# VISUAL ESTIMATION AND COMPRESSION OF FACIAL MOTION PARAMETERS ¾ ELEMENTS OF A 3D MODEL-BASED VIDEO CODING SYSTEM

*Hai Tao*

Department of Computer Engineering
University of California, Santa Cruz, CA 95063

*Thomas S. Huang*

Image Processing and Formation Laboratory, Beckman Institute
University of Illinois at Urbana-Champaign, Urbana, IL 61801

## Abstract

The MPEG4 standard supports the transmission and composition of facial animation with natural video by including a facial animation parameter (FAP) set that is defined based on the study of minimal facial actions and is closely related to muscle actions. The FAP set enables model-based representation of natural or synthetic talking head sequences and allows intelligible visual reproduction of facial expressions, emotions, and speech pronunciations at the receiver. This paper describes two key components we have developed for building a model-based video coding system: (1) a method for estimating FAP parameters based on our previously proposed piecewise Bézier volume deformation model (PBVD), and (2) various methods for encoding FAP parameters. PBVD is a linear deformation model suitable for both the synthesis and the analysis of facial images. Each FAP parameter is a basis function in this model. Experimental results on PBVD-based animation, model-based tracking, and spatial-temporal compression of FAP parameters are demonstrated in this paper

## 1 INTRODUCTION

Many applications in human-computer interface, 3D games, model-based video coding, talking agent, and distance learning demand communication of talking head videos. There are at least two possible solutions to this problem: transmitting the image pixel directly or transmitting the geometric model, the deformation method, and some deformation parameters to synthesize the images on the other end of the transmission. The latter approach is called the model-based video coding method [1,2,8,9]. As shown in Figure 1, a model-based facial image communication system consists of three main components: (a) analysis of nonrigid facial motions, (b) compression and transmission of the face geometry model and the motion parameters, and (c) synthesis of facial images. The core of a model-based coding system is the model of the object under consideration. Two key elements comprise of a face model: the geometric face model and the face deformation model. They are described in MPEG-4 SNHC by the facial definition parameters (FDP) and the facial animation parameters (FAP) respectively.

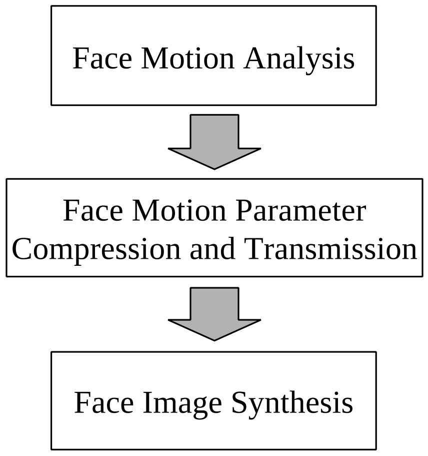


Figure 1. A model-based video coding system.

The geometric model defines the shape and the texture of a face. It is usually in the form of a 3D polygonal mesh. We have developed system to obtain 3D mesh models of faces from 3D CyberWare scanner data (Figure 2). The deformation model, on the other hand, describes how the face changes its shape and is used to generate various dynamic effects for intelligible reproduction of facial expressions. Four categories of facial deformation models have been proposed. They are parameterized models [3], physical muscle models [4], free-form deformation models [10], and performance-driven animation models [11]. In the motion analysis phase, these models are applied as constraints that regulate the facial movements. We have proposed a linear free-form facial deformation model called piecewise Bézier volume deformation (PBVD) in [5]. This method is suitable for both realistic facial animation and robust facial motion analysis. The differences between this approach and Kalra's method [10] are twofold. By using nonparallel volumes, irregular 3D manifolds are formed. As a result, fewer deformation volumes are needed and the number of control points is reduced. This is a desired property for tracking algorithms. Further, based on facial feature points, this model is mesh independent and can be easily adopted to articulate any face model. In addition, the MPEG-4 facial animation parameters, which are defined based on the study of minimal facial actions and that are closely related to muscle movements [3,4], can also easily built upon the PBVD model.

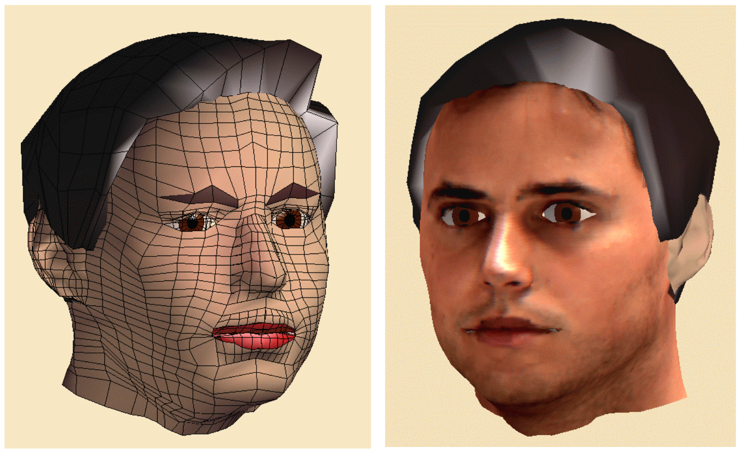


Figure 2. A face mesh model constructed using the CyberWare scanner data. Left: the mesh model. Right: the texture-mapped model.

After being estimated, the facial motion parameters need to be transmitted over various communication channels to drive the facial animation program at the remote locations. How to transmit these parameters efficiently is an interesting research problem. We have developed several compression algorithms for encoding the estimated MPEG4 FAP parameters. These algorithms exploit the spatial and temporal redundancy among FAP parameters. Combinations of these techniques dramatically reduce the amount of data needs to be transmitted. The three algorithms we have developed are: (1) face animation parameter interpolation scheme, (2) spatial FAP compression method based on principal component analysis, and (3) temporal transformation coding of FAP. The efficacy of these methods is demonstrated by both subjective and objective results.

## 2 FACE TRACKING BASED ON PBVD

### 2.1 Piecewise Bézier volume deformation model

In [5], we have introduced a new piecewise Bézier volume deformation (PBVD) model. This model describes the mapping function between some face motion control parameters and the 3D displacements of the face mesh nodes. The PBVD model consists of a collection of 3D Bézier volumes [7,10] in which the face mesh model is completely embedded (Figure 3a). As described in [5], the linear relationship between the displacements of the face mesh nodes $\mathbf{V}$ and the magnitudes of the various control parameters $P = [p_{0,} p_{1,\ldots,} p_m]^t$ is

$$\mathbf{V} = LP \tag{1}$$

where the matrix $L$ describes the basis functions composed of some linear combinations of Bernstein polynomials. Each column of $L$ represents an expression, a viseme (*vis*ual phon*eme*), or a FAP action unit that can be derived through manipulating the control points through an interactive tool. In Figure 3, the real control mesh and the rendered expression smile are illustrated. We have created 23 visemes to implement a talking head system. Six of them are shown in Figure 4. Basis functions for various expressions (see Figure 5) as well as some FAP-like action units have also been implemented. Once visemes and expressions are created, animation sequences can be generated by assigning appropriate values to the magnitudes of these visemes and expressions at each time instance. Figure 6 shows some frames from a synthetic visual speech sequence. A full deformation model that includes the rigid head motion is estimated in our tracking algorithm. The motion model is described as

$$R(\mathbf{V}_0 + LP) + T \tag{2}$$

where $\mathbf{V}_0$ is the neutral facial mesh, $R$ is the rotation decided by the three rotation angles $\Omega$, and $T$ is the 3D translation.

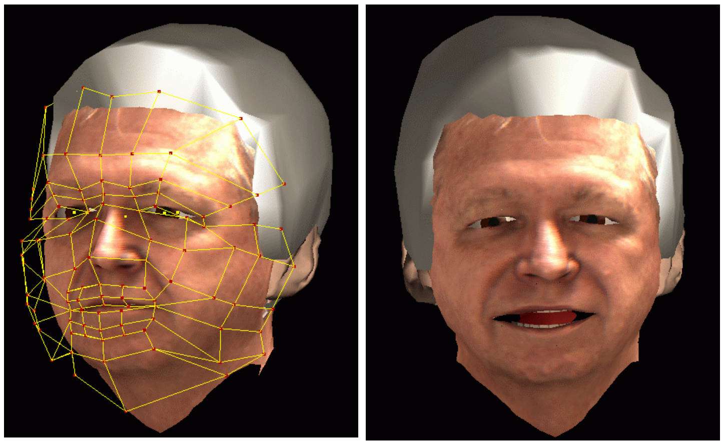

(a) (b)

Figure 3. (a) The PBVD volumes (b) the expression *smile*.

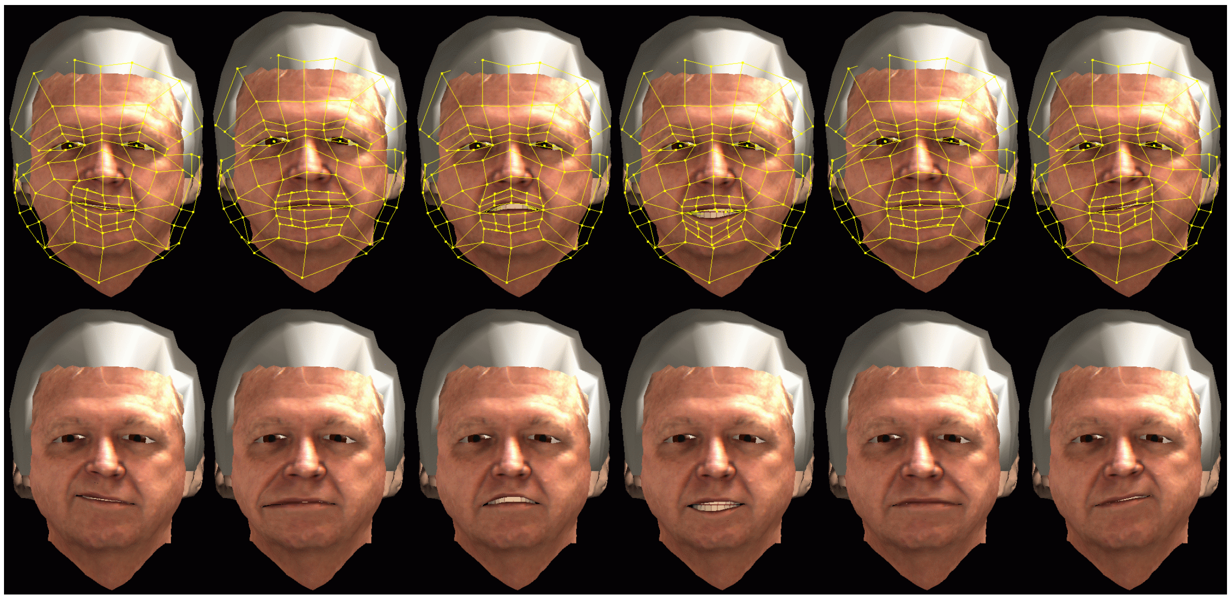


Figure 4. Action units around the mouth region. Top: the control nodes. Bottom: the action units.

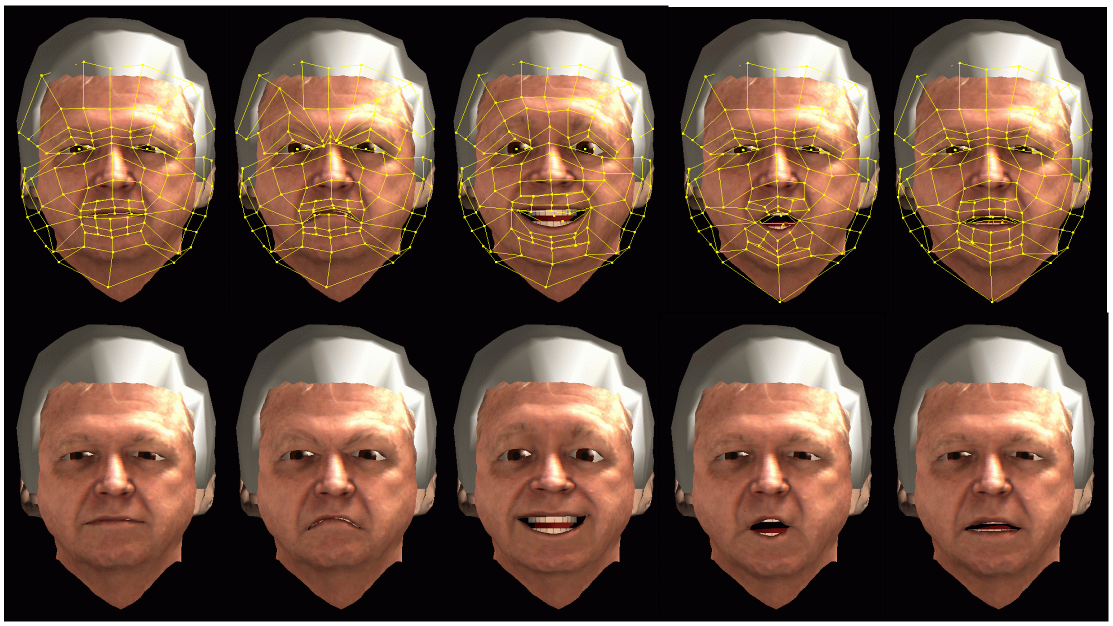


Figure 5. Expressions and visemes created using the PBVD model. The expressions and visemes (bottom row) and their corresponding control meshes (top row). The facial movements are, from left to right, *neutral*, *anger*, *smile*, *vowel_or*, and *vowel_met*.

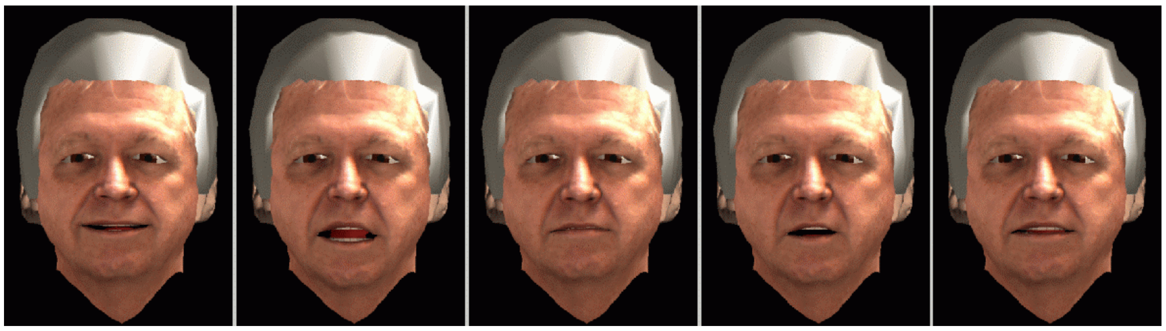


Figure 6. An animation sequence with *smile* and speech *I am George McConkie.*

## 2.2 PBVD model-based tracking algorithm

### 2.2.1 Video analysis of the facial movements

Several algorithms for extracting face motion information from video sequences have been proposed [1,2,8,9]. Most of these methods are designed to detect action-unit level animation parameters. The assumption is that the basic deformation model is already given and will not change. In this section, we propose a tracking algorithm in the same flavor but using the PBVD model. The algorithm adopts a coarse-to-fine framework to integrate the low-level motion field information with the high-level deformation constraints. Since the PBVD model is linear, an efficient optimization process using lease squares estimator is formulated to incrementally track the head poses and the facial movements. The derived motion parameters can be used for facial animation, expression recognition, and bimodal speech recognition.

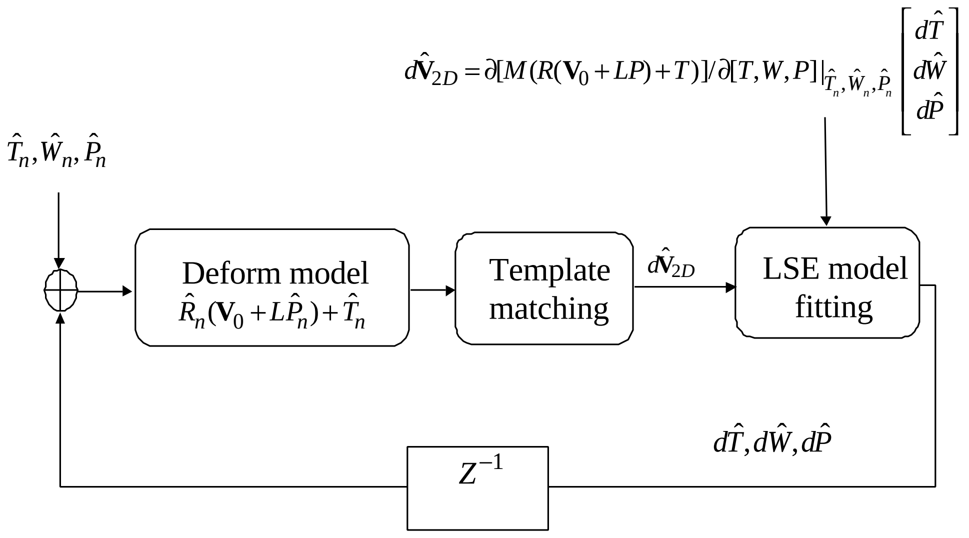

$d\hat{\mathbf{V}}_{2D} = \partial[M(R(\mathbf{V}_0 + LP) + T)]/\partial[T, W, P]|_{\hat{T}_n,\hat{W}_n,\hat{P}_n} \begin{bmatrix} d\hat{T} \\ d\hat{W} \\ d\hat{P} \end{bmatrix}$
$\hat{T}_n, \hat{W}_n, \hat{P}_n$
Deform model
$\hat{R}_n(\mathbf{V}_0 + L\hat{P}_n) + \hat{T}_n$
Template matching
$d\hat{\mathbf{V}}_{2D}$
LSE model fitting
$d\hat{T}, d\hat{W}, d\hat{P}$
$Z^{-1}$

Figure 7. Block diagram of the model-based PBVD tracking system.

### 2.2.2 Model-based tracking using the PBVD model

The changes of the motion parameters between two consecutive video frames are computed based on the motion field. The algorithm is shown in Figure 7. We assume that the camera is stationary. At the initialization stage, the face needs to be approximately frontal view so that the generic 3D model can be fitted. The inputs to the fitting algorithms are the positions of facial feature points, which are manually picked. All motion parameters are set to zeroes (i.e., $(\hat{T}_0, \hat{\Omega}_0, \hat{P}_0) = 0$), which means a neutral face is assumed. The intrinsic camera parameters are known in our implementation. Otherwise, a small segment of the video sequence should be used to estimate these parameter using photogrammetry techniques.

From the video frames $n$ and $n+1$, the 2D motion vectors of many mesh nodal points are estimated using the template matching method. In our implementation, the template for each node consists of $11\times 11$ pixels and the searching region is $17\times 17$ pixels. To deal with the drifting problem, both the templates from the previous frame and the templates from the initial frame are used: the even nodes of a patch are tracked using the templates from the previous frame and the odd nodes are tracked using those of the initial frame. Our experiments showed that this approach is very effective.

From the resulted motion vectors, 3D rigid motions and nonrigid motions (intensities of expressions/visemes or action units) are computed simultaneously using a least squares estimator. Since the PBVD model is linear, only the perspective projection and the rotation introduce non-linearity. This property makes the algorithm simpler and more robust. The 2D interframe motion for each node is

$$\begin{aligned} d\hat{\mathbf{V}}_{2D} &\approx \frac{\partial[M(R(\mathbf{V}_0+LP)+T)]}{\partial[T,\Omega,P]}\bigg|_{\hat{T}_{n+1},\hat{\Omega}_{n+1},\hat{P}_{n+1}} \begin{bmatrix} d\hat{T} \\ d\hat{\Omega} \\ d\hat{P} \end{bmatrix} \\ &= M\left( \begin{bmatrix} \begin{bmatrix} 1 & 0 & -x/z \\ 0 & 1 & -y/z \end{bmatrix} \begin{bmatrix} G_0 - x/zG_2 \\ G_1 - y/zG_2 \end{bmatrix} \begin{bmatrix} [RL]_0 - x/z[RL]_2 \\ [RL]_1 - y/z[RL]_2 \end{bmatrix} \end{bmatrix} \begin{bmatrix} d\hat{T} \\ d\hat{\Omega} \\ d\hat{P} \end{bmatrix} \right) \end{aligned} \tag{3}$$

where $d\hat{\mathbf{V}}_{2D}$ is the 2D interframe motion, and

$$G = \begin{bmatrix} 0 & z_1 & -y_1 \\ -z_1 & 0 & x_1 \\ y_1 & -x_1 & 0 \end{bmatrix}. \tag{4}$$

The projection matrix $M$ is

$$M = \begin{bmatrix} fs/z & 0 & 0 \\ 0 & fs/z & 0 \end{bmatrix} \tag{5}$$

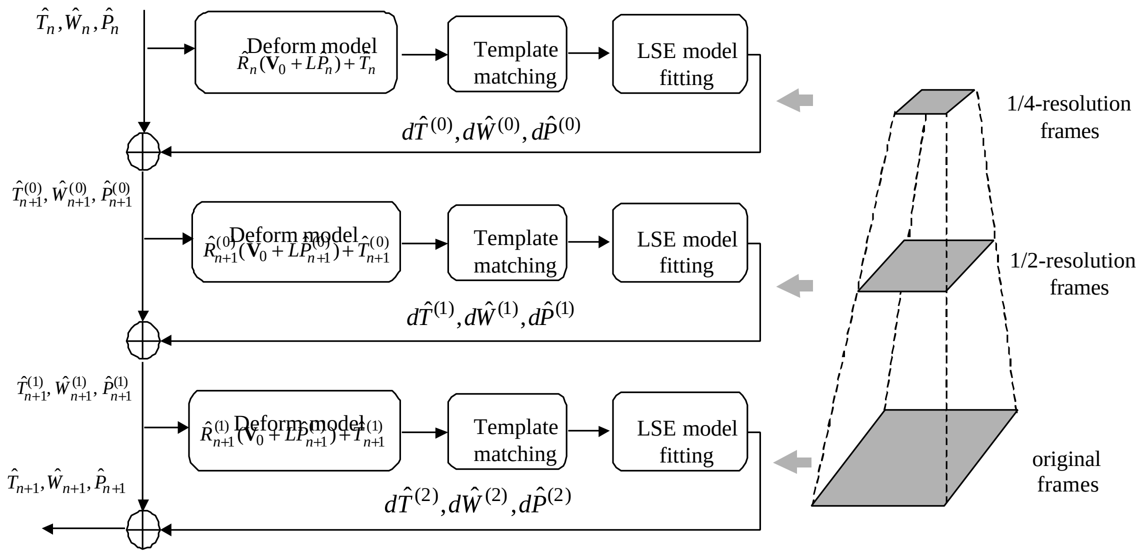
where $f$ is the focal length of the camera, $s$ is the scale factor, and $z$ is the depth of the mesh node. The vector $(x, y, z)$ represents the 3D mesh nodal position after both rigid and nonrigid deformation, or $R(\mathbf{V}_0 + LP) + T$. The vector $(x_1, y_1, z_1)$ represents the 3D mesh nodal position after only nonrigid deformation, but without translation and rotation (i.e., $\mathbf{V}_0 + LP$). Matrix $G_i$ and $[RL]_i$ denote the $i$ th row of the matrix $G$ and the matrix $RL$, respectively.

An overdetermined system is formed because many 2D inter-frame motion vectors are calculated. As the result, changes of the motion parameters $(d\hat{T}, d\hat{\Omega}, d\hat{P})$ can be estimated using the least squares estimator. By adding these changes to the previously estimated motion parameters $(\hat{T}_n, \hat{\Omega}_n, \hat{P}_n)$, new motion parameters $(\hat{T}_{n+1}, \hat{\Omega}_{n+1}, \hat{P}_{n+1})$ are derived.

### 2.2.3 Coarse-to-fine framework

Two problems with the above algorithm are the computationally expensive template matching and the noisy motion estimation. The first problem is obvious because the computational complexity for each motion vector is approximately $(17 \times 17) \times (11 \times 11) \times 3 = 104{,}907$ integer multiplication. The second problem is partially caused by the fact that in the above algorithm, the computation of the motion field is totally independent of the motion constraints, which makes it vulnerable to various noises. If the lower-level motion field measurements are very noisy, good estimation of motion parameters can never be achieved, even with the correct constraints.

A coarse-to-fine framework is proposed in this section to partially solve the above problems. The block diagram of this new algorithm is illustrated in Figure 8. An image pyramid is formed for each video frame. The algorithm proposed in the previous section is then applied to the consecutive frames sequentially from lowest resolution to the original resolution. For the diagram depicted in Figure 8, changes of motion parameters are computed in quarter-resolution images as $(d\hat{T}^{(0)}, d\hat{\Omega}^{(0)}, d\hat{P}^{(0)})$. By adding these changes to $(\hat{T}_n, \hat{\Omega}_n, \hat{P}_n)$, the estimated new motion parameters are derived as $(\hat{T}_{n+1}^{(0)}, \hat{\Omega}_{n+1}^{(0)}, \hat{P}_{n+1}^{(0)})$. Similarly, changes of motion parameters are computed in the half-resolution images as $(d\hat{T}^{(1)}, d\hat{\Omega}^{(1)}, d\hat{P}^{(1)})$ based on the previous motion parameter estimation $(\hat{T}_{n+1}^{(0)}, \hat{\Omega}_{n+1}^{(0)}, \hat{P}_{n+1}^{(0)})$. This process continues until the original resolution is reached.



$\hat{T}_n, \hat{W}_n, \hat{P}_n$
Deform model
$\hat{R}_n(\mathbf{V}_0 + L\hat{P}_n) + \hat{T}_n$
Template matching
LSE model fitting
$d\hat{T}^{(0)}, d\hat{W}^{(0)}, d\hat{P}^{(0)}$
$\hat{T}_{n+1}^{(0)}, \hat{W}_{n+1}^{(0)}, \hat{P}_{n+1}^{(0)}$
Deform model
$\hat{R}_{n+1}^{(0)}(\mathbf{V}_0 + L\hat{P}_{n+1}^{(0)}) + \hat{T}_{n+1}^{(0)}$
Template matching
LSE model fitting
$d\hat{T}^{(1)}, d\hat{W}^{(1)}, d\hat{P}^{(1)}$
$\hat{T}_{n+1}^{(1)}, \hat{W}_{n+1}^{(1)}, \hat{P}_{n+1}^{(1)}$
Deform model
$\hat{R}_{n+1}^{(1)}(\mathbf{V}_0 + L\hat{P}_{n+1}^{(1)}) + \hat{T}_{n+1}^{(1)}$
Template matching
LSE model fitting
$\hat{T}_{n+1}, \hat{W}_{n+1}, \hat{P}_{n+1}$
$d\hat{T}^{(2)}, d\hat{W}^{(2)}, d\hat{P}^{(2)}$
1/4-resolution frames
1/2-resolution frames
original frames

Figure 8. The coarse-to-fine PBVD tracking algorithm.

In this coarse-to-fine algorithm, motion vector computation can be achieved with smaller searching regions and smaller templates. In our implementation, for each motion vector, the number of multiplication is $[(5\times5)\times(7\times7)\times3]\times4 \;=14{,}700$, which is about seven times fewer than the model-based scheme. A more important property of this method is that, to certain extent, this coarse-to-fine framework integrates motion vector computation with high-level constraints. The computation of the motion parameter changes is based on the approximated motion parameters at low-resolution images. As the result, more robust tracking results are obtained.

### 2.2.4 Implementation and experimental results

The PBVD model has been implemented on a SGI ONYX machine with a VTX graphics engine. Real-time tracking at 10 frame/s has been achieved using the coarse-to-fine framework. It has also been used for bimodal speech recognition and bimodal emotion recognition. Explanation-based method has been implemented to improve the facial image synthesis.

In PBVD tracking algorithm, the choice of deformation units $\mathbf{D}_i$ depends on the application. In a bimodal speech recognition application, 6 action units are used to describe the motions around the mouth. The tracking result for each frame is twelve parameters including the rotation, the translation, and the intensities of these action units.

For the bimodal emotion recognition and the real-time tracking system, 12 action units are used. Users can design any set of deformation units for the tracking algorithm. These deformations can be either at expression level or at action unit level. Lip tracking results are shown in Figure 10. Figure 9 shows the results of the real-time tracker.

Facial animation sequences are generated from the detected motion parameters. Figure 11 shows the original video frame and the synthesized results. The synthesized face model uses the initial video frame as the texture. The texture-mapped model is then deformed according to the motion parameters.

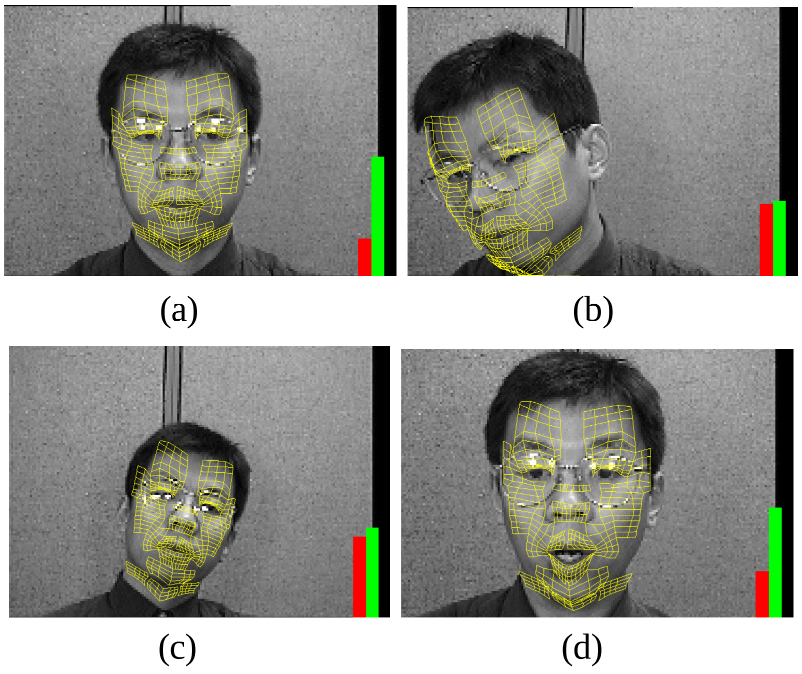


Figure 9. Snapshots of the real-time demonstration system. The ratio between the heights of the two color bars indicates the quality of the tracking result.

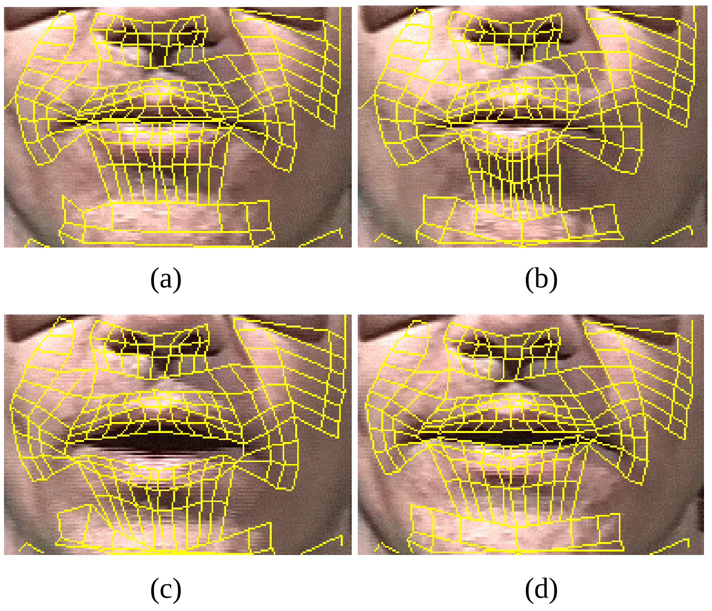


Figure 10. Lip tracking for bimodal speech recognition.

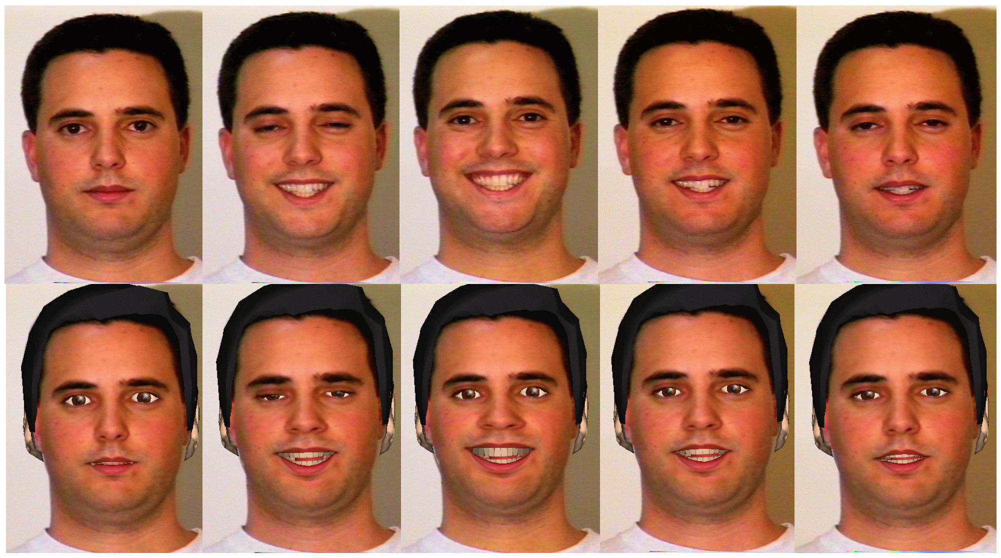


Figure 11. The original video frames (top) and the image synthesis results(bottom).

## 3 COMPRESSION OF THE FACIAL ANIMATION PARAMETERS

In this section, data compression methods for facial motion parameters will be described. The three algorithms we have developed are: (1) face animation parameter interpolation scheme, (2) spatial FAP compression method based on principal component analysis, and (3) temporal transformation coding of FAP. The efficacy of these methods is demonstrated by both subjective and objective results.

### 3.1 FAP Interpolation Table (FIT)

FAP interpolation scheme achieves data reduction by sending only a subset of the active FAPs. This subset is then employed to determine the values of other FAPs. It exploits the symmetry of human face or the articulation functions known *a priori*. For example, the top inner lip FAPs can be used to determine the top outer lip FAPs. FAP interpolation is a desirable tool to overcome channel bandwidth limitation. It is suitable for recovering missing FAPs caused by, for example, imperfect face feature extraction at the encoder or packet loss during data transmission.

In practice, it is also critical for the decoder to interpolate FAPs the same way as the encoder does. Otherwise, unpredictable results may be generated. A seemingly convenient solution is to predefine interpolation rules in a standard with which all FAP encoders and decoders comply. However, considering the *a priori* knowledge about the relations among FAPs are approximate and varies for individual persons, it is generally difficult to decide a set of fixed interpolation rules that fit all faces. Further, there are many possible ways of interpolating FAPs, both in terms of which FAPs need interpolation and how the interpolation is performed, that it is virtually impossible to exhaustively define all of them. A more feasible and efficient approach called FAP interpolation table (FIT) is proposed in [12]. The basic idea is to allow users to define interpolation methods and send this information at the setup stage of each FAP transmission session. FIT encodes both the interpolation syntax and the interpolation functions. An interpolation syntax indicates from which other FAPs a FAP can be interpolated. An interpolation function describes the exact numerical relations. The two major elements in FIT are the FAP interpolation graph (FIG), which describes the interpolation syntax, and the rational polynomials, which specify the interpolation functions. FIG is an efficient scheme that easily describes complicated relations between FAPs, while the multivariable rational polynomial is capable of representing both linear and nonlinear interpolations. The FAP interpolation table is also sent in the setup stage when FAP interpolation is required in the face animation session.

The FAP interpolation graph is a graph with directed links. Each node contains a set of FAPs. Each link from a parent node to a child node indicates that the FAPs in child node can be interpolated from parent node provided that all the FAPs in the parent node are available. An FAP may appear in several nodes, and a node may have multiple parents. For a node that has multiple parent nodes, the parent nodes are ordered as first parent node, second parent node, etc. During the interpolation process, if this child node needs to be interpolated, it is first interpolated from its first parent node if all required FAPs in that parent node are available. Otherwise, it is interpolated from its second parent node, and so on.

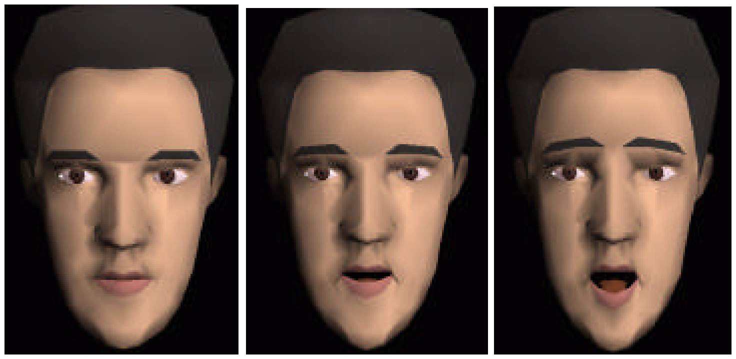


Figure 12: FAP interpolation results for the expression *Surprise*. Only the expression FAP is sent for each frame, and the 27 low-level FAPs are interpolated.

Each directed link in FIG represents a set of interpolation functions. Suppose $F_1$, $F_2$, $\ldots, F_n$ are the FAPs in a parent node and $f_1$, $f_2$, $\ldots$, $f_m$ are the FAPs in a child node. Then, there are $m$ interpolation functions denoted as

$$\begin{aligned} &f_1 = I_1(F_1, F_2, \ldots, F_n), \\ &f_2 = I_2(F_1, F_2, \ldots, F_n), \\ &\ldots \\ &f_m = I_m(F_1, F_2, \ldots, F_n). \end{aligned} \tag{6}$$

Each interpolation function $I_t()$ is in a rational polynomial form

$$I(F_1, F_2, \ldots, F_n) = \sum_{i=0}^{K-1}(c_i \prod_{j=1}^{n} F_j^{\,l_{ij}}) \Big/ \sum_{i=0}^{P-1}(b_i \prod_{j=1}^{n} F_j^{\,m_{ij}}), \tag{7}$$

where $K$ and $P$ are the numbers of polynomial products, $c_i$ and $b_i$ are the coefficient of the $i$th product, and $l_{ij}$ and $m_{ij}$ are the power of $F_j$ in the $i$th product. The encoder should send an interpolation function table which contains all $K$, $P$, $s_i$, $c_i$, $b_i$, $l_{ij}$, $m_{ij}$ to the decoder for each child FAP. Because rational polynomials form a complete functional space, any possible finite interpolation function can be represented in this form to any given precision.

Figure 12 illustrates the results of applying the FAP interpolation scheme to an MPEG4 test sequence. Only the magnitude of the expression parameter is transmitted and the rest of the FAPs are generated using the pre-transmitted interpolation method.

## 3.2 Compression of FAPs Using PCA

The aforementioned FIT approach encodes the fixed relationships among FAPs valid in all individual frames. The original FAP data is represented by its smaller subset. A more general tool for exploiting both deterministic and statistical correlation among FAPs is the principal component analysis method (PCA) [13], which converts the original FAP data to a new compact form. This method is motivated by the observation that different parts of a human face are articulated harmoniously and, though fixed relations may be absent or difficult to derive, statistically strong correlation does exist.

To apply PCA technique, major axes are computed as the eigenvectors of the covariance matrix computed from FAP vectors. Each FAP vector is formed by FAPs at a particular frame. The eigenvalues of the covariance matrix indicate the energy distribution. The major axes corresponding to significant eigenvalues form a new low-dimensional subspace. Compact representation is obtained by projecting the original FAP vector into this subspace. Re-projecting the compact representations back to the original FAP space produces good approximations of the original FAP vectors. This process is also called Karhunen Loeve transform (KLT) [14].

Suppose the original FAP vectors are $v_1$, $v_2, \ldots, v_n$, and each $v_i$ is a column vector contains $m$ FAPs in a particular frame. Then, the $m \times m$ covariance matrix is computed as:

$$C = \frac{1}{n-1} \sum_{i=1}^{n} (v_i - \bar{v})(v_i - \bar{v})^t , \tag{8}$$

where $\bar{v}$ is the mean of $v_i$. Since for most FAPs, the average positions are at neutral expression, or 0, the covariance matrix is:

$$C = \frac{1}{n-1}\sum_{i=1}^{n} v_i v_i^{\ t} . \qquad (9)$$

Since $C$ is a nonnegative definite matrix, all eigenvalues are nonnegative real values. We denote the eigenvalues in a scending order as $l_1$, $l_2$, … , $l_m$ and their corresponding igenvectors as $u_1$, $u_2$, … , $u_m$. Suppose that the first $k$ eigenvalues are significantly large, or that the percentage of energy $a = \sum_{i=1}^{k} l_i \Big/ \sum_{i=1}^{m} l_i$ exceeds a certain threshold; these $k$ eigenvectors then form a subspace that preserves most of the information in $v_i$. Each $v_i$ is projected into this new subspace by performing a linear transformation

$$q_i = \begin{bmatrix} u_1^t \\ u_2^t \\ \vdots \\ u_k^t \end{bmatrix} v_i . \qquad (10)$$

The derived $k$-dimensional vector $q_i$ is encoded and transmitted through the channel. To approximate $v_i$ from $q_i$, the following linear transformation is performed at the decoder side:

$$\hat{v}_i = [u_1 \quad u_2 \quad \dots \quad u_k]\, q_i . \qquad (11)$$

PCA reduces the dimension of FAP data dramatically. Although some new components of $q_i$ may possess larger data ranges and need more bits for coding, still significant bit savings are achieved. It should be noted that the eigen-vectors $u_j$, $j = 1,\dots,k$, for each FAP sequence also need to be sent in the setup stage to ensure that the decoder correctly recovers $\hat{v}_i$. For a low-bandwidth system with limited resources for downloading, a set of universal major axes $u_j$ defined so that both encoder and decoder include this KLT and no explicit setup is necessary for each sequence. This universal transform can be obtained by applying PCA to large amounts of training data with various facial motions.

## 3.3 Reduction of FAP Temporal Redundancy

In temporal sequence of each single FAP parameter or each PCA major component $q_i$, strong interframe correlation exists. Actually, the similarity of values between consecutive temporal frames appears in most animation parameters and can be explained as the result of the inertia forces in all mechanical systems. Compression techniques on temporal domain benefit from this characteristic to achieve bit savings. We describe two schemes in this section. They are the predictive coding (PC) method and the discrete cosine transform (DCT) method [15].

### 3.3.1 Predictive coding method (PC)

Instead of transmitting the parameters themselves, the differences between consecutive frames are encoded and transmitted. Because neighboring frames for each parameter contain similar values, the differences between the parameters tend to be small. Fewer bits are needed for representing these differences. For each FAP, its encoded value in the previous frame is used as the prediction for its current value. Because both the decoder and the encoder use the same prediction method, error accumulation is avoided. The prediction error (i.e., the difference between the current FAP value and its prediction) is then quantized and coded using an adaptive arithmetic coding algorithm. This process is also called interframe coding.

Intraframe coding, on the contrary, directly encodes the quantized FAP value. It is equivalent to setting the value in "Memory" to 0. A typical transmission session applies intraframe coding for the first frame or a frame that is not closely related to its previous frame.

A noteworthy problem in FAP predictive coding scheme is how to set appropriate quantization step for each FAP. The perceptual sensitivity of each FAP and their value range need to be considered simultaneously. For example, the jaw movements can be quantized more coarsely than the lip movements without affecting the perceptual quality of the resulted animation. This problem can also be addressed as:

for a given transmission bandwidth how to achieve the best visual animation results by adjusting the bit distributions on each parameters through setting its quantization step. Extensive empirical results are indispensable to derive a plausible solution.

Because there is no temporal latency for predictive coding, this scheme is suitable for real-time applications where long delay is a major concern. In many other situations, however, a small decoding latency is tolerable. An example of this type of applications is a multimedia mailing system with a front-end talking agent. The FAP sequences for this type of applications are usually stored in a disk, and compression efficiency has higher priority. Transformation coding method should be used for such applications.

### 3.3.2 FAP coding using DCT

Constrained by the physical properties of human faces, most motions in expressions or visemes are of frequency less than 10 Hz. Usually, visual speech motions are faster than emotional expressions. To guarantee satisfactory animation results, actual FAP data are often sampled at 30 Hz or 25 Hz. Consequently, strong correlation not only exists between consecutive frames, but also presents among multiple neighboring frames. To take advantage of this characteristic of FAP data, DCT compression method is employed. Similar to Fourier transform, DCT converts the original data into their frequency domain representations, namely, dc coefficients and ac coefficients. Since FAP data has relative low frequencies, most high-frequency ac coefficients are small and can be discarded. This dramatically reduces the amount of data need to be transmitted.

To perform DCT compression on each FAP parameter or PCA component, the temporal sequence is first decomposed into segments of 16 frames. A one-dimensional DCT is then applied to these individual segments. After DCT, the resulted 1 dc and 15 ac coefficients are quantized. Similar to the predictive coding scheme, for each FAP parameter, a particular quantization step is specified according to its perceptual sensitivity to error. Because dc coefficient is the mean value of the segment and is prone to error, its quantization step is three times smaller than that of ac coefficients.

For quantized dc coefficients, predictive coding method is applied between segments to take advantage of the correlation between them. In an intrasegment, the quantized dc coefficient value is directly stored. For intersegment, the quantized dc coefficient of the previous segment is used as a prediction and the prediction error is encoded using Huffman coding method.

For the nonzero ac coefficients in each segment, their positions and values need to be encoded. To encode their positions, for each ac coefficient, a run-length encoder is applied to record the number of leading zeros. A special symbol is defined to indicate the last nonzero ac coefficient in a segment. Since the segment length is 16, possible run-length values range from 0 to 14. Therefore, taking the "`end_of_segment`" symbol into account, the Huffman table of the run-length encoder contains 16 symbols. The values of nonzero ac coefficients are then encoded using a Huffman encoder.

As in a predictive FAP coding scheme, quantization steps need to be carefully assigned to each parameter. Since the property of DCT coefficients are different from that of the original data, different values need to be deduced. Again, empirical results are crucial for justifying these values. To further exploit the human perceptual properties, different quantization steps should be designed for ac coefficients of different frequencies. Subjective experiments need to be conducted on resulted animation. From careful examination, we proposed a set of DCT quantization steps, which is included in the MPEG-4 visual committee draft [16].

## 3.4 Reduction of FAP Spatial and Temporal Redundancy

Compression methods in spatial domain (among FAPs) are orthogonal to methods in temporal domain. The first approach benefits from the correlation among FAPs in a single frame, whereas the latter one takes advantage of the temporal correlation of each FAP parameter. Combining these two approaches results in a hybrid scheme that can achieve much better compression performance.

Based on predefined rules, the FIT method allows inputs with different number of FAPs in each frame. For example, in one frame, FAP for raising the left eyebrow presents but FAP for right eyebrow does not; in another frame, FAP for raising right eyebrow appears but the FAP for left eyebrow does not. A FIT for left-right duplication will easily handle both situations and interprets both frames correctly. The PCA method, on the other hand, requires no *a priori* knowledge about the data but accepts only one type of inputs. The same set of FAPs must appear in all frames. Neither FIT nor PCA introduces temporal latency. Different applications may choose the appropriate one for FAP dimension reduction.

Because predictive coding method can be lossless, it is the first candidate for temporal compression when fidelity is the major concern. However, because predictive coding only invests on de-correlating two consecutive frames, when FAP sampling rate is relatively high ($\geq 10$ Hz) and therefore strong correlation exist in each temporal segment, predictive method is much less efficient than DCT method.

Figure 13 shows the compression performance of various coding methods. We concluded that a combination of the PCA method and the DCT method gives the best performance at very low bit rates while the DCT method is superior to other methods at higher bit rates.

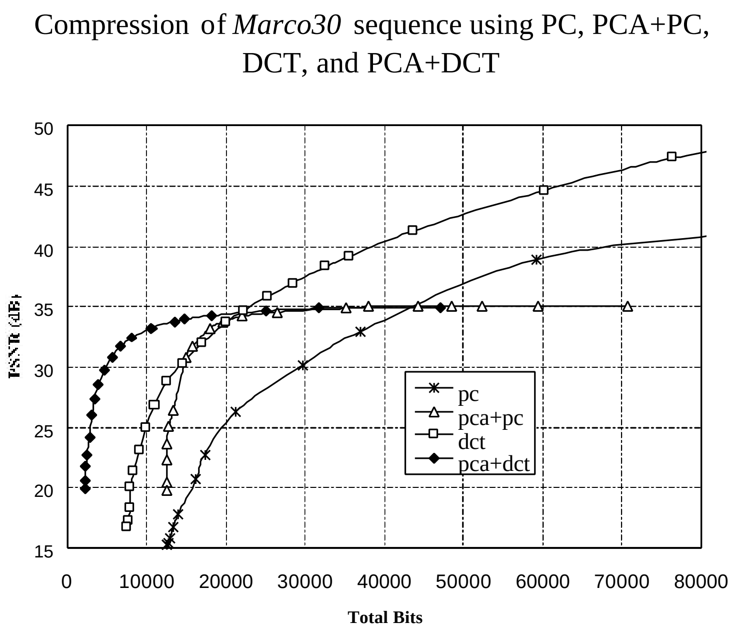


Figure 13: Compression results of the *Marco30* sequence.

## 4 CONCLUSIONS

In this paper, we described a model based video coding system based on the PBVD facial animation model. We described a PBVD model-based tracking algorithm and various compression techniques for encoding facial motion parameters. Our future research will focus on enabling more realistic facial animation and improving the accuracy and the robustness of the nonrigid motion estimation.

## ACKNOWLEDGMENTS

This work was supported in part by the Army Research Laboratory Cooperative Agreement No. DAAL01-96-0003.

## REFERENCES

[1] H. Li, P. Roivainen, and R. Forchheimer, “3-D motion estimation in model-based facial image coding,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 545-555, June 1993.

[2] C. S. Choi, K. Aizawa, H. Harashima, and T. Takebe, “Analysis and synthesis of facial image sequences in model-based image coding,” *IEEE Trans. Circuit Sys. Video Technol.*, vol. 4, no. 3, pp. 257-275, June 1994.

[3] F. I. Parke, “Parameterized models for facial animation,” *IEEE Comput. Graph. and Appl.*, vol. 2, no. 9, pp. 61-68, Nov. 1982.

[4] Y. Lee, D. Terzopoulos, and K. Waters, “Realistic modeling for facial animation,” in *Proc. SIGGRAPH 95*, 1995, pp. 55-62.

[5] Hai Tao and Thomas S. Huang, Explanation-based facial motion tracking using a piecewise Bézier volume deformation model, in *Proc. IEEE Comput. Vision and Patt. Recogn. (CVPR99)*, vol. 1, pp. 611-617, 1999.

[6] Hai Tao, Homer H. Chen, Wei Wu, and Thomas S. Huang, Compression of MPEG-4 facial animation parameters for transmission of talking heads, *IEEE Trans. Circuit and Sys. for Video Technol., vol. 9, no. 2,* pp. 264-276, March 1998.

[7] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” in *Proc. SIGGRAPH 86*, 1986, pp. 151-160.

[8] I. Essa and A. Pentland, “Coding, analysis, interpretation and recognition of facial expressions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 757-763, July 1997.

[9] D. DeCarlo and D. Metaxas, “The integration of optical flow and deformable models with applications to human face shape and motion estimation,” in Proc. CVPR '96, 1996, pp. 231-238.

[10] P. Kalra, A. Mangili, N. M. Thalmann, and D. Thalmann, ”Simulation of facial muscle actions based on rational free form deformations,” in *Proc. EUROGRAPHICS'92*, Sept. 1992, pp. 59-69.

[11] L. Williams, “Performance-driven facial animation,” in *Proc. SIGGRAPH 90*, Aug. 1990, pp. 235-242.

[12] “FAP Interpolation Table (FIT),” ISO/IEC JTC1/SC29/WG11 MPEG97/M2599, Aug. 1997.

[13] L. Sirovich and R. Everson, “Management and analysis of large scientific datasets,” *Int. J. Supercomput. Appl.*, vol. 6, no. 1, pp. 50-68, Spring 1992.

[14] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY: John Wiley & Sons, 1973.

[15] K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[16] “Text for CD 14496-2 Video,” ISO/IEC JTC1/SC29/WG11 N1902, Nov. 1997.