Online Expression Mapping for Performance-Driven Facial Animation

Hae Won Byun

School of Media & Information, Sung Shin Woman's University,
169-1 Dongsun-dong 2, Sungbuk-gu, Seoul, Republic of Korea
hyewon@sungshin.ac.kr

Abstract. Recently, performance-driven facial animation has been popular in various entertainment area, such as game, animation movie, and advertisement. With the easy use of motion capture data from a performer's face, the resulting animated faces are far more natural and lifelike. However, when the characteristic features between live performer and animated character are quite different, expression mapping becomes a difficult problem. Many previous researches focus on facial motion capture only or facial animation only. Little attention has been paid to mapping motion capture data onto 3D face model.

Therefore, we present a new expression mapping approach for performance-driven facial animation. Especially, we consider online factor of expression mapping for real-time application. Our basic idea is capturing the facial motion from a real performer and adapting it to a virtual character in real-time. For this purpose, we address three issues: facial expression capture, expression mapping and facial animation. We first propose a comprehensive solution for real-time facial expression capture without any devices such as head-mounted cameras and face-attached markers. With the analysis of the facial expression, the facial motion can be effectively mapped onto another 3D face model. We present a novel example-based approach for creating facial expressions of model to mimic those of face performer. Finally, real-time facial animation is provided with multiple face models, called "facial examples". Each of these examples reflects both a facial expression of different type and designer's insight to be a good guideline for animation. The resulting animation preserves the facial expressions of performer as well as the characteristic features of the target examples.

1 Introduction

Recently performance-driven facial animation has become popular for on-line animation. Its basic idea is capturing the facial motion from a real performer and adapting it to a virtual character. The performance animation not only makes it possible to acquire highly natural facial motion but also automates facial motion generation with little help from animators. Being difficult with conventional approaches, one can generate time-varying expressions such as lip-synch and eye-blink effectively with performance animation.

In on-line performance-driven facial animation, it is required to capture facial expressions in real time. For a live performer to feel comfortable in making expressions, it is also desirable, if not required, to avoid any devices such as a head-mounted camera

L. Ma, R. Nakatsu, and M. Rauterberg (Eds.): ICEC 2007, LNCS 4740, pp. 324–338, 2007.

and face-attached markers. These constraints on facial expression capture enforce additional difficulties: The performer naturally moves his/her head to express emotions while making facial expressions according to an animation script. Without a head-mounted camera, one needs to track the position and orientation of performer's head for more accurate facial expression capture. Moreover, without any markers attached on the face, extra effort is needed to track the features of the face that characterize facial expressions. The final difficulty comes from the real-time constraint, that is, to capture facial expressions in real time while addressing the former two difficulties.

The previous approaches for facial expression capture have tried to track the position of markers attached on a face performer [9,24]. To promote the convenience of face performer, facial feature extraction techniques without markers have been proposed in computer vision [6,7,22]. These researches are mostly used for face detection, face recognition, or facial expression analysis. Thus, they must cope with large variations in the appearance of facial features across rather general subjects as well as the large appearance variability of a single subject caused by changes in lighting. In this field, the real time constraint is not at issue. However, in the case of facial expression capture for performance-driven animation, it is required to track the facial features, in real time, for only a small number of specific persons who are employed as face performers.

Real-time facial animation is essential for on-line performance-driven facial animation. Multiple face models, called facial examples, are widely used for real-time facial animation. Those facial examples comprise a facial expression database from which we select appropriate face models to blend and to deform. However, a facial example consists of a set of vertices with few handles to control such geometric operations except the vertices themselves. To achieve smooth local deformation and non-uniform blending without any tools, one would have to deal with every vertex of facial examples involved in these operations. Due to its capability of local control on facial features, "wires"[20] is known as a popular tool for facial animation. However, we cannot expect that a designer does necessarily employ wire deformation to model a example of facial expression.

Those facial examples have been hardly used in conjunction with performance-driven facial animation. In order to combine an example-based facial animation and expression capture approach, we should address how to blend the facial examples according to the expression of face performer. The combined approach has the advantage that leverages the usefulness of facial examples, while gaining the efficiency of facial expression capture. The method is effective even when the size and characteristics of the face of performer is quite different from that of face model. Furthermore, attenuating or exaggerating an expression is also possible by making artificial facial examples with those expressions.

In this thesis, we present a new scheme for on-line performance-driven facial animation. As depicted in Figure 1, we address following three issues: First, we present a comprehensive solution for real-time facial expression capture from a stream of images without any special devices. The performer is allowed to move the head as long as all the facial features are observable from a camera. Next, in order to facilitate real-time expression mapping, we address how to blend facial examples with a set of wires

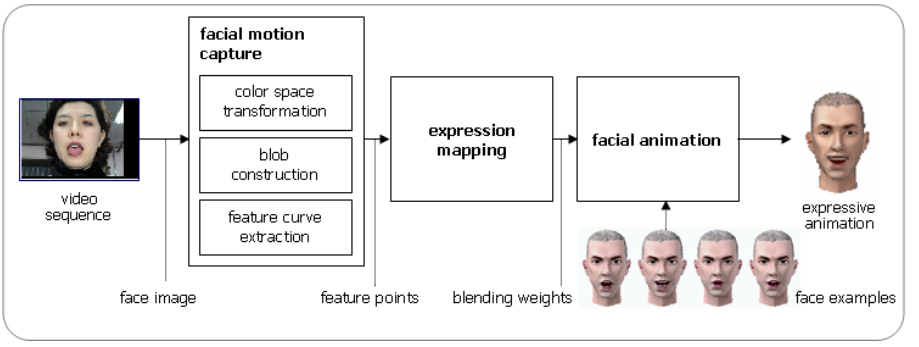

Fig. 1. The overall structure of on-line performance-driven facial animation

and deformation parameters extracted from a facial example. Finally, we propose a novel approach for real-time facial animation with facial examples represented by wire curves.

The remainder of this thesis is organized as follows: In Chapter 2, we explain the overall structure of on-line performance-driven facial animation. In Chapter 3, we introduce the first step in detail, that is, how to extract facial expression. To adopt facial examples for real-time facial animation, we present an expression mapping scheme for facial examples and their effective representation in Chapter 4. The step for facial animation is described in Chapter 5. We demonstrate some experimental results in Chapter 6. Finally, conclusion and future works are given in Chapter 7.

2 Related Work

There are rich results on facial expression capture. We specifically refer those that are directly related to our work. Williams[24] proposed an approach to capture the facial features with markers attached to the feature points on the face of a live performer. Due to its efficiency and robustness, this approach is widely used in practice. However, the markers not only cause discomfort to the performer but also are hard to attach on some facial features such as eyelids.

Terzopolous and Waters[22] adopted an active contour model called “snakes” presented by Kass *et. al.*[13] to track the outlines of facial features highlighted with special makeup. Thalmann *et. al.*[8] and Oliver *et. al.*[16] extracted facial features directly from an input image without any markers. Exploiting anthropometric knowledge, they were able to obtain geometric information such as the width, height, and area of a feature region. This type of approaches is simple and efficient. Instead of finding the outlines of the facial features, Essa *et. al.*[7] proposed a 3D model-based approach for tracking facial features. They used the optical flow field to displace the vertices of 3D models. DeCarlo *et. al.*[6] deformed a 3D facial model to produce a least-squares optical flow solution while relaxing constraints with an extended Kalman filter. Black and Yacoob[2] used locally parametrized models for recovering and recognizing the non-rigid and articulated motions of human faces.

For real-time facial animation, blending multiple face models with different expressions is popular [3,17,10,19,26,5]. Blanz et al.[3] and Pighin et al.[17] proposed an automatic face modeling technique by linearly blending a set of example faces from a database of scanned 3D face models. To avoid unlikely faces, they restricted the range of allowable faces with constraints derived from the example set. For local control on each facial feature, those approaches allow interactive segmentation of a face into a set of regions to assign a proper blending factor to every vertex. Joshi et al.[10] adopted an automatic segmentation method to learn the local deformation information directly from the multiple face models. Chuang et al.[5] extends blendshape retargetting technique to include subsets of morph targets.

Other alternatives are based on deformation techniques. Thalmann et al.[12] employed FFD to simulate the muscle action on the skin surface of a human face. Kahler et al.[11] and Terzopoulos et al.[14] proposed a physically-based method for skin and muscle deformation to enhance the degree of realism over purely geometric techniques. Guenter et. al.[9] and Zhang et. al. [25] used facial motion data captured from real actors to deform face models. Marschner et al.[15] computed the displacements of control points for a specific face model from the movements of sample points on a face performer by solving a system of linear equations in the least squares sense.

3 Facial Motion Capture

We describe how to track the facial features in real time without any devices such as face-attached markers and head-mounted cameras. We assume a stream of images is captured from a single camera with known parameters located at a given position with a fixed orientation. Feature tracking consists of three major tasks: color space transformation, blob construction, and feature curve extraction. Considering that each feature appears in an image as a blob[1,16], that is, a set of pixels with similar properties, we track the movement of the blob to extract the feature. The first task is for enhancing the features for robust tracking. In the second task, we first exploit anthropometric knowledge to confine a blob within a rectangle and then explore the rectangle for the blob. This accelerates constructing the blob, from which we extract its outline in the final task using snakes[13]. The outline is represented as a cubic Bezier curve with small number of control points. We describe each of these tasks in detail.

3.1 Color Space Transformation

For robust feature extraction, we transform the color space of the input image from the RGB model to a model in which the facial features such as eyelashes, eyebrows, nostrils, and lips are significantly distinguishable from their background, that is, the skin. Color spaces including the chromatic color model were employed for robust facial expression capture[1,16]. Based on statistical learning, this model is useful to detect the lips and the skin robustly.

To enhance the facial features, we design a new color transformation function from RGB values to gray-scale values(see Figure 2). By making observation of the face performers, we conceive that the skin has low values of the magenta (M) and black (K) channels in the CYMK color model. A low intensity (V) value of the HSV color model

Fig. 2. Proposed color transformation:(left)Original Image (right)The image obtained by color transformation

is observed for the pixels in dark features such as eyebrows, eyelashes, and nostrils. Moreover, the portion of the hue (H) band occupied by the color of lips is fairly different from that of the skin. Therefore, we use those four components to emphasize the features in an image.

With our transformation function, the intensity $I(u, v)$ of a pixel (u, v) is defined as follows:

$$I(u, v) = w_1 M(u, v) + w_2 K(u, v) + w_3 V(u, v) + w_4 G(H(u, v)). \quad (1)$$

Here, G is a function which has high values at a range of hue values which are similar to those of lips, and has very low values, otherwise. The terms, w_i, $1 \leq i \leq 4$ are weights for four components, M, K, V, and H. Here, w_3 is negative while the others are positive, since pixels in features have lower V values compared to those in the skin. We may further emphasize the features by a contrast enhancement function C. This function is used to amplify lighter pixels and to attenuate dimmer pixels. Thus, by applying this function on the intensity, that is, the result of Equation, we can make the pixels in the feature regions brighter and those of the skin darker.

The weights w_i, $1 \leq i \leq 4$ are varied according to both lighting condition change and skin color variation. In order to estimate the weights, we employ a three-layered neural network . A neural network which has learned several patterns of facial images can estimate the weights for the four color components M, K, V, H in Equation (1) from the RGB color values of facial features in the captured image. The input layer of neural network gets 11 representative intensity values for subareas of each facial feature(skin, eyes, eyelids, nostrils, a mouth, and so on). Each representative value is computed as the average of all the pixel values in the subarea which contains each facial feature in the image. The output layer consists of 4 units specifying the weights that we wanted to estimate in Equation (1). The hidden layer has 10 units, decided empirically, of a sigmoid function to effectively model the non-linear capacity. Once trained for each puppeteer, the training data can be used for other sessions of the same puppeteer without re-training. Whenever we capture facial motion, we can obtain the weights in real time by evaluating the trained neural network with back propagation.

3.2 Blob Contruction

A blob is said to be a set of connected pixels in an image that share similar visual properties such as color and intensity. Facial features such as eyes, lips, eyebrows, and etc. are normally projected onto the image as distinct blobs. By constructing those blobs properly, we can estimate the facial features from the image at each frame. In order to

accelerate blob construction, we confine a blob in a rectangle using anthropometric knowledge such as the relative positions of facial features and their size. A similar idea is used in Thalmann *et al.*[23]. Under the assumption that head movement is allowed as long as no blobs disappear in an image, we empirically determine the size of the rectangle for each feature. Initially, the edges of the rectangle are parallel with either the x-axis or y-axis of the global coordinate.

Given the rectangle containing a feature, we employ a blob growing algorithm in [1,16] to construct the blob. Initially, we sample a small number of pixels in the rectangle, of which intensities are above a threshold, as the seed points for blob growing. From each of those seeds, the algorithm searches an area within a given radius and collects the neighborhood pixels which have greater intensity than the threshold. For each collected yet not expanded pixel, this process is repeated until no such pixels remain. Ideally, these pixels would comprise a single connected component. However, in practice, the pixels give multiple connected components due to threshold and noise. Each of those components form a region that is either a false blob or a subset of a blob. To reduce the influence of threshold and noise, we apply morphological operations such as dilation and erosion[18] to connect the separate regions and eliminate their protrudent features. We finally take the largest region within each rectangle as a feature blob.

3.3 Feature Curve Extraction

In order to extract the outlines of features, we employ snakes as proposed by Kass *et al.*[13]. Snakes are energy-minimizing spline curves under the influence of three (possibly conflicting) forces: internal force, image force, and constraint force. Due to its high degrees of freedom, the snake may snap to unwanted boundaries. To avoid this problem, we need to design the constraint force carefully. Therefore, we remove the internal force from our formulation, differently from the original version of snakes. Instead, we employ cubic Bezier curves with a small number of control points to represent snakes. The outlines of facial features are so simple that they can be well represented by such curves. Moreover, the Bezier curve is infinitely differentiable within itself, and is therefore continuous to any degree. The property of Bezier curve guarantees its smoothness. This simplification increases time efficiency and robustness while sacrificing some flexibility that is not necessarily required for our purpose. In practice, we employ cubic Bezier curves with four control points.

The energy function of our contour model consists of two terms:

$$E(\mathbf{v}) = \int_0^1 E_{image}(\mathbf{v}(\mathbf{s})) + E_{con}(\mathbf{v}(\mathbf{s}))ds, \tag{2}$$

where E_{image} and E_{con} are respectively the energies due to the image force and the constraint force, and $\mathbf{v}(s)$ is a 2D cubic Bezier curve representing the contour of the feature. Therefore, by minimizing $E(\mathbf{v})$, we compute the unknown control points of the 2D curve $\mathbf{v}$.

The energy E_{image} is an edge detecting function, that is,

$$E_{image}(\mathbf{v}(\mathbf{s})) = -w_1|\nabla I(u,v)|^2. \tag{3}$$

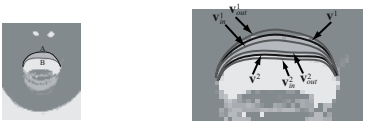

Fig. 3. Two candidates and offset curves:(a)Two candidate contours of the upper lip (b)Offset curves

Here, w_1 is a constant weight value. Large positive values of w_1 tend to make the snake align itself with sharp edge in the image. (u, v) is a point on a 2D cubic Bezier curve $\mathbf{v}(s)$ and $\nabla I(u, v)$ is the gradient at a point (u, v), that is, $\nabla I(u, v) = (\frac{\partial I(u,v)}{\partial u} \quad \frac{\partial I(u,v)}{\partial v})$, and $I(u, v)$ is obtained from Equation (1). This energy function makes the curve $\mathbf{v}$ be attracted to the contour of a blob with large image gradients, or the outline of a feature. However, using only image gradients may cause an unwanted result. For example, as shown in Figure 3, we cannot discriminate the upper curve (A) and lower curve (B) with image gradients alone.

We resolve this problem by employing the constraint energy together with simple upper and lower offset curves as illustrated in Figure 3. Suppose that we want to extract the upper curve (A). Provided with the up-vector of the head, for example, we make those offset curves by slightly shifting the pair of middle control points of each feature curve of lips in the opposite directions with respect to the up-vector. For each feature curve, one of its offset curves is supposed to lie inside the feature while the other outside. An offset curve of a feature curve $\mathbf{v}(s)$ is said to be its inner curve $\mathbf{v}_{in}(s)$ if it is supposed to lie in the corresponding feature. Otherwise, it is said to be its outer curve $\mathbf{v}_{out}(s)$ (see $\mathbf{v}^1$ in Figure 3). Let $I(\mathbf{v}_{out}(s))$ and $I(\mathbf{v}_{in}(s))$ be the intensity of $\mathbf{v}_{out}(s)$ and that of $\mathbf{v}_{in}(s)$, respectively. Because of the color transformation in Section 3.1, a point in a feature region has a high intensity value, and that in the skin has a low value. Given $I(\mathbf{v}_{out}(s))$ and $I(\mathbf{v}_{in}(s))$, the constraint energy of the feature curve $\mathbf{v}(s)$ is defined:

$$E_{con}(\mathbf{v}(s)) = w_{out} I(\mathbf{v}_{out}(s)) - w_{in} I(\mathbf{v}_{in}(s)), \tag{4}$$

where w_{out} and w_{in} are positive constants providing the relative weighting of the intensity terms. As illustrated in Figure 3, with w_{in} sufficiently greater than w_{out}, E_{con} is positive for a curve ($\mathbf{v}^2$ in the figure) that is not properly located, but negative for a properly located one ($\mathbf{v}^1$).

To make an initial guess of the outline of a feature, we first scale the bounding box of each feature, so that it tightly bounds the feature blob. We use the boundary of this box as the initial guess of the feature curve at the current frame. To minimize the total energy function $E(\mathbf{v})$, we adopt the downhill method which uses the gradient of the energy function. The local minimum problem can be alleviated by blurring the image intensity.

4 Expression Mapping

We extracted time-varying movements of feature points from a face performer in the previous capture stage. In this section, given the movements of feature points, our objective is to generate a facial motion for a 3D face model in real time, so as to mimic

the expression of performer. To create facial animation, we adopt an example-based approach to blend facial examples in accordance with their contributions to synthesize the resulting expression[4,10,19,26]. Each of facial example reflects both expressions of different types such as happiness, sadness, and anger and designer's insight to be a good guideline for animation. With the advantages of this approach that preserve the characteristic features of examples and reflecting a designer's intention accurately, it becomes a popular method for various shape modeling and animation. However, it is rarely used in combination with performance-driven facial animation. Our contribution is to present a realtime example-based scheme for facial motion synthesis in conjunction with facial motion capture.

Given the displacements of feature points, our problem is to find the best blend of examples at each frame to resemble the facial motion of a performer. Provided with the source examples and corresponding target examples, in the pre-processing, all the target examples are parameterized by using the corresponding source examples to apply multidimensional scattered data interpolation. We provide a simple, elegant parameterization scheme for effective expression blending. Provided with the parameterized target examples, the next step is for computing the contribution of every target example to the new expression using cardinal basis functions. The final step is to blend the corresponding target examples in accordance with their contributions to synthesize the resulting expression.

4.1 Parameterization

We parameterize the target examples based on the displacements between the source examples. In the capture step, the displacements of feature points are extracted from a face performer. Concatenating these displacements, the displacement vector of each source example is formed to parameterize the corresponding target example. Most individual parameter components tend to be correlated to each other. Thus, based on PCA (principal component analysis), the dimensionality of the parameter space can be reduced by removing less significant basis vectors of the resulting eigenspace.

The displacement vector $\mathbf{v}_i$ of a source example $\mathbf{S}_i$ from the source base model $\mathbf{S}_B$ is defined as follows:

$$\mathbf{v}_i = \mathbf{s}_i - \mathbf{s}_B,\ 1 \leq i \leq M, \tag{5}$$

where $\mathbf{s}_B$ and $\mathbf{s}_i$ are vectors obtained by concatenating, in a fixed order, the 3D coordinates of feature points on $\mathbf{S}_B$ and those on $\mathbf{S}_i$, respectively, and M is the number of source examples. As shown in Figure 4, $\mathbf{v}_i$ places each target example $\mathbf{T}_i$ in the N-dimensional parameter space, where N is the number of components, that is, three times the number of feature points.

Since the dimensionality N of the parameter space is rather high compared to the number M of examples, we employ PCA to reduce it. Given M displacement vectors of dimension N, we first generate their component covariance matrix, which is an $N \times N$ square matrix, to compute the eigenvectors of the matrix and the corresponding eigenvalues. These eigenvectors are called the *principal components* representing the principal axes that characterize the distribution of displacement vectors. The dimensionality of the parameter space can be reduced by removing less significant eigenvectors, which have small eigenvalues. In our experiments, we use an empirical threshold

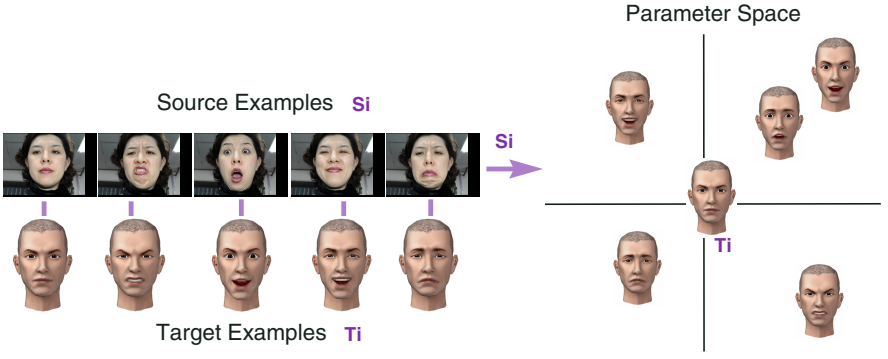

Fig. 4. The displacement vector of each source key-model $\mathbf{S}_i$ is used for parameterizing the corresponding target key-model $\mathbf{T}_i$

value of 0.00001 to remove those eigenvectors. The removal of such eigenvectors may cause some characteristics of the examples not to be parameterized. With our choice of the threshold, we have observed that the effect is negligible. In experiments, the dimensionality of the parameter space can be reduced from 60 to 18 without any difficulty.

Let $\mathbf{e}_i, 1 \leq i \leq N$ be the eigenvector corresponding to the ith largest eigenvalue. Suppose that we choose $\bar{N}$ eigenvectors as the coordinate axes of the parameter space, where $\bar{N} < N$. To transform an original N-dimensional displacement vector into an $\bar{N}$-dimensional parameter vector, an $\bar{N} \times N$ matrix $\mathbf{F}$ called the *feature matrix* is constructed:

$$\mathbf{F} = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3 \ldots \mathbf{e}_{\bar{N}}]^\top, \tag{6}$$

Using the feature matrix $\mathbf{F}$, the parameter vector $\mathbf{p}_i$ corresponding to the displacement vector v_i of a target example $\mathbf{T}_i$ is derived as follows:

$$\mathbf{p}_i = \mathbf{F}\mathbf{v}_i,\ 1 \leq i \leq M, \tag{7}$$

which reduces the dimensionality of the parameter space from N to $\bar{N}$. This is equivalent to projecting each displacement vector $\mathbf{v}_i$ onto the eigenspace spanned by the $\bar{N}$ selected eigenvectors. We later use this feature matrix $\mathbf{F}$ to compute the parameter vector from a given displacement vector.

5 Facial Animation

With the target 3D face examples thus parameterized, our problem is how to blend the examples so as to resemble the input expression extracted from a face performer. Our problem is essentially one of scattered data interpolation, as we have very sparse target examples in a relatively high dimensional parameter space. To solve the problem, we predefines an weight function for each target example based on cardinal basis functions [21], which consist of linear and radial basis functions. The global shape of weight function is first approximated by linear basis functions, and then adjusted locally by radial basis functions to exactly interpolate the corresponding example.

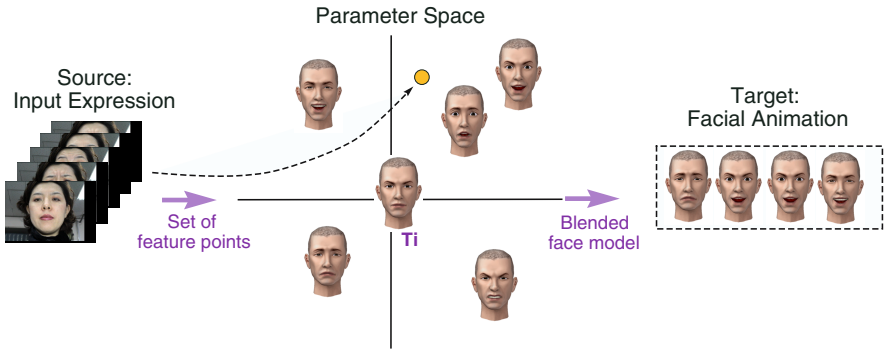

Fig. 5. Generating a new face model by blending target key-models

5.1 Weight Function

The weight function $w_i(\cdot)$ of each target example $\mathbf{T}_i, 1 \leq i \leq M$ at a parameter vector $\mathbf{p}$ is defined as follows:

$$w_i(\mathbf{p}) = \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}) + \sum_{j=1}^{M} r_{ji} R_j(\mathbf{p}). \tag{8}$$

where $A_l(\mathbf{p})$ and a_{il} are the linear basis functions and their linear coefficients, respectively. $R_j(\mathbf{p})$ and r_{ij} are the radial basis functions and their radial coefficients. Let $\mathbf{p}_i, 1 \leq i \leq M$ be the parameter vector of a target example $\mathbf{T}_i$. To interpolate the target examples exactly, the weight of a target example $\mathbf{T}_i$ should be one at $\mathbf{p}_i$ and zero at $\mathbf{p}_j, i \neq j$, that is, $w_i(\mathbf{p}_i) = 1$ for $i = j$ and $w_i(\mathbf{p}_j) = 0$ for $i \neq j$.

5.2 Linear Approximation

Our scheme first approximates the global shape of weight function by finding the hyperplane through the parameter space that best fists each example. Formally, in Equation (8), we would like to solve for the linear coefficients a_{il} to fix the first term. By ignoring the second term, we obtain the following Equation 11:

$$w_i(\mathbf{p}) = \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}). \tag{9}$$

The linear bases are simply $A_l(\mathbf{p}) = \mathbf{p}^l, 1 \leq l \leq \bar{N}$, where $\mathbf{p}^l$ is the lth component of $\mathbf{p}$, and $A_0(\mathbf{p}) = 1$. Using the parameter vector $\mathbf{p}_i$ of each target example and its weight value $w_i(\mathbf{p}_i)$, we employ a least squares method to evaluate the unknown linear coefficients a_{il} of the linear bases.

5.3 Radial Basis Function

Given the linear approximation, there still remain residuals between the examples and the approximated hyper-planes. To correct for the residuals, we associate radial basis

functions with each example. The radial basis function is a function of the distance between one point and another point indicating the location of example in the parameter space. The radial basis function is a bell shaped curve centered at the example point. Thus, the radial basis function is used to limit the influence of each example to a local region of the parameter space, that is, allows for local refinement of the weight function.

Mathematically, to fix the second term of Equation (8), we compute the residuals for the target examples:

$$w_i'(\mathbf{p}) = w_i(\mathbf{p}) - \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}) \text{ for all } i. \tag{10}$$

The radial basis function $R_j(\mathbf{p})$ is a function of the Euclidean distance between $\mathbf{p}$ and $\mathbf{p}_j$ in the parameter space:

$$R_j(\mathbf{p}) = B\left(\frac{\| \mathbf{p} - \mathbf{p}_j \|}{\alpha}\right) \text{ for } 1 \leq j \leq M, \tag{11}$$

where $B(\cdot)$ is the cubic B-spline function, and α is the dilation factor, which is the separation to the nearest other example in the parameter space. The radial coefficients r_{ij} are obtained by solving the matrix equation,

$$\mathbf{rR} = \mathbf{w}', \tag{12}$$

where $\mathbf{r}$ is an $M \times M$ matrix of the unknown radial coefficients r_{ij}, and $\mathbf{R}$ and $\mathbf{w}'$ are the matrices of the same size defined by the radial bases and the residuals, respectively, such that $\mathbf{R}_{ij} = R_i(\mathbf{p}_j)$ and $\mathbf{w'}_{ij} = w_i'(\mathbf{p}_j)$.

5.4 Runtime Expression Synthesis

Given the input expression captured from a face performer, the application computes a novel output model at runtime by blending the target examples as illustrated in Figure 5. The resulting expression are produced so as to resemble the input expression. Our scheme consists of three steps. The first, as an input, the displacement vector of feature points on a face performer is given. The next step is to derive the parameter vector from the displacement vector by using the Equation 13. Finally, the predefined weight functions are computed at this parameter vector to produce the weight values, and the output model is generated by blending the target examples with respect to those weight values.

First, we form the N-dimensional displacement vector $\mathbf{d}_{in}$ by concatenating, in a fixed order, the 3D displacements of feature points captured from a face performer. N is three times the number of feature points for X, Y, and Z coordinates.

Next, given this N-dimensional displacement vector $\mathbf{d}_{in}$, we then obtain the corresponding $\bar{N}$-dimensional parameter vector $\mathbf{p}_{in}$ as follows:

$$\mathbf{p}_{in} = \mathbf{F}\mathbf{d}_{in}, \tag{13}$$

where $\mathbf{F}$ is the feature matrix defined in Equation (6).

Finally, using the predefined weight functions for the target examples $\mathbf{T}_i$ as given in Equation (8), we estimate the weight values $w_i(\mathbf{p}_{in})$ of all target examples $\mathbf{T}_i, 1 \leq i \leq M$ at the parameter $\mathbf{p}_{in}$ to generate the output face model $\mathbf{T}_{new}(\mathbf{p}_{in})$:

$$\mathbf{T}_{new}(\mathbf{p}_{in}) = \mathbf{T}_B + \sum_{i=1}^{M} w_i(\mathbf{p}_{in})(\mathbf{T}_i - \mathbf{T}_B), \tag{14}$$

where $\mathbf{T}_B$ is the target base model corresponding to the source base model $\mathbf{S}_B$ with the neutral expression.

6 Experimental Results

To evaluate effectiveness and performance of the proposed method, we performed experiments on a PC with Pentium IV 1.2 GHz CPU and 512 MB memory. Face images were captured with a single digital camera and sent to the PC through a video capture board at 30 frames per second. To illuminate the puppeteer's face, we used two desktop lamps each of which has a single 13W bulb. As shown in Figure 6, neither any markers were attached to performer's face nor any head-mounted camera was employed. The head was allowed to move and rotate during facial expression capture on condition that all facial features were visible.

Figures 6 (a), (d) and (g) show the captured face images of nine puppeteers. Face images after color space transformation are given in Figures 6 (b), (e), and (h). Here,

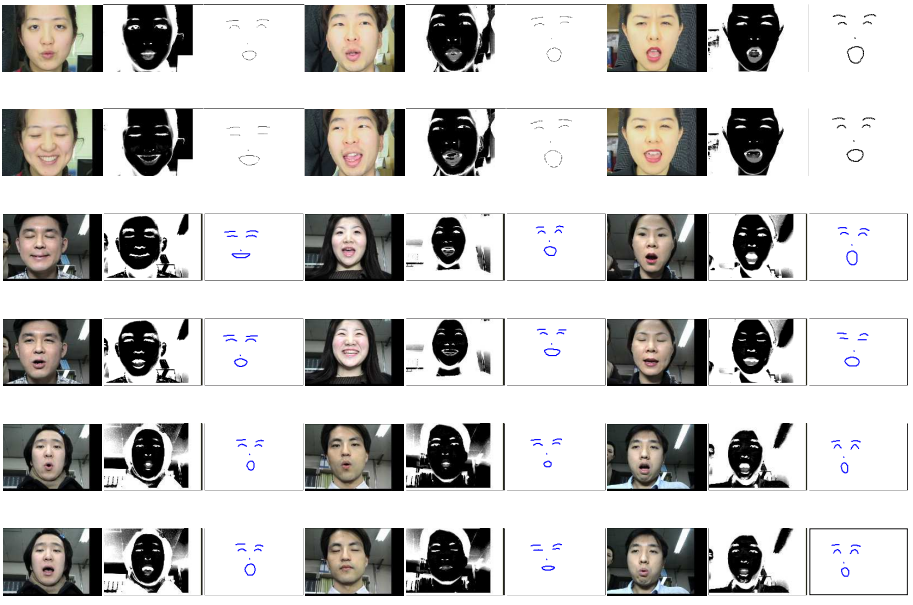

Fig. 6. Original images, color-transformed images, and extracted curves

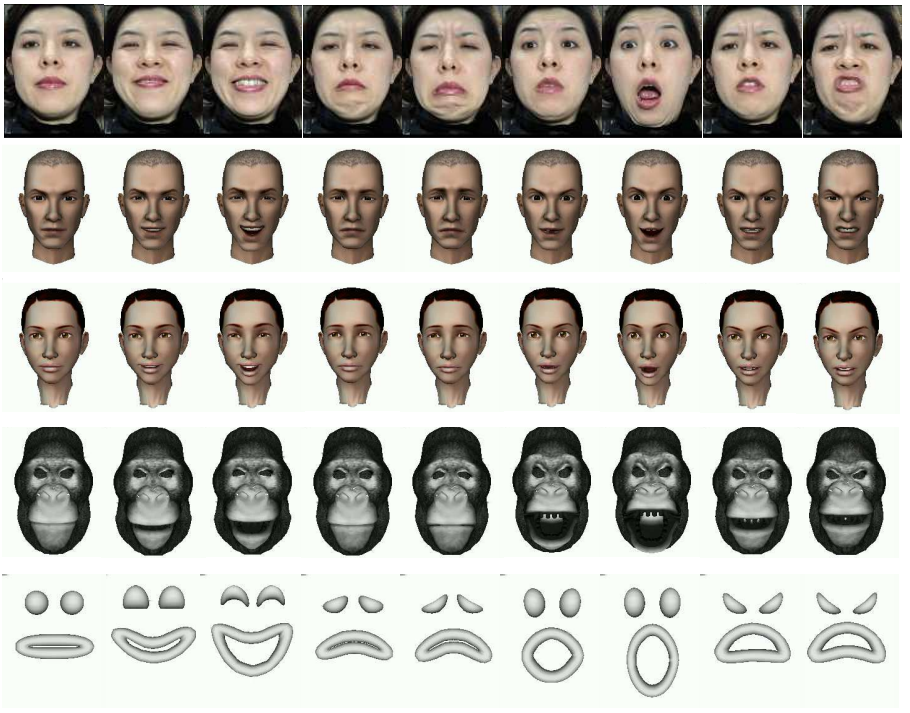

Fig. 7. Facial Animation

for each of the puppeteers, we could automatically obtain the weights of the color components in the color transformation function by using the trained neural network. From those images, we can observe that the intensity values of pixels in the region of the skin are so different from those of the facial features such as eyelashes, eyebrows, nostrils, and lips that the facial features are clearly distinguishable from the skin. Indeed, we were able to extract the facial features robustly from the transformed images. Figures (c), (f), and (i) exhibits the feature curves extracted from the images.Our method for facial expression capture can process more than 100 frames per second to exhibit a sufficient efficiency for real-time on-line performance-driven animation.

To demonstrate the final facial animation, we create several examples for various 3D face models. Figure 5.4 illustrates the facial animation as a result of deriving a 3D face model from the feature curve, extracted from a face performer. The first column of Figure 6 shows the original video of the face performer. The video comprises a total sequence of 800 frames that are recorded at 30 frames per second. With thirteen target key-models, we made the facial animation for each 3D face model, shown in the next four rows. Four different styles(Man, Woman, Monkey, and Toy) of 3D face models were used to show the usefulness of our example-based approach. Each result of facial animation keeps the personalities of each face model and reflects the designer's original intention. We can observe that our approach works well even though the shape of the performer's face and 3D face model largely differ.

7 Conclusion

In this paper, we have proposed online expression mapping method for performance-driven facial animation, reflecting the animator's creativity and imagination for a face model, by using a combination of facial expression capture and example-based animation. Our method is useful to overcome the characteristic differences between the performer's face and the target face model. Our solution consists of three major steps: facial motion capture, expression mapping, and facial animation. Our approach extracts facial expression from a performer in real time without employing a head-mounted camera or markers. With facial example-based approach, we adapt the captured motion of a performer to a specific face model, even if it is a anthropomorphized animal, in an on-line manner. Moreover, we achieve real-time facial animation by blending facial examples with their wire curves. As shown in the experimental results, our approach has achieved very convincing and lifelike facial animation following the expression of face performer, while reflecting an animator's intention, with great efficiency.

One limitation of our method require animators to prepare a set of facial examples. At the same time, it can be an advantage that it allows for human control of animation results so that the characteristics of examples are fully reflected. However, even for skilled artists, it is time consuming work to create a number of facial examples with extreme expressions. Automatic construction of facial examples from the captured expression or animator's sketch will be a good future research topic on facial animation. For more realistic facial animation, we would like to extend our scheme to incorporate subtle movements in addition to verbal and emotional expressions, such as eyeballs rolling and tongue movement.

References

1. Basu, S., Oliver, N., Pentland, A.: 3D modeling and tracking of human lip motions. In: Proceedings of ICCV 98 (1998)
2. Black, M.J., Yacoob, Y.: Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motions. In: International Conference on Computer Vision 95, pp. 374–381 (1995)
3. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIGGRAPH 1999 Conference Proceedings, pp. 187–194 (1999)
4. Chuang, E., Bregler, C.: Performance driven facial animation using blendshape interpolation. Stanford University Computer Science Technical Report, CS-TR-2002-02 (2002)
5. Chuang, E., Bregler, C.: Mood swings: Expressive speech animation. ACM Transactions on Graphics 24(2), 331–347 (2005)
6. DeCarlo, D., Metaxas, D.: Optical flow constraints on deformable models with applications to face tracking. International Journal of Computer Vision 38(2), 99–127 (2000)
7. Essa, I., Pentland, A.: Facial expression recognition usin a dynamic model and motion energy. In: Proceedings of ICCV 95, pp. 360–367 (1995)
8. Goto, T., Kshirsagar, S., Thalmann, N.M.: Real time facial feature tracking and speech acquisition for cloned head. IEEE Signal Processing Magazine 18(3), 17–25 (2001)
9. Guenter, B., Grimm, C., Wood, D., Malvar, H., Pighin, F.: Making faces. In: SIGGRAPH 98 Conference Proceedings, pp. 55–67 (1998)

10. Joshi, P., Tien, W.C., Desbrun, M., Pighin, F.: Learning controls for blend shape based realistic facial animation. In: Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
11. Kahler, K., Haber, J., Seidel, H.-P.: Reanimating the dead: Reconstruction of expressive faces from skull data. In: SIGGRAPH 2003 (2003)
12. Kalra, P., Mangili, A., Thalmann, N.M., Thalmann, D.: Simulation of facial muscle actions based on rational free form deformations. In: Eurographics' 92, vol. 58, pp. 59–69 (1992)
13. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331 (1987)
14. Lee, Y., Terzopoulos, D., Waters, K.: Realistic modeling for facial animation. In: SIGGRAPH' 95 Conference Proceedings, pp. 55–62 (1995)
15. Marschner, S.R., Guenter, B., Raghupathy, S.: Modeling and rendering for realistic facial animation. In: EUROGRAPHICS Rendering Workshop 2000, pp. 98–110 (2000)
16. Oliver, N., Pentland, A., Berard, F.: Lafter: Lips and face tracking. In: Computer Vision and Pattern Recognition '97 (1997)
17. Pighin, F., Szeliski, R., Salesin, D.: Resynthesizing facial animation through 3d model-based tracking. In: International Conference on Computer Vision, pp. 143–150 (1999)
18. Pratt, W.K.: Digital Image Processing, 2nd edn. Wiley Interscience, Chichester (1991)
19. Pyun, H., Kim, Y., Chae, W., Kang, H.W., Shin, S.Y.: An example-based approach for facial expression cloning. In: Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
20. Singh, K., Fiume, E.: Wires: A Geometric Deformation Technique. In: SIGGRAPH' 98 Conference Proceedings, pp. 299–308 (1998)
21. Sloan, P.-P., Rose, C.F., Cohen, M.F.: Shape by example. In: Proceedings of 2001 Symposium on Interactive 3D Graphics, pp. 135–144 (2001)
22. Terzopoulos, D., Waters, K.: Analysis and synthesis of facial image sequences using physical and anatomical models. IEEE Transactions of Pattern Analysis and Machine Intelligence 15(6), 569–579 (1993)
23. Thalmann, N.M., Pandzic, I., Kalra, P.: Interactive facial animation and communication. In: Tutorial of Computer Graphics International '96, pp. 117–130 (1996)
24. Williams, L.: Performance-driven facial animation. In: Proceedings of ACM SIGGRAPH Conference, pp. 235–242. ACM Press, New York (1990)
25. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: High resolution capture for modeling and animation. ACM Transactions on Graphics 23(3), 548–558 (2004)
26. Zhang, Q., Liu, Z., Guo, B., Shum, H.: Geometry-driven photorealistic facial expression synthesis. In: Eurographics/SIGGRAPH Symposium on Computer Animation (2003)