation* as the parameter to control the facial expression in this work. The effectiveness has been already explored elsewhere.[5,6] An advantage of using muscle actuation parameters is that retargeting of an expression to other faces becomes a trivial job. This is based on our assumption that even though muscle size and layout are different for each individual, people use the same actuation pattern to make a similar expression. Another important advantage is that the muscle parameters can be easily converted to higher-level control parameters such as Facial Action Coding System[7] or MPEG-4 Facial Animation Parameters.[8]

There have been numerous algorithms already developed for editing gross body motion.[9–11] However, there are significant structural and representational

*Correspondence to: Byoungwon Choe, 133-316 Seoul National University Kwanak-gu, Seoul, Korea.
E-mail: drei@graphics.snu.ac.kr

differences between the gross body and the face. Gross body motion, when the body is modeled with rigid links and joints, can be kinematically controlled by manipulating joint angle values. On the other hand, the effect of muscle actuation on the facial expression is *not direct*. To obtain the geometrical shape, we have to simulate the complicated mechanism between muscle actuation and facial surface.

Establishing the relationship between muscle actuation and facial shape is not a trivial task. Therefore, this article devotes a significant portion to establishing the relationship. For the forward relationship (to deform the skin surface from the muscle actuation), we model the skin surface using the finite element model, and place expression muscles beneath the surface. For the inverse relationship (to estimate the muscle actuation parameters from the captured performances), we employ an optimization algorithm on the tracked marker positions. Establishing the inverse relationship is essential to retarget or edit a performance in the muscle parameter domain. Although there were remarkable attempts to extract muscle actuation information from facial performance in the literature of facial expression recognition,[12,13] not much practical result has been reported in facial animation.

Once the forward and inverse relationships are established, retargeting or editing a performance becomes an easy job. First, we extract muscle actuation data from given performance. Then, we apply the actuation data to different facial models to produce a similar expression, or deform the original actuation curves to satisfy the key-expression constraints imposed by animators.

Figure 1 shows the overview of our facial expression synthesis system. Input to the system is the performance recorded on video, which is analyzed to obtain 3D trajectories of the markers placed on the performer's face. The resulting trajectories are processed to estimate the muscle actuation parameters. The parameters are now edited for a new expression and sent to the finite element solver to calculate the shape of the skin surface.

The rest of this paper is organized as follows. The next section reviews related work on facial modeling and animation. The third section describes the modeling of muscle–skin structure. The fourth section explains how we capture expressions and process the data to find muscle actuation parameters. The fifth section describes editing and retargeting of the expressions. The experimental results are shown in the sixth section. Finally, we conclude this paper in the seventh section.

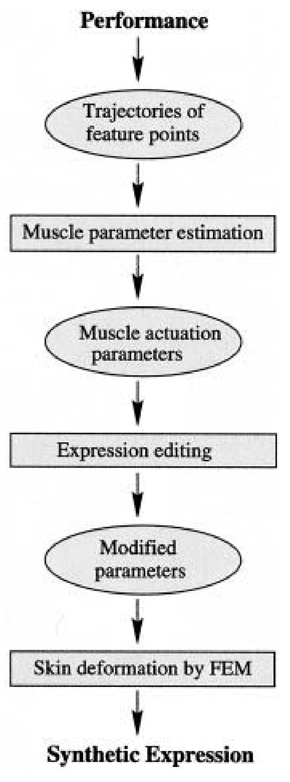


Figure 1. Overview of our facial expression synthesis.

Background

This section reviews three different approaches in facial modeling and animation: performance-driven animation, physically based modeling, and image-based modeling. More topics on facial modeling and animation can be found elsewhere.[14,15] We also introduce several motion editing techniques relevant to our work.

Williams[1] pioneered performance-driven facial animation. He synthesized expressions by changing texture coordinates from the tracked 2D positions of feature points. Guenter *et al.*[2] reproduced the subject's facial expressions by extracting the geometry and texture information from video streams. They produced quite life-like facial expressions. It seemed difficult, however, to modify the original performance or apply the data to different models. Kouadio *et al.*[3] captured live facial expressions to animate a synthetic character in real time by blending previously modeled 3D facial expressions. Terzopoulos and Waters[12] automatically estimated facial muscle contraction parameters from video sequences. Essa *et al.*[13,16] developed a system to estimate muscle actuation corresponding to the skin deformation of a given expression using feedback control theory.

Platt and Badler[17] proposed an abstract muscle fiber structure with skin, muscle, and bone nodes which were connected by springs. Terzopoulos *et al.*[6,18] modeled the face based on facial anatomy and dynamics. They represented the mesh of skin surface by the mass-spring model, and calculated skin deformation due to muscle actuation using the discrete deformable model. With the finite-element facial model, Koch *et al.*[19] predicted the geometry of skin surface due to skull shape change, for plastic surgery.

Pighin *et al.*[20] reconstructed the geometry and texture of an individual face from five photo images of the subject. The result was photo-realistic, showing detailed wrinkles and creases. Lee *et al.*[21] modeled the face of a person from two orthographic images, and simulated aging wrinkles. Blanz and Vetter[22] reconstructed realistic faces from one or two photographs using the database of 3D shapes and textures.

For editing gross body motion, Bruderlin and Williams[9] introduced multi-resolution motion filtering, dynamic time-warping, and motion displacement mapping. Witkin and Popović[11] presented a motion editing technique based on warping of the motion parameter curves. Lee and Shin[10] edited the motion of human-like structure using the hierarchical curve-fitting technique. Rose *et al.*[23] presented a system for real-time motion interpolation. Motion retargeting techniques have been developed to retarget a motion to a character a motion to a character with different body dimensions.[24–26]

Muscle and Skin Model

This section presents how we model the muscle–skin structure of the human face. We represent the skin surface by the finite element model, which was used for free-from shape design[27] and facial surgery simulation.[19] We place expression muscles underneath the skin surface. Since our method computes skin deformation by the finite element method, we had to use a new muscle model different from the one used by Lee *et al.*,[6] in which skin surface is represented by the mass-spring model.

Muscle Model

Expression muscles that produce facial expressions are located within the layer of subcutaneous facia.[28,29] Facial muscles are classified into two kinds: parallel and sphincteral muscles.[5] This section describes our parallel and sphincteral muscle models. It also describes skin deformation at the cheek and chin due to jaw rotation.

Parallel Muscle. We model a parallel muscle with a quadrilateral as shown in Figure 2. It differs from the vector model in the previous work;[5,6] the origin and insertion are represented by line segments ($\overline{p_1p_2}$ and $\overline{p_3p_4}$ respectively) rather than points, and only the vertices within the insertion area are pulled toward the origin when a muscle contracts. The vertices between the origin and insertion, which do not lie in the insertion area, are moved by the force to resist stretching and bending in the finite element surface. We simulate the situation by assuming that the pulling force is toward the origin and the magnitude of the force decreases as the distance between the

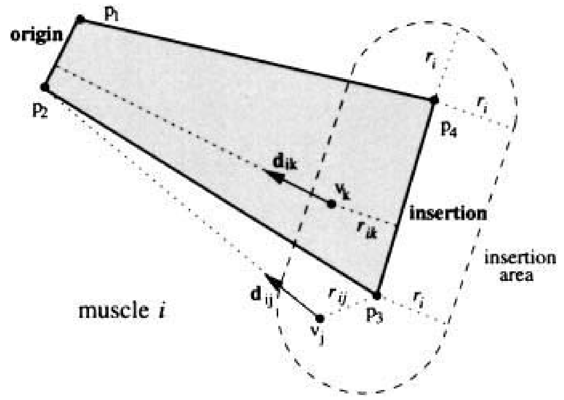


Figure 2. The parallel muscle model, which is represented by a quadrilateral (p_1, p_2, p_3, p_4). $\overline{p_1p_2}$ represents the origin, and $\overline{p_3p_4}$ represents the insertion. When the muscle contracts, the vertices within the insertion area are pulled toward the origin. The insertion area has radius r_i. r_{ij} is the distance from v_j to the closest point in $\overline{p_3p_4}$. $\boldsymbol{d}_{ij}$ is the unit vector from v_j to the closest point in $\overline{p_1p_2}$. Therefore, if the vertex happens to be inside the quadrilateral (v_k in the figure), the distance r_{ik} and the direction $\boldsymbol{d}_{ik}$ are defined toward a point lying within the edge.

surface point and insertion increases; we model the applied force $\mathbf{f}_{ij}$ at a skin vertex u_j contributed from the muscle m_i as

$$\mathbf{f}_{ij} = a_i f_i (1 - \frac{r_{ij}}{r_i})\mathbf{d}_{ij}$$

where a_i is the muscle actuation parameter which is normalized within the range of [0, 1], f_i is the predefined unit muscle force for the muscle m_i, r_i is the radius of the insertion area, r_{ij} is the the distance between the vertex v_j and the insertion of muscle m_i, and $\mathbf{d}_{ij}$ is a unit vector from v_j toward the origin of muscle m_i. When the vertex v_j is not located in the insertion area of muscle m_i, the force from that muscle is zero.

In determining r_{ij} and $\mathbf{d}_{ij}$, the closest points in $\overline{p_3p_4}$ and $\overline{p_1p_2}$ from v_j are used respectively. The force $\mathbf{f}_{ij}$ is given by setting the variable a_i, because f_i, r_i, r_{ij} and $\mathbf{d}_{ij}$ are all predefined or given by the relative position of the origin, insertion, and v_j. If each muscle actuation value a_i is provided, we can calculate the total force applied to a vertex v_j by

$$\mathbf{f}_j = \sum_i \mathbf{f}_{ij} = \sum_i a_i f_i (1 - \frac{r_{ij}}{r_i})\mathbf{d}_{ij} \tag{1}$$

Sphincteral Muscle. Orbicularis oris, a sphincteral muscle which is located around the mouth, operates differently. We assume the force applied to the vertex v_j within the influenced area (Figure 3(a)) is proportional to the muscle actuation, but also depends on the vertex position within the influenced area; i.e., the force applied at v_j is given by

$$\mathbf{f}_{ij} = a_i f_i \sigma(s)\mathbf{d}_{ij} \tag{2}$$

Here, $s = \sqrt{(l_{jx}/g)^2 + (l_{jy}/h)^2}$, where l_{jx} and l_{jy} are the distance of the vertex v_j from the center along the X and Y axes, and g and h are the two radii of the 2D ellipse. The shape of the function σ used in our implementation is plotted in Figure 3(b). $\mathbf{d}_{ij}$ is the unit vector from v_j toward the center of the ellipse.

Since all the forces are headed toward the center, when the orbicularis oris is actuated significantly the lips are pursed up to preserve the volume of the lips. However, our finite element model for skin deformation (see below) does not consider collision between the lips. Therefore, (2) alone cannot simulate the pursing of the lips. To implement the effect, we used the following equation for the third component f^z_{ij} of the force $\mathbf{f}_{ij}$ along the Z axis:

$$\mathbf{f}^z_{ij} = a_i f_i \tau(s)$$

where a_i and f_i are the same values used in (2). The shape of the function τ used for our implementation is shown in Figure 3(c).

There are also two sphincters in the eyes, which operate quite differently from the one around the mouth. Currently, the sphincters in the eyes are not included in our model. Instead, we implement the blinking of eyes kinematically.

Jaw Rotation. When the mandible rotates, forces are applied to a range of vertices that form the jaw. The direction $\mathbf{d}_{ij}$ of the force is perpendicular to both the rotation axis and $\mathbf{e}_j$, which is the vector heading from v_j toward the closest point in the rotation axis (Figure 4). The magnitude of the force is proportional to $|\mathbf{e}_j|$, i.e.,

$$\mathbf{f}_{ij} = a_i f_i |\mathbf{e}_j| \mathbf{d}_{ij} \tag{3}$$

where a_i is the rotation parameter of the jaw (normalized in [0, 1]) and f_i is the predefined unit force of the jaw. Natural opening and closing of the

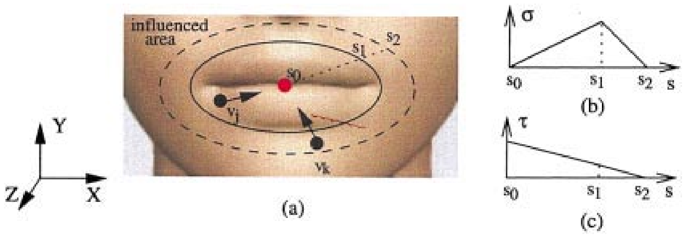


Figure 3. Sphincteral muscle model around a mouth. When the sphincter actuates, the force is applied to the vertices within the influenced area, inside the dashed ellipse. (a) The direction of the force is toward the center. (b) Plotting of the force magnitude values (along X and Y axes) used in our implementation. (c) Plotting of the third component (along Z axis) of the vertex force to simulate the pursing effect.

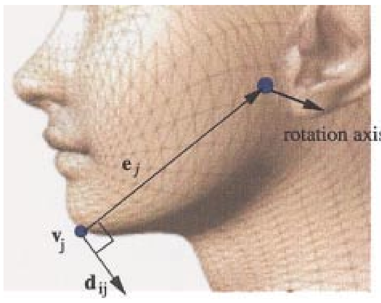


Figure 4. Vertex force due to jaw rotation. When jaw is rotated, a force is applied to the vertex v_j. The force is in the direction of $\boldsymbol{d}_{ij}$, which is perpendicular to both $\boldsymbol{e}_j$ and the rotation axis. $\boldsymbol{e}_j$ is the vector from v_j to the jaw axis. The magnitude of the force is proportional to $|\boldsymbol{e}_j|$.

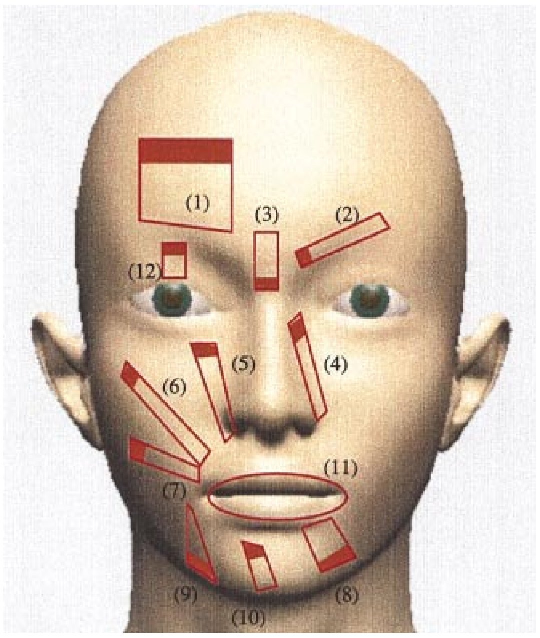


Figure 5. Muscles embedded in our face model. Shaded sides represent the origins. Even though not shown in the figure, all the muscles exist in pairs except for (3) and (11). (1) Frontalis. (2) Corrugator. (3) Procerus. (4) Levator labii superioris alaeque nasi. (5) Levator labii superioris. (6) Zygomatic major. (7) Risorius. (8) Depressor labii inferioris. (9) Triangularis. (10) Mentalis. (11) Orbicularis oris. (12) Orbicularis ocull (Levator palpebrae).

mouth is basically caused by rotation of the mandible. Even though the vertex force in (3) produces reasonable results for the cheek and chin, it does not produce an acceptable result for the shape of the lips. Therefore, we had to put heuristic correction forces near the mouth.

Muscle Layout. Twenty-two expression muscles are embedded in our face model, as shown in Figure 5. The sizes and layout of the muscles might be an important factor. However, finding out such data of living subjects is beyond the current medical technology. Therefore, we used the standard sizes and layout shown in Clemente[28] and Faigin.[29]

Skin Deformation Model

This section formulates skin deformation corresponding to muscle actuation. We model the skin surface by the finite element model with linear elasticity.[19,27] The soft tissue in the human body is known to have visco-elastic non-linear properties.[30] The major disadvantage of the visco-elastic non-linear model is its high computational complexity. In the linear elastic model, we can compute skin deformation in real time by utilizing the linear relationship between muscle force and skin deformation. It also simplifies the formulation of the muscle parameter estimation procedure, described below. Although the linear finite element model does not accurately represent the physical aspects of skin deformation compared to the visco-elastic non-linear model, it still provides a reasonable level of visual realism. We briefly introduce the finite element model used elsewhere,[19,27] and show how deformation of the skin surface from each muscle actuation can be calculated in real time.

Finite Element Model for Skin Surface. The 3D geometry of the face can be represented by a surface of two parameters (u, v) as $\mathbf{w}(u, v) = [x(u, v), y(u, v), z(u, v)]^T$ over a parameter domain Ω. If the surface internally resists stretching and bending, and external force is applied to the surface, we can find the shape of the surface by minimizing the energy functional

$$E_{surface} = \int_{\Omega} \left(\begin{array}{c} \alpha_{11}|\mathbf{w}_u|^2 + 2\alpha_{12}\mathbf{w}_u \cdot \mathbf{w}_v + \alpha_{22}|\mathbf{w}_v|^2 \\ + \beta_{11}|\mathbf{w}_{uu}|^2 + \beta_{22}|\mathbf{w}_{uv}|^2 + \beta_{33}|\mathbf{w}_{vv}|^2 \end{array} - 2\mathbf{f}\mathbf{w} \right) d\Omega \quad (4)$$

where α_{11}, α_{12}, and α_{22} control tension, β_{11} and β_{33} control bending, β_{22} controls twisting, and $\mathbf{f} = \mathbf{f}(\mathbf{w})$ is the external force. The surface is obtained by finite element analysis.

We use a nine-DOF triangular element[27] to describe the finite element patch. Within an element e, we can describe the estimated surface $\widetilde{\mathbf{w}}_e=[\widetilde{\mathbf{x}}(u,v),\widetilde{\mathbf{y}}(u,v),\widetilde{\mathbf{z}}(u,v)]^T$ as

$$\widetilde{\mathbf{w}}_e=\sum_{j=1}^{9}\phi_{e,j}\mathbf{q}_{e,j}=\Phi_e\cdot\mathbf{q}_e \tag{5}$$

where $\phi_{e,j}$ are the shape functions, and ordered into $\mathbf{\Phi}_e=[\phi_{e,1},\phi_{e,2},\ldots,\phi_{e,9}]$, and $\mathbf{q}_{e,j}=[q_x,q_y,q_z]^T$ are unknown weights, and ordered into $\mathbf{q}_e=[\mathbf{q}_{e,1},\mathbf{q}_{e,2},\ldots,\mathbf{q}_{e,9}]$. The matrices for the finite element computation over the parameter domain Ω_e of element e take the following forms. The stiffness matrix $\mathbf{K}_e$ from the energy functional (Equation (4)) is

$$\mathbf{K}_e=\int_{\Omega_e}(\Phi_{e,s}^T\alpha\Phi_{e,s}+\Phi_{e,b}^T\beta\Phi_{e,b})d\Omega_e$$

where

$$\Phi_{e,s}=\begin{bmatrix}\partial\Phi_e/\partial u\\ \partial\Phi_e/\partial v\end{bmatrix}\quad \Phi_{e,b}=\begin{bmatrix}\partial^2\Phi_e/\partial u^2\\ \partial^2\Phi_e/\partial u\partial v\\ \partial^2\Phi_e/\partial v^2\end{bmatrix}$$

$$\alpha=\begin{bmatrix}\alpha_{11} & \alpha_{12}\\ \alpha_{12} & \alpha_{22}\end{bmatrix}\quad \beta=\begin{bmatrix}\beta_{11} & 0 & 0\\ 0 & \beta_{22} & 0\\ 0 & 0 & \beta_{33}\end{bmatrix}$$

The external force matrix $\mathbf{F}_e$ is

$$\mathbf{F}_e=\int_{\Omega_e}(\Phi_e^T\mathbf{f})d\Omega_e \tag{6}$$

We assemble the local matrices $\mathbf{K}_e$ and $\mathbf{F}_e$ into global matrices $\mathbf{K}$ and $\mathbf{F}$. Then, the problem of minimizing $E_{surface}$ of (4) can be written as

$$min(\mathbf{q}^T\mathbf{K}\mathbf{q}-\mathbf{F}^T\mathbf{q}) \tag{7}$$

where $\mathbf{q}$ is assembled from the local weight matrices $\mathbf{q}_e$. Equation (7) is equivalent to solving

$$\mathbf{K}\mathbf{q}=\mathbf{F} \tag{8}$$

for $\mathbf{q}$. Finally, we can calculate the surface $\mathbf{w}$ from $\mathbf{q}$, using the relation in (5).

Calculation of Skin Surface Deformation. External force $\mathbf{F}$ is the sum of geometric constraint force $\mathbf{F}_{init}$ and muscle actuation force $\mathbf{F}_m$ ($\mathbf{F}=\mathbf{F}_{init}+\mathbf{F}_m$). $\mathbf{F}_{init}$ is used to fit the surface into the initial geometry of the face, and is calculated by

$$\mathbf{F}_{init}=\mathbf{K}\mathbf{q}_{init}$$

where q_{init} is estimated from the position and normal of each vertex.[19] The muscle force matrix $\mathbf{F}_m$ is calculated by supplying the values of force f in (6) as described earlier. Let $\mathbf{f}_i$ represent the full actuation of muscle m_i, and a_i be the actuation of muscle m_i. Then, the total force from all the muscle actuation is assembled by

$$\mathrm{f}=a_1\mathbf{f}_1+a_2\mathbf{f}_2+\cdots+a_n\mathbf{f}_n$$

Using (6), we can write muscle force matrix $\mathbf{F}_\mathrm{m}$ as

$$\mathrm{F}_m=\int_\Omega(\Phi^T\mathrm{f})d\Omega=\int_\Omega(\Phi^T\sum_i a_i\mathrm{f}_i)d\Omega$$
$$=\sum_i a_i\int_\Omega(\Phi^T\mathrm{f}_i)d\Omega=\sum_i a_i\mathrm{F}_{m_i}$$

where F_{m_i} is the force matrix when the muscle m_i is fully actuated.

Thus, the linear equation (8) is given by

$$\mathrm{Kq}=\mathrm{F}_{init}+\sum_i a_i\mathrm{F}_{m_i}$$

Let $\mathbf{q}_i$ be the solution of $\mathbf{K}\mathbf{q}_i=\mathbf{F}_{m_i}$. Then, we can write $\mathbf{q}$ by

$$\mathbf{q}=\mathbf{q}_{init}+\sum_i a_i\mathbf{q}_i$$

Therefore, the surface $\mathbf{w}$ is represented by

$$\mathbf{w}=\mathbf{w}_{init}+\sum_i a_i\mathbf{w}_i \tag{9}$$

where $\mathbf{w}_{init}$ is the initial surface geometry, and $\mathbf{w}_i$ is the displacement of the surface due to the contraction of muscle m_i. Therefore, if we calculate each $\mathbf{w}_i$ in the preprocessing step, we can compute skin deformation in real time using (9), when muscle actuation values a_i are given.

Analysis of Facial Performance

This section presents the method to capture the facial performance of an actor, and process the data to obtain muscle actuation values. We first model the 3D geometry of the performer's neutral face using image-based modeling technique,[20] and set up the muscle–skin model introduced earlier. We record a facial performance using video cameras, and track the 3D position of feature points on the face. We analyze the resulting trajectories of feature points to find the muscle actuation parameters.

Tracking 3D Position of Feature Points

We place three digital video cameras in front of the subject's face, and record the performance from three different views. Twenty-four retro-reflective markers are glued to the face to detect explicitly the movement of facial features such as eyebrows or lips. The video cameras are calibrated, and synchronized with each other. Figure 6 shows snapshots taken simultaneously from three video cameras. Sixteen markers are used to capture facial movement, and the other eight markers are used to detect gross motion of the head. We find the 3D position of each marker by performing stereo analysis[2,31,32] on the corresponding points of the three 2D images. We further analyze the resulting 3D trajectories to obtain the muscle actuation parameters, which is described in the following section.

Finding Muscle Actuation Parameters

Alignment of Coordinate Systems. The synthetic model of the neutral face is defined in its own local coordinate system $\{M\}$. The position of feature points from the performance is described in a different coordinate system $\{P\}$. To find muscle actuation parameters, we first have to transform the position of feature points in $\{P\}$ to $\{M\}$. The transform from $\{P\}$ to $\{M\}$ has scale s, rotation $\mathbf{R}$, and translation t. The transform is calculated only once at the first frame of the recorded video. Let the position of the i-th marker (among the 16 markers) at the first frame of the video be $\mathbf{p}_{i,r}$ in $\{P\}$. Then, the transformed position $\mathbf{p}^*_{i,r}$ in $\{M\}$ is given by

$$\mathbf{p}^*_{i,r} = s\mathbf{R}\mathbf{p}_{i,r} + \mathbf{t}$$

We define the *virtual marker* $\mathrm{p}_{i,v}$ to be the position in $\{M\}$ that corresponds to the real marker position in $\{P\}$, which is specified explicitly. Then, we determine s, $\mathbf{R}$, and t, so that they minimize

$$e_p = \sum_i \left|\mathbf{p}_{i,v} - \mathbf{p}^*_{i,r}\right|^2$$

Starting from the initial guess, we iteratively solve the linear least square problem to find s, $\mathbf{t}$, and $\mathbf{R}$.

The above assumes that the facial geometry at the first frame of recorded performance is the same with the pre-modeled computer face. One way to guarantee such a condition is to model the geometry of the face using the cameras calibrated under the same condition in recording the performance. A problem of this approach is that we have to remodel the geometry of the face every time the performance is captured. Another way, which we took in this article, is to depend on the actor's skill to make a consistent neutral face.

Estimation of Muscle Actuation Parameters. Now, we find the muscle actuation parameters. We find the optimal muscle actuation parameters so that the virtual markers follow the trajectories of the real markers in the real performance. Let the displacement of a feature point p_j be $\mathbf{v}_{ij}$ when muscle m_i is fully actuated (i.e., when the muscle actuation parameter is 1). If the actuation of muscle m_i is a_i, then the total displacement $\mathbf{v}_j$ of p_j is given by

$$\mathbf{v}_j = \sum_i a_i \mathbf{v}_{ij}$$

which is derived from (9). We find the muscle actuation parameters $a_1, a_2, \ldots, a_m$ that make v_j as close as possible to the observed displacement $\mathbf{v}^*_j$ from the performance. We calculate a_i by minimizing

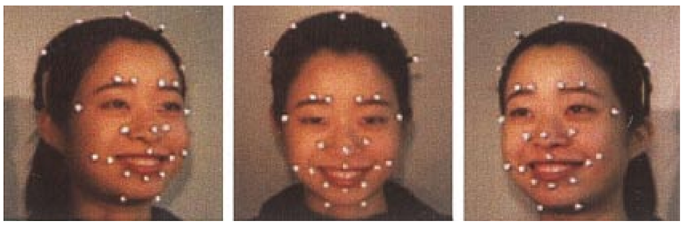


Figure 6. Snapshots taken simultaneously from three video cameras.

$$\begin{aligned} e_a &= \sum_j \left| \mathbf{v}_j^* - \mathbf{v}_j \right|^2 + c \sum_i D(a_i) \\ &= \sum_j \left| \mathbf{v}_j^* - \sum_i a_i \mathbf{v}_{ij} \right|^2 + c \sum_i D(a_i) \end{aligned} \tag{10}$$

where c is a constant, and $D(a_i)$ is the penalty function to restrain a_i to lie in [0, 1], and has the following form:

$$D(a_i) = \begin{cases} -a_i, & a_i > 0 \\ a_i - 1, & a_i > 1 \\ 0, & otherwise \end{cases}$$

We use the steepest descent method[33] to solve the minimization problem (10).

Facial Expression Editing

Now, we can edit the muscle actuation data extracted earlier to produce a novel expression. We can also apply the same actuation data to different facial models to produce similar expressions.

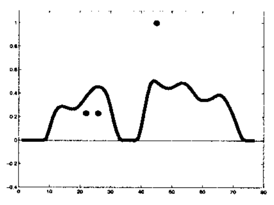
Expression Displacement Mapping

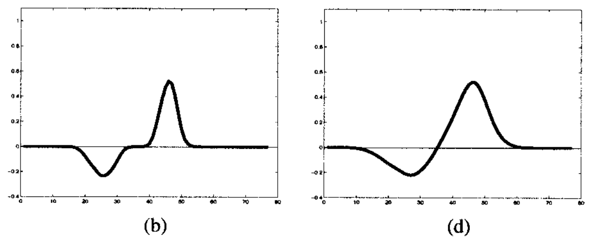
Animators may want to modify captured expressions by specifying several *key expressions*. Our expression displacement mapping algorithm can accommodate such needs. We allow two ways to set up key-frame constraints. Animators can (1) directly set the values of muscle actuation parameters at desired time ticks, or (2) control the position of the feature points. In the latter case, the position inputs are internally converted to muscle actuation parameters.

The specified key-frame constraints are now realized by adjusting the original actuation parameter curves. We impose the following properties to our solution:

1. Key-frame constraints should be satisfied.
2. Delicate movement (i.e., high-frequency component) of the original data should be preserved.
3. Modification should be local around each key-frame, and the user should be able to control the duration (around the key-frame) that is modified by the key-frame constraint.

For each muscle actuation parameter, we fit the discrepancy caused by key-frame constraints into a B-spline curve. The fitted curve is the displacement


(a)


(b) (d)

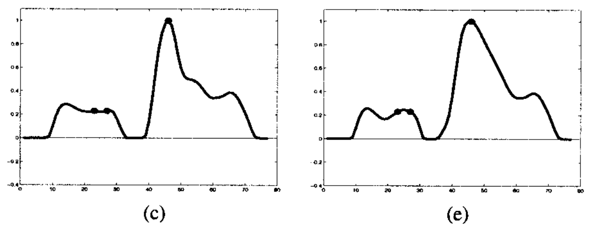

(c) (e)

Figure 7. Expression displacement mapping. (a) The original data with key-frame constraints (three dots). (b) Displacement curve with knot-spacing 4. (c) Deformed data obtained by the displacement in (b). (d) Displacement curve with knot-spacing 8. (e) Deformed data obtained by the displacement in (d). The original curve is modified over a wider region than in (c).

map.[9] We obtain the new parameter curve by adding the displacement map to the original parameter curve. Since the displacement map is a B-spline, it consists of low-frequency components which do not alter the delicate movement of the original data (Property 2 is satisfied). In fitting the discrepancy, we can control the affected region by adjusting knot-spacing as shown in Figure 7, which realizes Property 3.

Expression Retargeting

Retargeting[1] an expression to a different character would not be an easy job if vertices, feature points, or the weights for blending static expressions were used

[1]Motion retargeting in the gross body animation is different from expression retargeting discussed in this section. In gross body animation, retargeting can be formulated as a constrained optimization problem with end-effector constraints,[25] and is much more complicated than in expression retargeting.

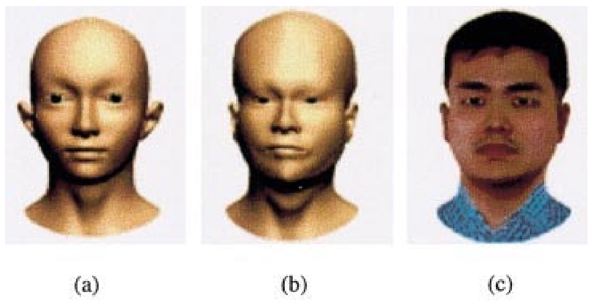


Figure 8. Modeling the geometry of an individual face from the generic model. (a) The generic model. (b) The geometry of an individual face adapted from the generic model. (c) Final result rendered with textures.

as control parameters. When we control expressions with muscle parameters, retargeting of an expression becomes trivial if we assume the same muscle actuation profile generates a similar expression. If the two faces have equivalent muscle structure, we can simply apply the muscle actuation parameters of the original character to animate the target face. The minor differences in the source and target expressions are caused by variations in the muscle layout and attributes such as muscle size or unit muscle force.

Expression retargeting is particularly useful for character animation (as shown in Figure 10(d)). Generating dynamic facial expressions of a cartoon-like character is very difficult when it is done manually, especially when the character utters speech. Our expression retargeting can automatically map the original performance of an actor to the character with the mouth movement synchronized with the voice.

Experiments

We implemented our algorithm on a PC platform, and produced the the video clips located in http://graphics.snu.ac.kr/research/pmface/. The following description refers to those video clips.

For the experiments, we modeled the faces of *H. Park*, *C. Kim*, and *S. Shin* using an image-based modeling technique. Figure 8 shows the process of modeling the face of *H.* Park. Starting from the geometry of a generic face model (Figure 8(a)), we adjust the vertex positions so that the geometry is coherent with the photographs, and recover texture information.[20] All three faces were created from the same generic model, and thus had the same geometrical topology and muscle structure.

We captured Park's 'surprise' expression and reproduced it by following the procedures (1) feature point tracking, (2) muscle actuation parameter estimation, and (3) finite element computation. Intrinsically, our reproduction does not make an exact recovery of the original expression (see 'Finding Muscle Actuation Parameters', above). Moreover, errors can be introduced at different stages: modeling, feature point tracking, muscle actuation parameter estimation, and/or finite element computation. Nevertheless, the

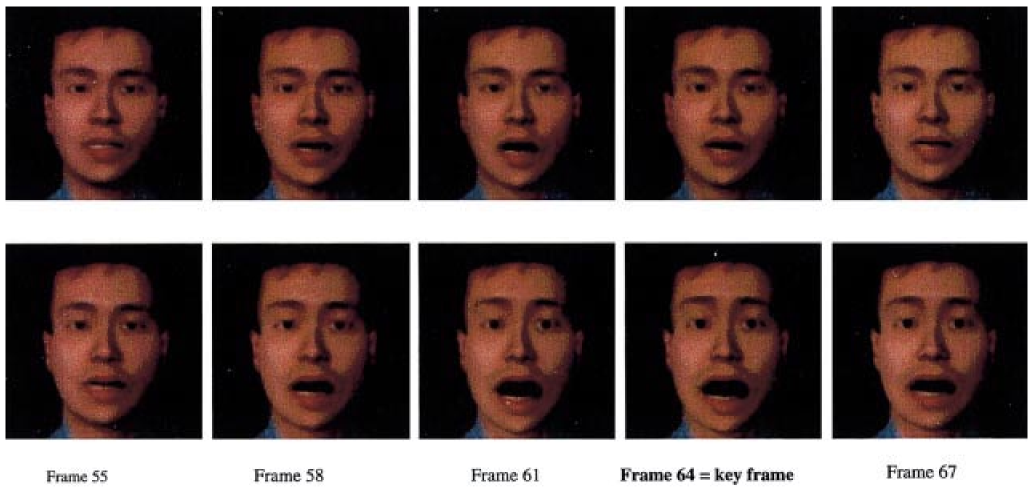


Figure 9. The result of expression displacement mapping. Upper: original expression. Lower: edited by the key-expression constraints to raise eyebrows and open the mouth wider.

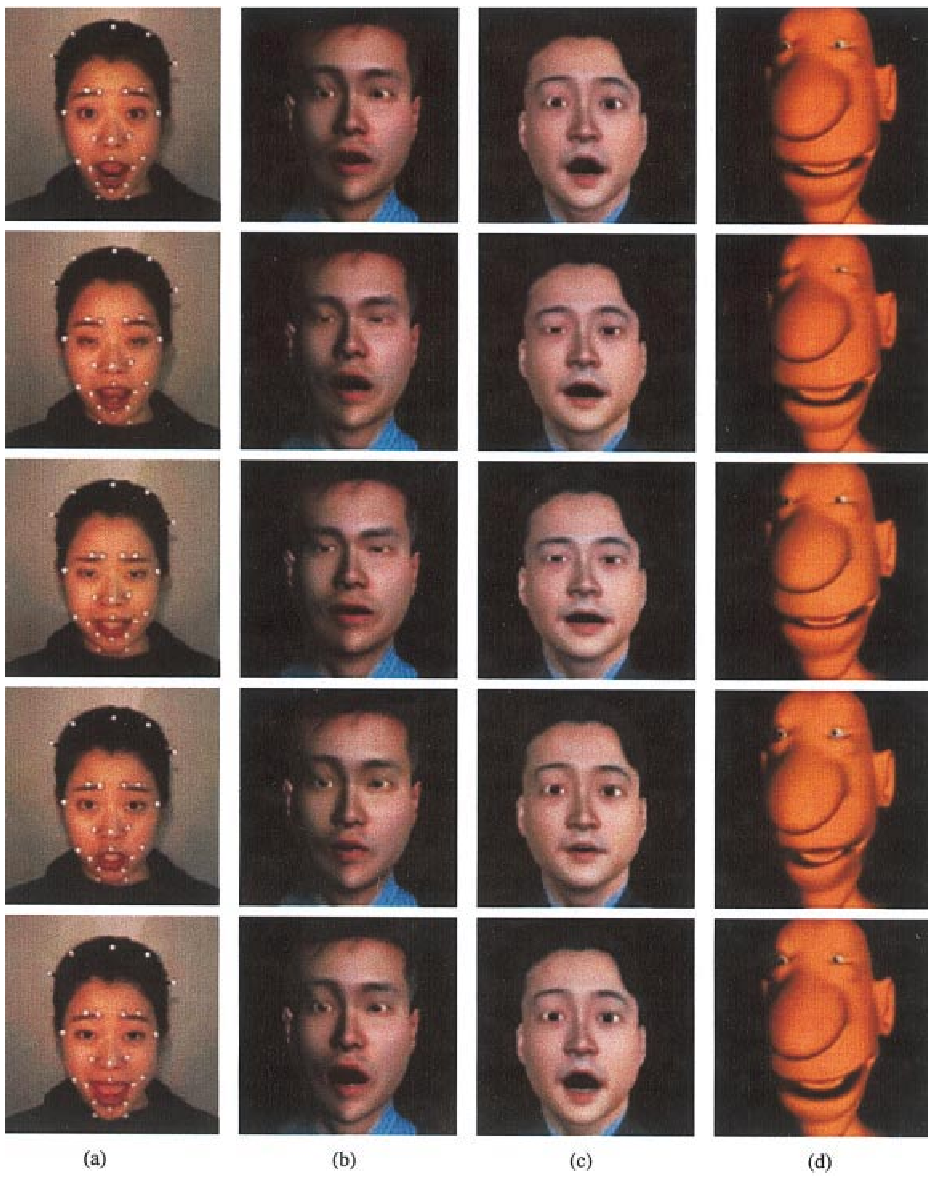


Figure 10. Retargeting facial expressions to different characters. (a) Original performance of Shin. (b) Retargeting to Park. (c) Retargeting to Kim. (d) Retargeting to Big Nose.

reproduced result showed that the muscle parameters were quite effective for capturing the essence of human expressions.

We also captured Park's uttering of 'muscle-based editing of facial performances'. After reproducing it, we set two key-expressions: Frames 64 and 103. At Frame 64, we edited the muscle parameter values to raise the eyebrows and open the mouth wider. At Frame 103, the eyebrows were lowered to generate an angry face. Then, the muscle actuation curves were adjusted to pass through the given key-frame constraint points. In this experiment, we used B-spline curves with knot spacing 8. Figure 9 compares the original and the new expressions after the expression displacement mapping.

We took an expression sample from Shin while she was performing a short script. We extracted the muscle parameters from the recording, and retargeted the result to the models of Park and Kim. We did not directly capture the region of eyes from the original performance. The blinking of the eyes shown in the demo video clips was the result of applying manually expression displacement mapping on the *orbicularis oculi* muscle. We also retargeted the performance to Big Nose with the same muscle structure but different facial geometry. To produce exaggerated expressions, we used a different method to implement the skin deformation of Big Nose; the skin deformation due to the actuation of each muscle was modeled manually by an animator, rather than using the finite element solver. Even though the retargeted expressions were different from the original performance in detail, we could observe that they gave quite similar impression. Figure 10 shows several snapshots taken from the retargeted results.

Conclusion

We have presented a facial animation system to synthesize facial expressions by editing captured performances. As control parameters, we used the actuation of 19 parallel and three sphincteral muscles, and a jaw rotation value. Several algorithms to edit captured motion were developed for gross body animation, which were not directly applicable to facial animation due to structural and representational differences. Therefore, a significant portion of this paper was devoted to establish the relationship between muscle actuation and skin surface deformation. We used the finite element method to model skin surface deformation from muscle contraction. For the inverse relationship to estimate the muscle actuation profile from a performance, we applied an optimization algorithm on the trajectory of feature points on the face. When the recording of a facial performance was given as the input, we first estimated the muscle actuation values over time, then edited the resulting muscle parameter curves, and finally sent the actuation data into the FEM solver to synthesize expressions.

This work was a practical attempt to bridge the gap between performance-driven animation and physically based modeling. Experimental results showed that our system reproduced the captured expressions on the muscle-based computer model quite well. Moreover, we could apply well-known editing techniques in gross body animation to facial animation due to the inverse relationship from muscle actuation to skin deformation.

Although we could capture the essence of facial expressions, we need to develop a method to reproduce facial movement more accurately. Capturing the delicate movement in the vicinity of eyes and lips is extremely important in synthesizing realistic expressions. We also need to develop more sophisticated editing algorithms to serve the various needs of animators.

ACKNOWLEDGEMENTS

This work was supported by the Brain Korea 21 Project. This work was also supported by ASRI (the Automation and Systems Research Institute) at Seoul National University.

References

1. Williams W. Performance-driven facial animation. In *Computer Graphics (SIGGRAPH'90 Proceedings*, Dallas, Texas), Vol. 24, August 1990, pp 235–242.
2. Guenter B, Grimm C, Wood D, Malvar H, Pighin F. Making faces. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH. Addison-Wesley Reading, MA, July 1998, pp 55–66.
3. Kouadio C, Poulin P, Lachapelle P. Real-time facial animation based upon a bank of 3D facial expressions. In *Proceedings of Computer Animation '98 Conference*. IEEE Computer Society Press, Las Alamitas, CA, 1998.
4. Pighin F, Szeliski R, Salesin DH. Resynthesizing facial animation through 3D model-based tracking. In *ICCV'99 Conference Proceedings*, Kerkyra, Greece, 1999.
5. Waters K. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (SIGGRAPH'87 Proceedings*, Araheim, California), Vol. 21, July 1987, pp 17–24.
6. Lee Y, Terzopoulos D, Waters K. Realistic face modeling for animation. In *SIGGRAPH'95 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH. Addison-Wesley, Reading, MA, August 1995, pp 55–62.
7. Ekman P, Friesen WV. *Facial Action Coding System*. Consulting Psychologists Press, Palo Alto, CA, 1978.
8. MPEG4 Systems Group. Text for ISO/IEC FCD 14498-1 systems. *ISO/IEC JTC1/SC29/WG11 N2201*, 15 May 1998.
9. Bruderlin A, Williams L. Motion signal processing. In *Proceedings of SIGGRAPH 95*, Los Angeles, CA, August 1995, pp 97–104.
10. Lee J, Shin SY. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH'99*, Los Angeles, California, August 1999.
11. Witkin A, Popovic Z. Motion warping. In *Proceedings of SIGGRAPH'95*, Los Angeles, CA, August 1995, pp 105–108.
12. Terzopoulos D, Waters K. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1993; **15**(6): 569–579.

13. Essa IA, Pentland AP. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1997; **19**(7): 757–763.
14. Parke FI, Waters K. *Computer Facial Animation*. AK Peters, 1996.
15. Ekman P, Huang T, Sejnowski T, Hager J (eds). Final report to NSF of the planning workshop on facial expression understanding. Technical Report USCS, CA 94143, National Science Foundation, Human Interaction Lab., 1993.
16. Essa I, Basu S, Darrell T, Pentland A. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of Computer Animation'96 Conference*, Geneva, Switzerland, June 1996.
17. Platt SM, Badler NI. Animating facial expression. *Computer Graphics* 1981; **15**(3): 245–252.
18. Terzopoulos D, Waters K. Physically-based facial modelling, analysis, and animation. *Journal of Visualization and Computer Animation* 1990; **1**: 73–80.
19. Koch RM, Gross MH, Carls FR, von Büren DF, Fankhauser G, Parish Y. Simulating facial surgery using finite element methods. In *SIGGRAPH'96 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH. Addison-Wesley Reading, MA, August 1996, pp 421–428.
20. Pighin F, Hecker J, Lischinski D, Szeliski R, Salesin DH. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH'98 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH. Addison-Wesley, Reading, MA, July 1998, pp 75–84.
21. Lee W-S, Wu Y, Magnenat-Thalmann N. Cloning and aging in a vr family. In *Proceedings of IEEE VR'99 (Virtual Reality)*, Houston, TX, 13–17 Mach 1999.
22. Blanz V, Vetter T. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH'99*, Los Angeles, CA, Angust 1999, pp 187–194.
23. Rose C, Cohen MF, Bodenheimer B. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics & Applications* 1998; **18**(5): 32–40.
24. Hodgins JK, Pollard NS. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH'97*, Los Angeles, CA, August 1997; pp 153–162.
25. Gleicher M. Retargeting motion to new characters. In *Proceedings of SIGGRAPH'98*, Orlando, FL, July 1998, pp 33–42.
26. Choi K, Ko H. On-line motion retargetting. In *Pacific Graphics '99 Conference Proceedings*, Seoul, Korea, October 1999.
27. Celniker G, Gossard D. Deformable curve and surface finite-elements for free-form shape design. In *Computer Graphics (SIGGRAPH'91 Proceedings)*, Vol. 25, July 1991, pp 257–266.
28. Clemente CD. *Anatomy: A Regional Atlas of the Human Body* (2nd edn). Urban & Schwarzenberg, 1981.
29. Faigin G. *The Artist's Complete Guide to Facial Expression*. Watson-Guptill, 1990.
30. Fung YC. *Biomechanics: Mechanical Properties of Living Tissues* (2nd edn). Springer, Berlin, 1993.
31. Faugeras O. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.
32. Haralick RM, Shapiro LG. *Computer and Robot Vision*. Addison-Wesley, Reading, MA, 1993.
33. Avriel M. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, Englewood Cliffs, No, 1976.

Authors' biographies:

Hyeong-Seok Ko is an assistant professor in the School of Electrical Engineering at Seoul National University. He received his B.A. and M.S. in Computer Science from Seoul National University in 1985 and 1987, respectively, and Ph.D. in Computer Science from the University of Pennsylvania in 1994. After he received his Ph.D., he joined the Computer Science Department at the University of Iowa during the academic years 1994 and 1995 as an assistant professor. Since then, he has been working at Seoul National University. Currently, he is the director of Seoul National University Human Animation Center. His research interests include gross-body motion editing, facial animation, hairstyle modeling and animation, body deformation, cloth simulation, and virtual reality applications.

Byoungwon Choe is a Ph.D. student in the Graphics and Media Lab. in the School of Electrical Engineering at Seoul National University, Seoul, Korea. He received his B.S. and M.S. in Electrical Engineering from Seoul National University. His current research interests include facial animation and gross body animation.

Hanook Lee received a Master's degree in 1999 from the School of Electrical Engineering at Seoul National University. He is currently working for Kumyang Co. Ltd. His research interests are facial animation, simulation games, and development of motion capture equipment.