Compression of MPEG-4 Facial Animation Parameters for Transmission of Talking Heads

Hai Tao, *Member, IEEE,* Homer H. Chen, *Member, IEEE,* Wei Wu, and Thomas S. Huang, *Fellow, IEEE*

***Abstract*—The emerging MPEG-4 standard supports the transmission and composition of facial animation with natural video. The new standard will include 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 addresses the data-compression issue of talking heads and presents three methods for bit-rate reduction of FAP's. Compression efficiency is achieved by way of transform coding, principal component analysis, and FAP interpolation. These methods are independent of each other in nature and thus can be applied in combination to lower the bit-rate demand of FAP's, making possible the transmission of multiple talking heads over band-limited channels. The basic methods described here have been adopted into the MPEG-4 Visual Committee Draft [1] and are readily applicable to other articulation data such as body animation parameters. The efficacy of the methods is demonstrated by both subjective and objective results.**

***Index Terms*—Face animation, MPEG-4, synthetic and natural hybrid coding.**

I. Introduction

MANY applications in human–computer interface, three-dimensional (3-D) video games, model-based video coding [2], [3], talking agents [4], and distance learning demand rendering of realistic human faces [5]–[7]. In recent years, computer speed boosted by dramatic hardware improvement has made real-time rendering of face models possible. Recognizing the technology evolution in this field and its potential market value, MPEG is developing a new standard for animation and communication of talking heads. It is envisioned that this standardization will greatly accelerate the deployment of talking-head technology to a wide range of applications.

There are two possible approaches to communication of talking-head video. The pixel-based approach renders the facial images and transmits the resulting images as arrays of pixels, whereas the model-based approach transmits the facial animation parameters (FAP's) that describe the face motions and renders the images at the receiver. The latter approach is more appealing because it requires much less bandwidth and allows video editing and manipulation at the bitstream level. The model-based approach divides the task into geometric and articulation modeling. They are described by the MPEG-4 Synthetic/Natural Hybrid Coding (SNHC) group as the facial definition parameters (FDP's) and the FAP's, respectively. The geometric model defines the polygonal mesh of face and the associated skin texture from which visually realistic facial images from different view angles can be synthesized, and the articulation model deals with the deformation of static geometric models to generate various dynamic effects for intelligible reproduction of facial expressions. In an MPEG-4 talking-head transmission session, FDP data (if available) are transmitted at the setup stage to initialize the geometric model. Later, only FAP data are transmitted to deform the facial model [8]–[10] (Fig. 1).

Animation parameters, which have to be updated continuously to drive a face model, play a key role in model-based talking-head systems. The well-known facial action coding system (FACS) developed by Ekman and Friesen [11] describes the action of a group of muscles by an action unit. In their system, an articulation model converts the perceptually meaningful animation parameters to 3-D displacements of mesh vertices. Because the conversion is generally nonlinear and complicated, several approximation methods have been proposed and can be classified into four major categories: parameterized model [12]–[16], physical muscle model [17]–[20], free-form deformation model [21], and performance-driven animation model [22], [23].

The MPEG-4 standard will include a set of 68 FAP's that are defined based on the study of minimal facial actions and that are closely related to muscle movements [1]. The FAP set is a compromise solution that allows intelligible visual reproduction of facial expressions, emotions, and speech pronunciations at the receiver end but does not require the standardization of a geometric face model. Therefore, a manufacturer of MPEG-4 receivers can design its own proprietary face model to represent, for example, a company's identity while being able to communicate with any MPEG-4 face encoder. Depending on the applications, the FAP's can be obtained by analyzing the video sequence of a person [24]–[27], or by converting a text or phoneme stream to visual speech parameters [4].

The goal of this paper is to address the compression issue of FAP's. The FAP data must be compressed to a sufficiently low bit rate because after the setup stage, they dominate the

Manuscript received October 1, 1997; revised November 1, 1998. This paper was recommended by Guest Editors H. H. Chen, T. Ebrahimi, G. Rajan, C. Horne, P. K. Doenges, and L. Chiariglione.

H. Tao was with the Beckman Institute, University of Illinois, Urbana, IL 61801 USA. He is now with the Vision Technologies Laboratory, Sarnoff Corp., Princeton, NJ 08543 USA.

H. H. Chen and W. Wu are with the Rockwell Science Center, Thousand Oaks, CA 91360 USA.

T. S. Huang is with the Beckman Institute, University of Illinois, Urbana, IL 61801 USA.

Publisher Item Identifier S 1051-8215(99)02339-3.

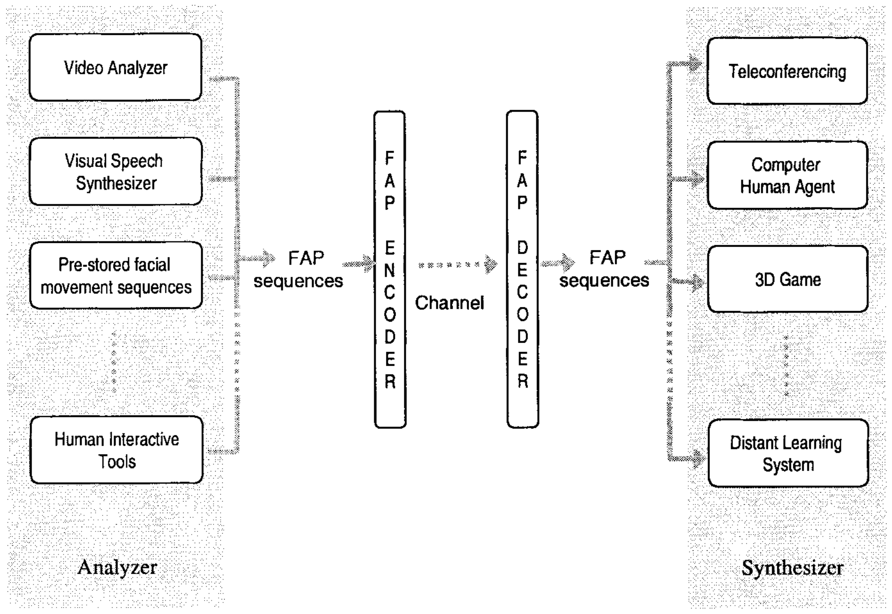


Fig. 1. An FAP-driven talking-head system.

usage of the channel and because most applications envisaged have a low bandwidth constraint. As an illustration, assuming that ten bits are used for each parameter and the frame rate is 30 Hz, the raw FAP data rate without compression can reach as high as 20 kbit/s for a single talking head if all 68 FAP's are sent. Handling more than two talking heads is beyond the capability of an ordinary computer modem (56 kbit/s). Thus, the FAP data have to be compressed. The FAP compression methods described here employ techniques such as transform coding, principle component analysis, and interpolation to remove temporal or spatial data redundancy. These methods are independent of each other and hence can be applied together to achieve very high data compression for talking-head video.

The remaining sections are organized as follows. A brief description of the MPEG-4 FAP's and the associated grouping for efficiency purposes is given in Section II. The subject of FAP interpolation for data reduction and a general scheme for representation of FAP interpolation are discussed in Section III. Compression methods that remove spatial or temporal data redundancy are described in Sections IV–VI. The performance of the proposed methods, including both objective and subjective experimental results, is described in Section VII, followed by concluding remarks in Section VIII.

II. MPEG-4 Facial Animation Parameters

As described earlier, MPEG-4 adopts a model-based approach that allows user-defined face models to communicate with each other without requiring the standardization of a common face model. The result of this approach is the definition of 68 facial animation parameters as the basic data set that must be supported by all MPEG-4 face decoders. Among the 68 FAP's, two are high-level parameters [visual phoneme (viseme) and expression] and the others are low-level parameters that describe the movements of facial features defined over jaw, lips, eyes, mouth, nose, cheek, ears, etc. Unlike the low-level parameters, the viseme and expression parameters describe facial movements at an abstract level, and each is a compound parameter consisting of multiple subparameters [1].

The movement represented by each FAP is defined with respect to a neutral face and expressed in terms of the facial animation parameter units (FAPU's); see Fig. 2. The FAPU's correspond to fractions of the distances between some salient facial features, such as eye separation (ES0), mouth-nose separation (MNS0), etc. These units are defined in order to allow a consistent interpretation of FAP's on any face model. Each FAP represents a one-dimensional measurement. For example, the third FAP `open_jaw` defines the vertical displacement of the jaw and is unidirectional, with an FAPU MNS equal to MNS0/1024. A positive value represents downward motion. Details of these definitions are described in [1].

The FAP's specified by MPEG-4 are intended to be exhaustive. Typically, not all the FAP's are active in a face animation session. Furthermore, some FAP's tend to appear more often or to require more precision than the others. For the purpose of bit savings, therefore, it is advantageous to divide the FAP's into groups, each with its own quantization step size and FAPU. A talking-head encoder will then have to update only the active groups of FAP's. The complete FAP's and groupings adopted by MPEG-4 are listed in Table I. By the same token, not all FAP's within a group are active. Therefore, further data reduction can be accomplished by employing a mask to indicate which FAP's are active, and only the active FAP's within each group are transmitted or updated. By applying the masking to the test sequence *Aleta*, for example, the number of FAP's required for transmission decreases to 19, and the resulting bit rate drops by a factor of 3.5 with respect to the

TABLE I
THE TEN GROUPS OF MPEG-4 FACIAL ANIMATION PARAMETERS

Groups	FAP #	FAP's
1	1-2	viseme, expression
2	3-18	open_jaw, lower_t_midlip, raise_b_midlip, stretch_l_cornerlip, stretch_r_cornerlip, lower_t_lip_lm,lower_t_lip_rm, raise_b_lip_lm, raise_b_lip_rm, raise_l_cornerlip, raise_r_cornerlip, thrust_jaw, shift_jaw, push_b_lip, push_t_lip, depress_chin
3	19-30	close_t_l_eyelid, close_t_r_eyelid, close_b_l_eyelid, close_b_r_eyelid, yaw_l_eyeball, yaw_r_eyeball, pitch_l_eyeball, pitch_r_eyeball, thrust_l_eyeball, thrust_r_eyeball, dilate_l_pupil, dilate_r_pupil
4	31-38	raise_l_i_eyebrow, raise_r_i_eyebrow, raise_l_m_eyebrow, raise_r_m_eyebrow, raise_l_o_eyebrow, raise_r_o_eyebrow, squeeze_l_eyebrow, squeeze_r_eyebrow
5	39-42	puff_l_cheek, puff_r_cheek, lift_l_cheek, lift_r_cheek
6	43-47	shift_tongue_tip, raise_tongue_tip, thrust_tongue_tip, raise_tongue, tongue_roll
7	48-50	head_pitch, head_yaw, head_roll
8	51-60	lower_t_midlip_o, raise_b_midlip_o, stretch_l_cornerlip_o, stretch_r_cornerlip_o, lower_t_lip_lm_o, lower_t_lip_rm_o, raise_b_lip_lm_o, raise_b_lip_rm_o, raise_l_cornerlip_o, raise_r_cornerlip_o
9	61-64	stretch_l_nose, stretch_r_nose, raise_nose, bend_nose
10	65-68	raise_l_ear, raise_r_ear, pull_l_ear, pull_r_ear

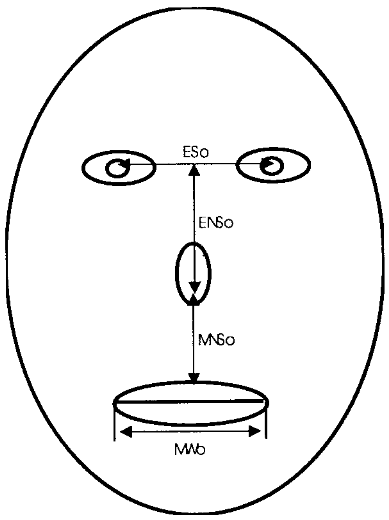


Fig. 2. The FAP units.

raw data rate. Grouping and masking are simple but effective methods for reducing the amount of FAP data to be transmitted without introducing much computational cost. However, the spatial-temporal redundancy among FAP's is not exploited. As can be seen later, the methods presented in this paper compress the FAP data by an additional factor of ten or even more.

The four major components in an MPEG-4 facial animation system are an analyzer that generates FAP's, an encoder that compresses FAP's and the FDP's, a decoder that decodes the bitstream, and an image synthesizer that renders the face based on the reconstructed FAP data. The FDP's allow the definition of a precise facial shape, skin texture, and animation rule in the setup stage when so desired. Details of the FDP's are specified in the MPEG-4 systems committee draft [28] and are beyond the scope of this paper.

III. FAP INTERPOLATION

One way to achieve data reduction for talking heads is to send only a subset of active FAP's. This subset is then used to determine the values of other FAP's. Such FAP interpolation exploits the symmetry of a human face or the *a priori* knowledge of articulation functions. For example, the top-inner-lip FAP's can be sent and then used to determine the top-outer-lip FAP's. The inner-lip FAP's would be mapped to the outer-lip FAP's by interpolation. Those two contours are not identical, but one can be approximately derived from the other while still achieving reasonable visual quality. FAP interpolation is a desirable tool to overcome channel bandwidth limitation. It is also useful for data recovery where a face decoder needs to determine the values of missing FAP's caused by, for example, imperfect face feature extraction at the encoder or by packet loss during data transmission.

In practice, it is critical that the decoder interpolate FAP's 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 coders must comply. However, the relations among FAP's vary from person to person. It is generally difficult to decide a set of fixed interpolation rules that fits all faces. Further, there are so many possible ways of interpolating FAP's, depending on which FAP needs interpolation and which FAP's are available. It is virtually impossible to exhaustively define all of them. A more plausible approach using FAP

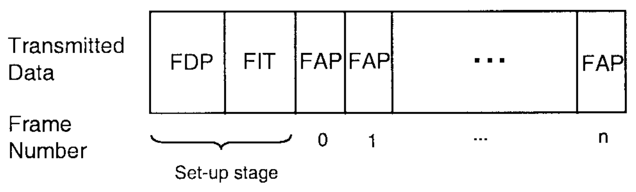


Fig. 3. Data transmitted in a face animation session.

interpolation table (FIT) is proposed in [29]. 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 specifies both interpolation syntax and interpolation functions. The interpolation syntax describes from which other FAP's an FAP can be interpolated. The interpolation functions describe the mathematical relations. The two major elements of FIT are an FAP interpolation graph (FIG), which describes the interpolation syntax, and rational polynomials, which specify the interpolation functions. FIG is an efficient scheme capable of describing complicated relations between FAP's, while the multivariable rational polynomial is capable of representing both linear and nonlinear interpolations and is general enough to describe FAP-to-mesh interpolation as well. The FAP interpolation table is also downloaded in the setup stage when FAP interpolation is required in the face animation session, as shown in Fig. 3.

A. FAP Interpolation Graph

To make the scheme general, sets of FAP's are specified, along with a graph between the sets that specifies which sets are used to determine which other sets. In some situations, a set of FAP's can be determined from more than one other set of FAP's, in which case the links that determine the relationship between sets of FAP's also have a priority.

The FIG is a graph with directed links. Each node contains a set of FAP's. Each link from a parent node to a child node indicates that the FAP's in a child node can be interpolated from a parent node provided that all the required FAP's 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 latter are ordered as first parent node, second parent node, etc. During the interpolation process, if this child node needs to be interpolated, it is interpolated from its first parent node if all required FAP's in that parent node are available. Otherwise, it is interpolated from its second parent node, and so on.

An example of FIG for representing the interpolation of inner- and outer-lip contours is shown in Fig. 4. Each node has an ID. The numerical label on each incoming link indicates the order of these links. For example, consider the `top_inner_lip` FAP's in node 3 (ID = 3). These FAP's can be interpolated from either node 1 or node 4. Since the priority of the link from node 4 is one, higher than that of the link from node 1, interpolation from node 4 is performed first. Once an FAP is interpolated, it is considered available and can be used to interpolate other FAP's.

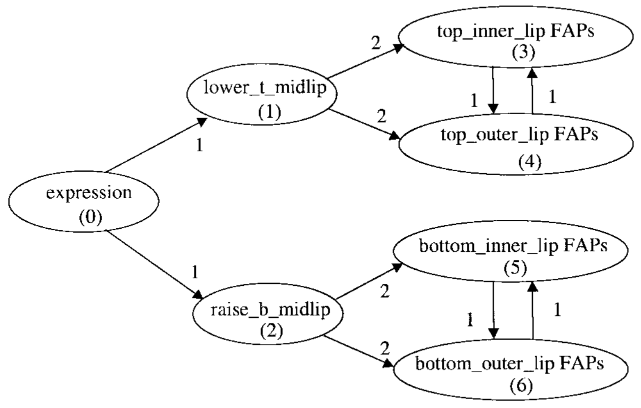


Fig. 4. An FAP interpolation graph.

In general, an FAP interpolation process based on the FIG can be described using pseudo C code as follows:

```
do {
    interpolation_count = 0;
    for ( eachNode_i) { /* from Node_0
    to Node_N−1 */
        if ( some FAPs in Node_i
        need interpolation) {
            for (each ordered Node_i's
            parent nodes Node_k) {
                if ( all FAPs are
                available in Node_k) {
                    interpolate FAPs
                    in Node_i from
                    FAPS in Node_k;
                    /* using interpola-
                    tion function */
                    interpolation_count
                    ++;
                    break;
                }
            }
        }
    }
} while (interpolation_count != 0);
```

Both encoder and decoder must follow the same interpolation process in order to guarantee identical results.

B. Interpolation Functions

Each directed link in FIG represents a set of interpolation functions that determine the values of child FAP's. Suppose $F_1, F_2, \cdots, F_n$ are the FAP's in a parent node and $f_1, f_2, \cdots, f_m$ are the FAP's in a child set. Then, there are m interpolation functions denoted as

$$\begin{aligned} f_1 &= I_1(F_1, F_2, \cdots, F_n) \\ f_2 &= I_2(F_1, F_2, \cdots, F_n) \\ &\cdots \\ f_m &= I_m(F_1, F_2, \cdots, F_n). \end{aligned} \tag{1}$$

Each interpolation function $I_t()$ is in a rational polynomial form if the parent node does not contain viseme FAP or

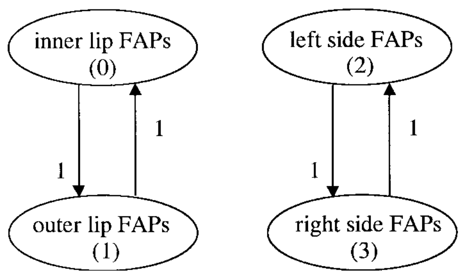


Fig. 5. An example of FAP interpolation using FIT.

expression FAP

$$I(F_1, F_2, \cdots, F_n) = \sum_{i=0}^{K-1} \left(c_i \prod_{j=1}^{n} F_j^{l_{ij}} \right) \Bigg/ \sum_{i=0}^{P-1} \left(b_i \prod_{j=1}^{n} F_j^{m_{ij}} \right). \quad (2)$$

Otherwise, an impulse function is added to the numerator to allow selection of expression or viseme

$$I(F_1, F_2, \cdots, F_n) = \sum_{i=0}^{K-1} \delta(F_{s_i} - a_i) \left(c_i \prod_{j=1}^{n} F_j^{l_{ij}} \right) \Bigg/ \sum_{i=0}^{P-1} \left(b_i \prod_{j=1}^{n} F_j^{m_{ij}} \right) \quad (3)$$

where K and P are the numbers of polynomial products, c_i and b_i are the coefficient of the ith product, and l_{ij} and m_{ij} are the power of F_j in the ith product. An impulse function equals one when $F_{s_i} = a_i$; otherwise, the function equals zero. The variable F_{s_i} can only be `viseme_select1`, `viseme_select2`, `expression_select1`, and `expression_select2`. The variable a_i is an integer that ranges from zero to six when F_{s_i} is `expression_select1` or `expression_select2`, and ranges from zero to 14 when F_{s_i} is `viseme_select1` or `viseme_select2`. The encoder should send an interpolation function table that contains all K, P, s_i, a_i, c_i, b_i, l_{ij}, and 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.

C. Examples of FIT

Two more examples of FIT are given below for illustration purposes. The first example describes the interpolation between left and right FAP's and the interpolation between inner- and outer-lip contours. The FIG of this example is shown in Fig. 5. There are four nodes with ID's from zero to three. Nodes 0 and 1 each have ten FAP's representing inner (FAP_4–FAP_{13}) and outer (FAP_{51}–FAP_{60}) lip contours, respectively. Similarly, there are 23 FAP's in both nodes 2 and 3 representing right-side and left-side faces, respectively.

The interpolation functions between nodes 2 and 3 are simply duplications. Thus, we have $f_i = F_i, \ i = 1, 2, \cdots, 23$, where F_1–F_{23} denote right-side or left-side FAP's and f_1–f_{23} denote FAP's of the other side. Interpolation functions between inner-lip and outer-lip FAP's are more complicated and vary significantly from the individual face. An interpolation function from `lower_t_midlip` (denoted as F_1) to

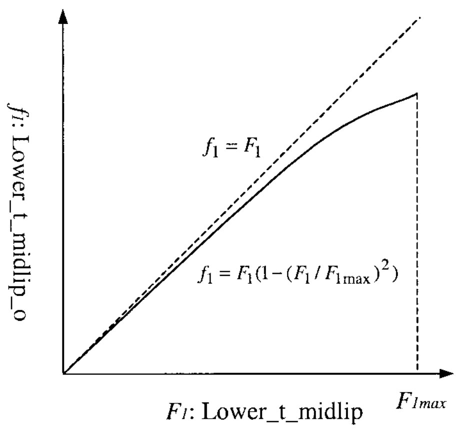


Fig. 6. Interpolation of `lower_t_midlip_o` from `lower_t_midlip`.

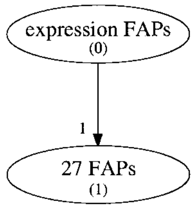


Fig. 7. An FIT for describing an expression.

`lower_t_midlip_o` (denoted as f_1) in the form of $f_1 = F_1(1 - (F_1/F_{1_{\max}})^2)$, where $F_{1_{\max}}$ is the maximum range of F_1, simply emulates the effect of upper-lip thinning when it is raised (Fig. 6).

The second example illustrates the usage of FIT for defining a high-level expression FAP in terms of other low-level FAP's. Fig. 7 shows a FIG that accomplishes this task. There are two nodes. Node 0 contains four subparameters of the expression FAP, the two expression identities FAP_{72} and FAP_{73} (denoted as F_1 and F_2), and their intensities FAP_{74} and FAP_{75} (denoted as F_3 and F_4). Variables F_1 and F_2 range from one to six, representing one of the six expressions: joy, sadness, anger, fear, disgust, and surprise. Variables F_3 and F_4 range from zero to 63, indicating the intensities of these two expressions. Node 1 contains 27 low-level FAP's for this specific implementation. Thus, a total of 27 interpolation functions need to be defined. Each function is in the form $f_i = \sum_{j=1}^{6} \delta(F_1 - j) p_{ij} F_3 + \sum_{j=1}^{6} \delta(F_2 - j) p_{ij} F_4; \ i = 1, \ 2, \cdots, 27$, where $\delta(F_1 - j)$ is an impulse function that equals one when $F_1 = j$ and equals zero otherwise. For a particular expression, or equivalently, a particular j, p_{ij}, $i = 1, 2, \cdots, 27$ are low-level FAP values that achieve that expression. These values are either extracted from real data using the principal component analysis (PCA) technique, which will be discussed in the next section, or obtained from interactive animation tools.

IV. Reduction of FAP Spatial Redundancy

A more general tool for exploiting both deterministic and statistical correlation among FAP's is PCA [30], which converts the original FAP data to a new compact form. Application of this technique 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 deduce, statistically a strong correlation exists.

To apply the PCA technique, major axes are computed as the eigenvectors of the covariance matrix computed from FAP vectors. Each FAP vector is formed by FAP's 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 in which most portions of FAP data are reserved. Compact representation is obtained by projecting the original FAP vector into this subspace, and then the new representation is transmitted through the channel. Projecting the transmitted data back to the FAP space produces good approximations of the original FAP vectors. The projection process is also called the Karhunen–Loeve transform (KLT) [31].

To formalize the above process, suppose the original FAP vectors are $v_1, v_2, \cdots, v_n$, and each v_i is a column vector that contains m FAP's in a particular frame. Then, the $m \times m$ covariance matrix is computed as

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(v_i - \overline{v})(v_i - \overline{v})^t \tag{4}$$

where $\overline{v}$ is the mean of v_i. Since for most FAP's the average positions are at neutral expression, or zero, the covariance matrix is simply written as

$$C = \frac{1}{n-1}\sum_{i=1}^{n} v_i v_i^t. \tag{5}$$

Since C is a nonnegative definite matrix, all eigenvalues are nonnegative real values. We denote them in descending order as $\lambda_1, \lambda_2, \cdots, \lambda_m$ and their corresponding normalized eigenvectors as $u_1, u_2, \cdots, u_m$. Suppose that the first k eigenvalues are significantly large, or that the percentage of energy $\alpha = \sum_{i=1}^{k}\lambda_i / \sum_{i=1}^{m}\lambda_i$ exceeds a certain threshold; these k eigenvectors then form a subspace that preserves most of the information in v_i. Each v_i is transformed into this new subspace by performing a linear transformation

$$q_i = \begin{bmatrix} u_1^t \\ u_2^t \\ \vdots \\ u_k^t \end{bmatrix} v_i. \tag{6}$$

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

$$\hat{v}_i = [u_1 \quad u_2 \quad \cdots \quad u_k] q_i. \tag{7}$$

PCA reduces dimension dramatically for FAP data with strong spatial correlation. Although some new components of q_i may possess larger data ranges and need more bits for coding, overall, significant bit savings are achieved. Also (in addition to q_i), u_j, $j = 1, \cdots, k$, for each FAP sequence needs to be sent in the setup stage to ensure that the decoder correctly recovers v_i. For a low-bandwidth system with limited room for downloading, a set of universal major axes u_j is pursued 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 that sample various facial movements.

Another important issue that deserves further discussion is group PCA. It is well known that PCA provides meaningful results only if the measurement of each vector element has approximately equal significance, which is not true for FAP. For example, the numerical ranges of the eye-movement FAP's are much larger than those of the FAP's around the mouth. Their measurements have remarkably different significance. A coding error of 50 FAPU's on eyeball-rotation FAP's may leave no trace in the animated images; however, visually irritating results are observed if this amount of error is in any mouth FAP. A meaningful PCA is achieved only inside each FAP subgroup, in which FAP values have similar significance.

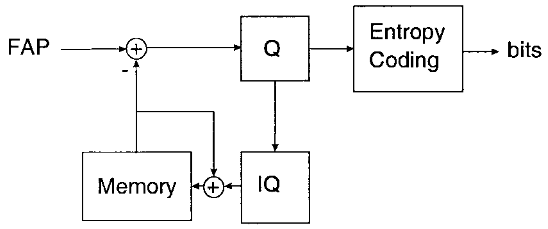


Fig. 8. Block diagram of a predictive coder.

V. Reduction of FAP Temporal Redundancy

In a temporal sequence of each FAP parameter or each PCA major component q_i, strong interframe correlation exists. Actually, the similarity between consecutive temporal frames appears in most animation parameters and can be explained as the result of the inertia factors existing in all mechanical systems. Compression techniques in temporal domain exploit this characteristic to achieve bit savings. Two schemes discussed in this section are the predictive coding (PC) and discrete cosine transform (DCT) methods [32].

A. PC Method

Instead of transmitting the parameters themselves, the differences between consecutive frames are encoded and sent. Because neighboring frames for each parameter contain similar values, the differences between the parameters tend to be smaller quantities centering around zero. Fewer bits are needed to represent these differences as the result of their smaller data ranges and concentrated distributions. The block diagram of a variable-length predictive encoder is shown in Fig. 8.

For each FAP, its encoded value in the previous frame is used as the prediction for its current value. Because both decoder and encoder use this same prediction, 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 and is the bit-saving source.

Intraframe coding, on the contrary, directly encodes the quantized FAP value. It is equivalent to set the value in "Memory" to zero. 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 the FAP predictive coding scheme is how to set the appropriate quantization step size for each FAP. Both its visual sensitivity to errors and its original data

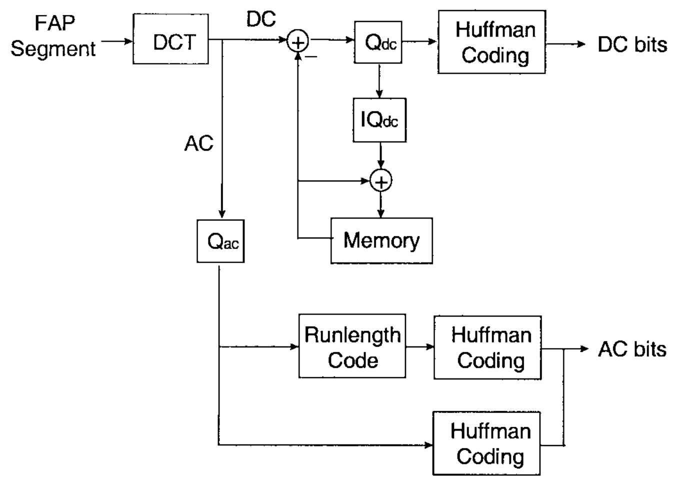


Fig. 9. FAP coding using the DCT method.

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 resulting animation. Equivalently, this problem is also addressed as, for a given transmission bandwidth, how to achieve the best visual results by adjusting the bit distributions of each parameter through setting its quantization step. Extensive empirical results are indispensable to deduce a plausible solution.

Because there is no temporal latency, the predictive coding scheme is suitable for real-time applications. In many other situations, however, a small decoding latency in exchange for a higher compression is tolerable. An example of this type of application is a multimedia mailing system with a front-end talking agent, where the coded FAP sequences are usually stored on a disk. For such applications, the transform coding method is an appropriate choice.

B. FAP Coding Using DCT

At video rates such as 30 or 25 Hz, strong correlation exists not only between consecutive frames but also among multiple neighboring frames. KLT is applicable for decorrelating original data into a compact representation for data reduction in a manner similar to that described in Section IV. However, the temporal causal relations, or Markov properties, exist in facial articulation. It is well known that when a signal is Markovian, the decorrelation efficiency of faster and simpler DCT approaches that of KLT. DCT converts the original data into their frequency domain representations, namely, dc and ac coefficients. Since FAP data have relatively low frequencies, most high-frequency ac coefficients are small and can be discarded. This dramatically reduces the amount of data that need to be transmitted.

A block diagram of the proposed DCT coder is shown in Fig. 9. For each FAP parameter or PCA component, the temporal sequence is decomposed into segments. Each segment contains 16 frames. A one-dimensional length-16 DCT is then applied to these individual segments.

After DCT, the resulting dc and ac coefficients are quantized. Similar to the predictive coding scheme, for each FAP parameter, a particular quantization step is specified according

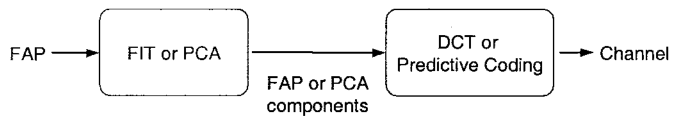


Fig. 10. A hybrid FAP encoding scheme.

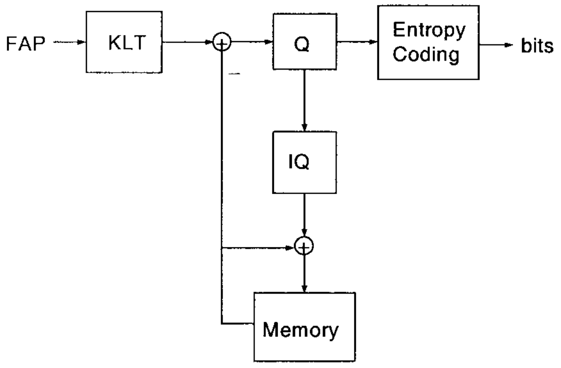


Fig. 11. Block diagram of a PCA + PC coder.

to its perceptual sensitivity to error. Because the dc coefficient is the mean value of the segment and is prone to error, its quantization step size is three times smaller than that of ac coefficients.

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

For the nonzero ac coefficients in each segment, their positions and values need to be encoded. To encode their positions, a run-length coder 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 zero to 14. Therefore, taking the `end_of_segment` symbol into account, the Huffman table of the run-length coder contains 16 symbols. The values of nonzero ac coefficients are then encoded using a Huffman coder.

As in the predictive FAP coding scheme, quantization steps need to be carefully assigned to each parameter. Again, empirical results are used to determine these values. To further exploit the human perceptual properties, different quantization steps are assigned to different ac coefficients. Subjective experiments need to be conducted on the resulting animation. A set of DCT quantization steps are specified in the MPEG-4 Visual Committee Draft [1].

VI. Reduction of FAP Spatial and Temporal Redundancy

Compression methods in spatial domain are orthogonal to compression schemes in temporal domain in the sense that the former exploit the correlation among FAP's in a single frame, whereas the latter take advantage of the temporal correlation of each FAP parameter. They can be combined in a hybrid scheme to achieve higher compression performance. Fig. 10 describes possible ways of configuring this type of scheme.

(a)

(b)

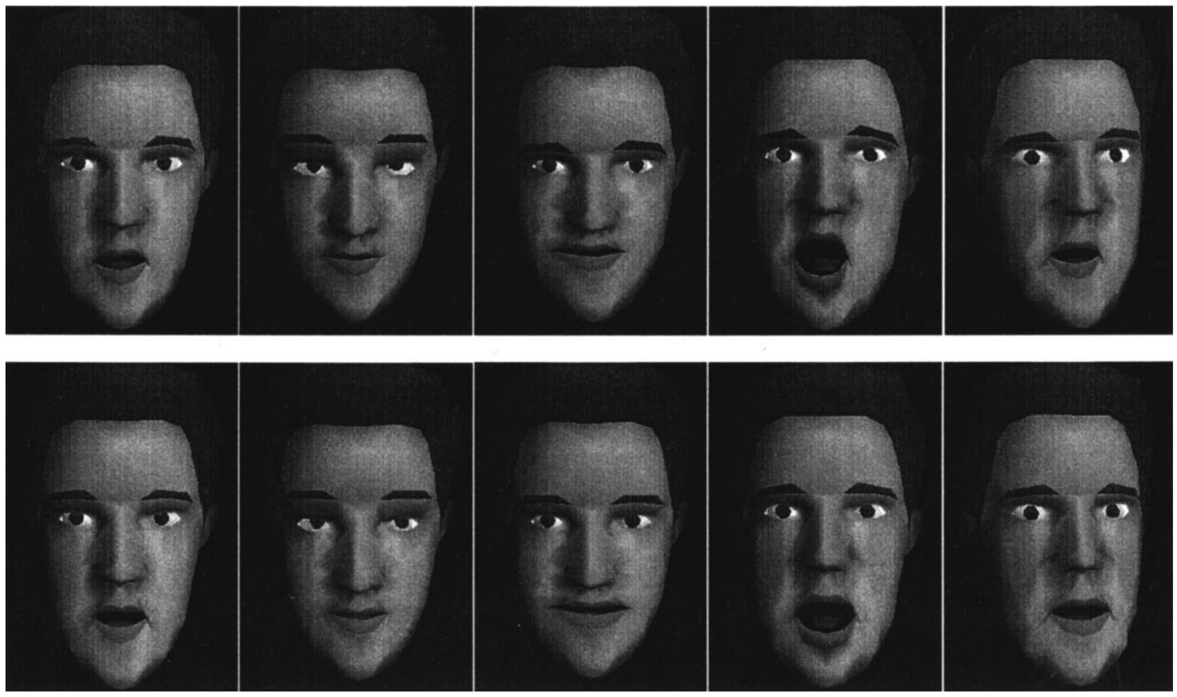


Fig. 12. Using FIT on *Marco30* sequence. Frames from the (a) original sequence and (b) decoded sequence.

FIT allows variable FAP inputs in each frame. For example, in one frame, the FAP for raising the left eyebrow exists but the FAP for the right eyebrow does not; in another frame, the FAP for raising the right eyebrow appears but the FAP for the left eyebrow does not. A FIT for left–right duplication will easily handle both situations and interprets both frames correctly. PCA, on the other hand, requires no *a priori* knowledge about the data but the same set of FAP's must appear in all frames. Neither FIT nor PCA introduces temporal latency. Different applications may choose the appropriate one for FAP dimension reduction. Fig. 11 shows the block diagram of a hybrid scheme using PCA and PC. "Q" is a quantizer and "IQ" is an inverse quantizer. Prediction is stored in "Memory."

Because the predictive coding method can achieve lossless coding, it is the first candidate for temporal compression when fidelity is the major concern. However, when the FAP sampling rate is relatively high ($\geq$10 Hz) and therefore strong correlation exists in each temporal segment, the predictive method is less efficient than the DCT method.

VII. Experimental Results

Results on three FAP test files are shown in this section. In all test files, the FAP's are sampled at 30 frames/s. The first sequence *Marco30* contains 1542 frames, 32 active FAP's per frame. The facial movements in this sequence include global head motions, expressions, and speech. The second sequence *Emotions* has 1050 frames and 27 active FAP's in each frame and contains six facial expressions. The third sequence *Aleta* contains 300 frames and 19 mouth-related FAP's in each frame. Only speech contents appear in the sequence.

For objective comparison, the following formula is used to compute the peak signal-to-noise ratio (PSNR) between the original and the reconstructed FAP data:

$$\mathrm{PSNR} = 10 \cdot \log\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{R_i^2}{\mathrm{MSE}_i^2}\right)\right) \tag{8}$$

where N is the number of active FAP's, R_i is the range of the ith FAP parameter, and MSE_i is the mean square error of the decoded ith FAP sequence.

To inspect the results subjectively, the original and the reconstructed FAP data are fed into a face-animation program provided by MIRALAB of EPFL, Switzerland. Rendered animation sequences are visually compared side by side to evaluate the perceptual effects of distortion. Even at the same PSNR and bit rate, animation results can be dramatically different if the bits are distributed to each parameter differently. Many subjective tests have been performed to choose a set of FAP quantization steps that consequently leads to visually optimal bit distributions. A general observation is that humans are very sensitive to distortion in global motion FAP's and FAP's around the eyes and mouth. With this factor in consideration, relatively smaller quantization steps are applied to these FAP's. A set of FAP quantization steps is defined in [1].

A. FIT Results

Experiments on FIT have been conducted on the *Marco30* and *Emotions* sequences. By defining left side to right side, inner lip to outer lip, bottom-mid lip to open jaw, and middle eyebrow to side eyebrow interpolations, only 13 of the original 32 FAP's in *Marco30* sequence need to be transmitted. The interpolation functions are either duplications or in the form described in Section III-C. This process dramatically reduces the data volume by a factor of 2.5. Then, from these 13 FAP's, 19 untransmitted FAP's are interpolated. The resulting FAP data are compared subjectively with the original data. As shown in Fig. 12, the first row contains frames from the

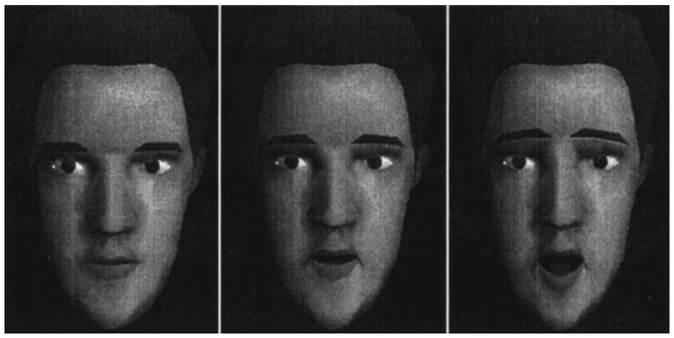


Fig. 13. FIT results for expression "surprise." Only expression FAP is sent for each frame, and the 27 low-level FAP's are interpolated from it using FIT.

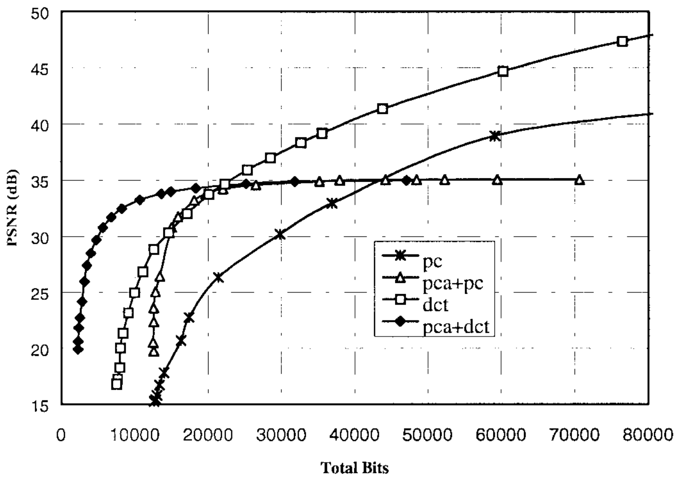


Fig. 14. Compression results of *Marco30* sequence.

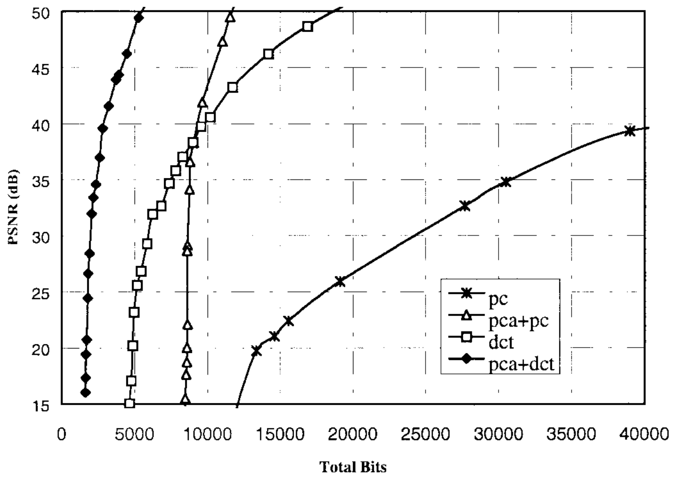


Fig. 15. Compression results of *Emotions* sequence.

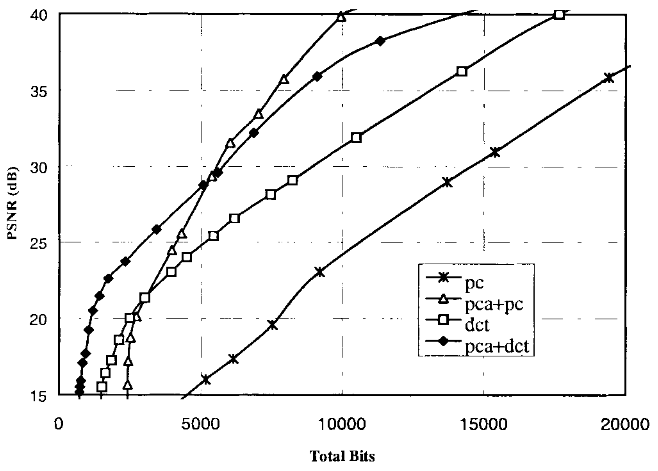


Fig. 16. Compression results of *Aleta* sequence.

sequence generated by animating the original *Marco30* data; the second row includes results by FAP interpolation. The difference is almost unnoticeable.

In the second experiment, the expression "surprise" in the *Emotions* sequence is analyzed using PCA and FIT, as described in Section III-C. The interpolation from FAP2 (expression) to 27 other FAP's (FAP3–FAP13, FAP19–FAP26, and FAP31–FAP38) is derived. The data-reduction ratio is 27 to 4. Note that there are four subfields in the expression FAP. Again, as shown in Fig. 13, convincing animation results are achieved. An important conclusion of this experiment is that FIT is also a suitable scheme for transmitting the definition of expression or viseme FAP's during the setup stage of an animation session.

B. PC Method

The PC algorithm is implemented and tested on all three sequences. For each FAP parameter, we use the corresponding quantization step defined in the MPEG-4 Visual Committee Draft [1]. The final quantization step size is determined by multiplying each FAP's quantization step by `fap_quant`. The `fap_quant` is used to adjust the total bit consumption, and the quantization step of each individual FAP controls the bit distributions among different FAP's. By adjusting `fap_quant`, a PSNR versus total-bit-number curve is obtained for each FAP sequence. Such curves are shown in Figs. 14–16, labeled "PC". In these figures, the horizontal axis is the total number of bits for each sequence; the vertical axis indicates the PSNR values. Eventually, the predictive method becomes lossless at a high enough bit rate (not shown). It achieves better PSNR than other methods. The predictive coding scheme is simple to implement and is used as the base method for all performance comparisons.

C. Results of PCA Method

The PCA results obtained show that to preserve 98% of the signal energy, six principal components are needed for the *Aleta* sequence, seven components for the *Emotions* sequence, and nine for the *Marco30* sequence. These values are determined by applying PCA independently to each sequence. To obtain a universal KLT, all sequences have to be considered simultaneously. Our results show that around ten major components are required in a universal basis for these three sequences. This implies a dramatic cut in bit rate for the *Marco30* and *Emotions* sequences. For the *Aleta* sequence, since after KLT most values of those ten components are very small, further savings are achieved by applying a temporal compression scheme to them. The bit savings of the PCA method is due to the compactness of representation.

Results of the PCA method concatenated with a temporal predictive coding scheme on the three test sequences are

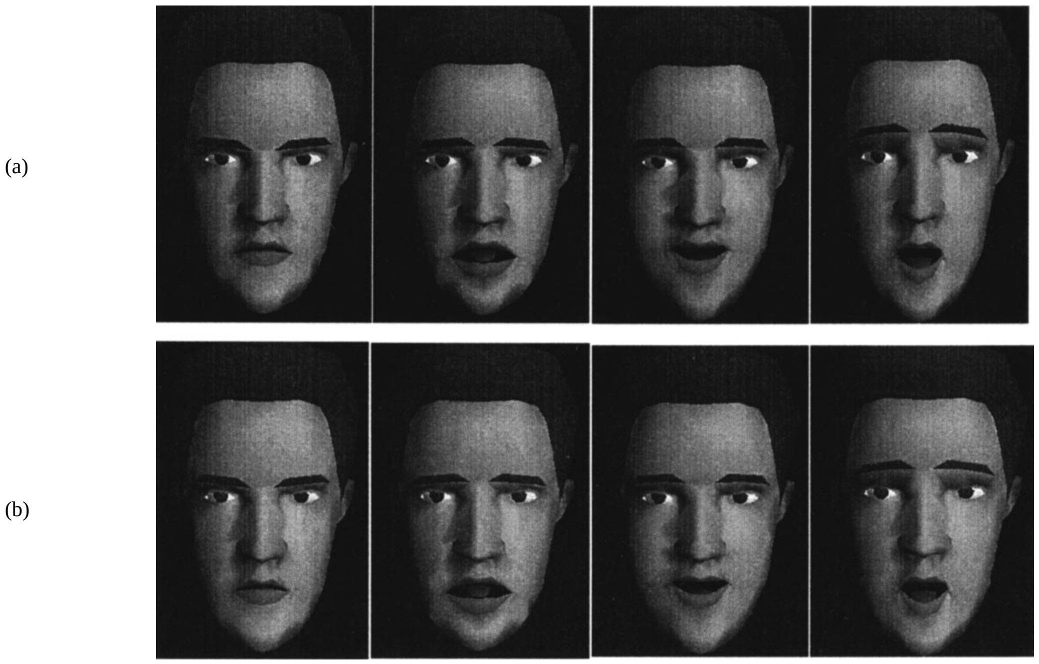


Fig. 17. PCA + PC results of *Emotions* sequence (PSNR = 34 dB, bit rate = 0.24 kbit/s). Frames from the (a) original and (b) decoded sequence.

shown in Figs. 14–16, marked with a PCA + PC label. By adjusting the quantization scalar factor for all FAP's, different bit rates are obtained. Since not all transformed components are kept, this method is always lossy. PSNR versus bit-rate curves of the PCA method tend to be flat at high bit rates. This is caused by the fact that not all components are reserved in the coding process. No matter how many bits are allocated for encoding these components, the errors caused by the discarded components are unavoidable. At higher bit rates, where the quantization error is small, PSNR increases when more PCA components are kept.

At low bit rates, the PCA + predictive coder outperforms the predictive coder by approximately 3–10 dB. This is because fewer parameters are encoded after KLT. Though some parameters may use more bits than the original FAP, the total number of bits decreases for the same PSNR. Fig. 15 shows that at 0.24 kbit/s (or a total of 8500 bits), a PSNR of 34 dB is achieved when nine components are transmitted after KLT. Similar bit savings are demonstrated in Figs. 14 and 16.

In Fig. 17, frames from the animated *Emotions* sequence are shown. The first row is created from the original *Emotions* sequence. The second row contains animation frames from a reconstructed FAP sequence with a bit rate of 0.24 kbit/s and a PSNR of 34 dB. More experimental results are also demonstrated in [33].

D. DCT Method

To implement the proposed DCT method, the temporal segment length, the quantization steps for each FAP, the predictive coding scheme for dc coefficients, and the coding method for the positions and values of nonzero ac coefficients have to be determined.

Currently, segment length 16 is determined from empirical results. When the segment length is too large, more nonzero ac coefficients appear because FAP data in each segment are weakly correlated. This reduces the overall compression performance. On the other hand, when the segment length is too small, more dc coefficients need to be coded, and the total number of nonzero ac coefficients also increases. These two factors combined affect the total efficiency.

As mentioned in Section V-B, the quantization step for each FAP parameter is deduced according to a perceptual test. Currently, a single quantization step is applied to all ac coefficients in each segment. A more sophisticated quantization scheme based on experimental results will achieve an even better compression ratio. Another issue of the DCT scheme is concerned about the quantization step for dc coefficients and the quantization step for ac coefficients. If the quantization step for dc coefficients is too small (relatively), more bits will be allocated for the dc coefficients but not enough bits for the ac coefficients. As a result, the animated face may lack the detailed movements and therefore is less expressive. On the other hand, if the quantization step for dc coefficients is too large, jerky motion will be observed at segment boundaries. From our experiments, the dc quantization step should be about three times smaller than the ac quantization step.

The dc predictive error falling in the range of -255 to 255 is assigned a symbol in the Huffman table. An "ESC" symbol is defined if it exceeds this range. More bits are then allocated to encode this value. As a result, the Huffman table for dc predictive errors contains 512 entries. Coding of ac values uses a similar strategy, and the corresponding Huffman table also contains 512 symbols.

The performance of the proposed DCT method is shown in Figs. 14–16, labeled with "DCT." It is found that when PSNR is around 30 dB, the average total bit number is about 50% less than that of the predictive coding method. Fig. 18 shows (a) the rendered sequences using the original *Marco30* sequence and (b) the reconstructed DCT sequence at a bit

(a)

(b)

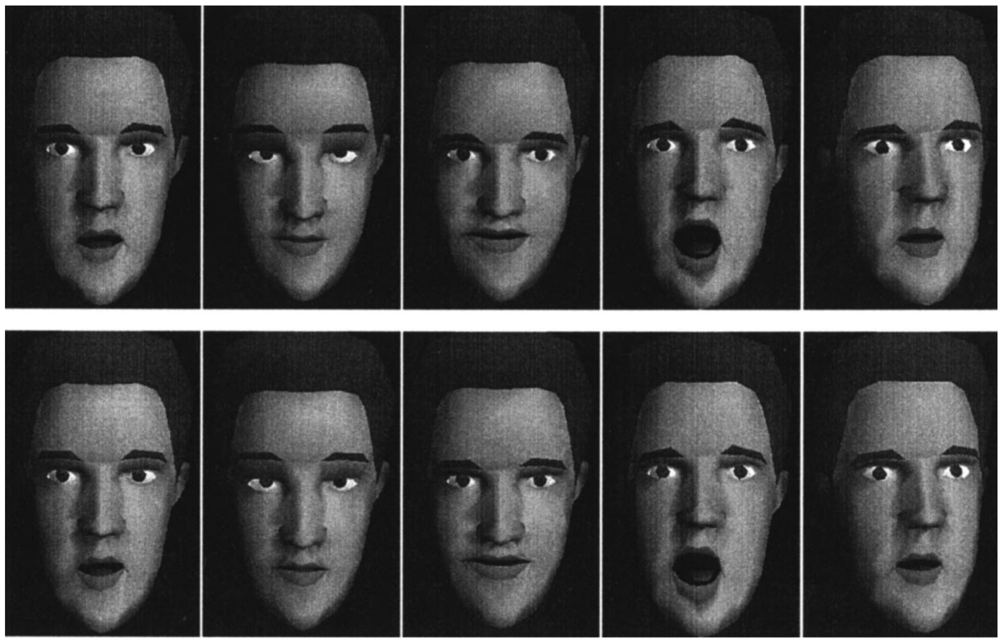


Fig. 18. DCT results of *Marco30* sequence (PSNR = 34.7 dB, bit rate = 0.4 kbit/s). Frames from the (a) original and (b) decoded sequence.

TABLE II
BIT RATE (bit/s) FOR *Marco30* SEQUENCE UNDER DIFFERENT PSNR

	PSNR	20 dB	30dB	35dB	45dB
Compression methods	PC	311	564	934	2191
	PCA+PC	243	284	738	-
	DCT	156	284	452	1211
	PCA+DCT	44	100	916.42	-

TABLE III
BIT RATE (bit/s) FOR *Emotions* SEQUENCE UNDER DIFFERENT PSNR

	PSNR	20 dB	30dB	35dB	45dB
Compression methods	PC	391	742	914	1742
	PCA+PC	245	248	251	305
	DCT	138	170	215	374
	PCA+DCT	48	56	70	117

TABLE IV
BIT RATE (bit/s) FOR *Aleta* SEQUENCE UNDER DIFFERENT PSNR

	PSNR	20 dB	30dB	35dB	40dB
Compression methods	PC	773	1454	1897	2375
	PCA+PC	273	556	779	1061
	DCT	249	899	1315	1765
	PCA+DCT	112	581	855	1434

rate of 0.4 kbit/s (or totally, 20 000 bits) with PSNR equal to 34.7 dB. In Fig. 19, the FAP `open_jaw` is plotted. The solid curve is for the original sequence, and the dotted curve is for the reconstructed sequence.

E. PCA + DCT Method

In Section VII-C, a hybrid FAP compression scheme using both PCA and PC methods is discussed. An alternative approach is to replace the PC module with the DCT scheme. In Figs. 14–16, results using this approach are shown. The curves are labeled "PCA + DCT." For all three sequences, this method achieved the highest compression ratio when the PSNR is around 30 dB. This indicates that strong temporal redundancy exists in principal component sequences. By combining PCA with a DCT module, correlation in both spatial and temporal domains is exploited. Because of the distortion introduced in PCA by discarding components and in DCT by quantizing transform coefficients, very high PSNR is difficult to achieve using this approach. However, satisfactory visual results are obtained. Tables II–IV summarize the experimental results.

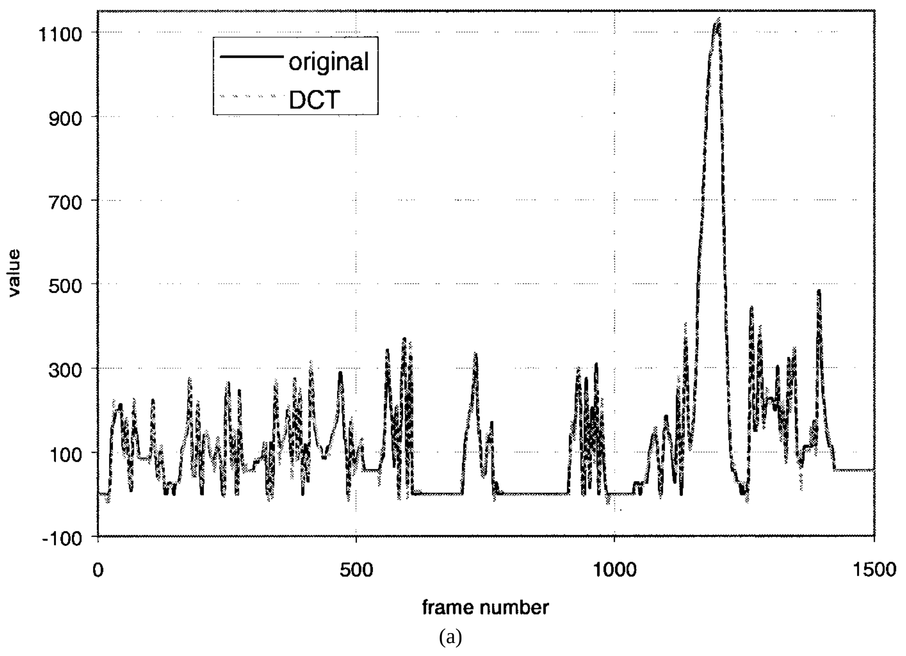
F. Discussion

Three major factors affecting the choice of FAP compression algorithms are real-time requirement, fidelity requirement, and FAP sampling rate.

In spatial FAP compression schemes, no coding latency is introduced. The same is true for the temporal predictive coding method. For applications with strong real-time requirements, combinations of these methods are favorable. In DCT, a delay of a single segment length, or 16 frames in our implementation, is introduced. The DCT method is appropriate for applications in which the real-time requirement is less crucial.

Experiments show that if the PSNR is above 30 dB, the animated face is almost free of major visual distortion. In Figs. 14–16, we observed that at this PSNR, DCT and PCA outperform the predictive coding method by a factor of 1.5 to 4 in compression. For a hybrid approach employing both PCA and DCT methods, an even higher compression is achieved.

When the frame rate is high, the DCT method is superior to the predictive coding method. On the other hand, if the frame rate of FAP is lower than 10 Hz, the predictive method performs better than the others. All the methods also work for



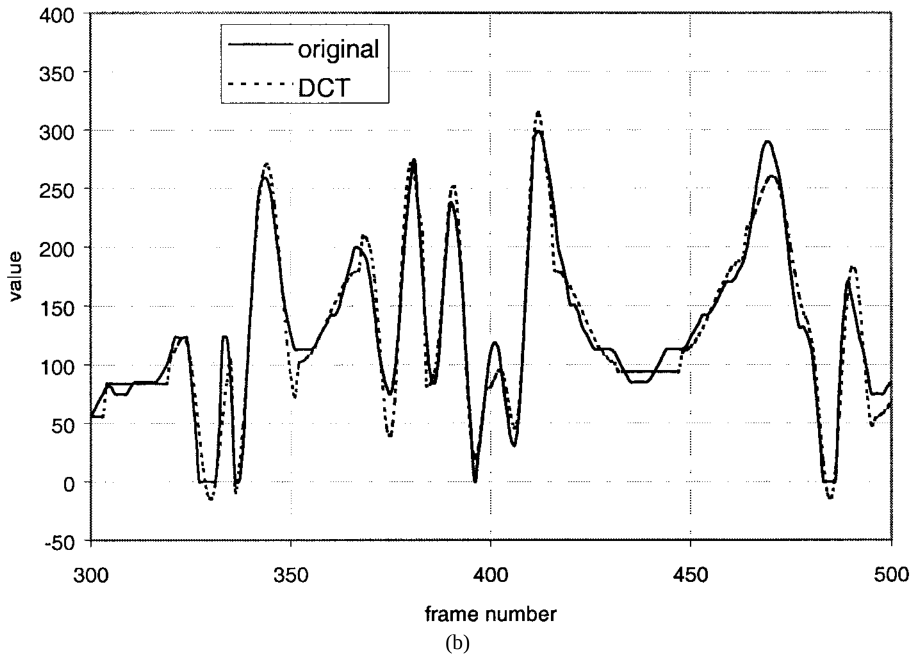


Fig. 19. FAP `open_jaw` in *Marco30* sequence (PSNR $= 34.7$ dB, bit rate $= 0.4$ kbit/s). (a) The sequence and (b) a portion of the sequence.

other types of animation parameters, such as body animation parameters and animal animation parameters.

VIII. Conclusions

Spatial and temporal compression methods, as well as a representation of interpolation for MPEG-4 facial animation parameters, have been investigated and compared. Experiments of these proposed methods on several test sequences are described, and their performances are analyzed. These methods, which are efficient for FAP compression, are independent of each other. These methods can be combined at the user's choice to meet the requirements of each particular application at hand.

Since the compression of the animation parameter is a new research topic, many unique problems are encountered and investigated. Unlike image or video data, animation parameters possess more object-specified properties. Our goal is to provide compression algorithms not only for the face object but also for other objects. The methods described here are general enough to achieve this goal.

References

[1] "Text for CD 14496-2 video," ISO/IEC JTC1/SC29/WG11 N1902, Nov. 1997.
[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. Circuits Syst. Video Technol.*, vol. 4, pp. 257–275, June 1994.
[3] T. Fukuhara and T. Murakami, "3-D motion estimation of human head for model-based image coding," *Proc. Inst. Elect. Eng.*, vol. 140, no. 1, pp. 26–35, Feb. 1993.
[4] K. Nagao and A. Takeuchi, "Speech dialogue with facial displays: Mul-

timodal human–computer conversation," in *Proc. 32nd Annu. Meeting Association for Computational Linguistics (ACL'94),* 1994, pp. 102–109.
[5] F. I. Parke and K. Waters, *Computer Facial Animation.* Wellesley, MA: Peters, 1996.
[6] N. M. Thalmann, H. T. Minh, M. de Angelis, and D. Thalmann, "Design, transformation and animation of human faces," *Visual Comput.,* vol. 5, no. 1/2, pp. 32–39, Mar. 1989.
[7] N. M. Thalmann and D. Thalmann, "Complex models for animating synthetic actors," *IEEE Comput. Graph. Appl. Mag.,* vol. 11, no. 5, pp. 32–44, Sept. 1991.
[8] P. Doenges, T. K. Capin, F. Lavagetto, J. Ostermann, I. S. Pandzic, and E. D. Petajan, "MPEG-4: Audio/video & synthetic graphics/audio for mixed media," *Signal Process.: Image Commun.,* vol. 9, no. 4, pp. 433–464, May 1997.
[9] R. Koenen, F. Pereira, and C. Chiariglione, "MPEG-4: Context and objectives," *Signal Process.: Image Commun.,* vol. 9, no. 4, pp. 295–304, May 1997.
[10] "MPEG-4 video verification model 2.0," ISO/IEC JTC1/SC29/WG11 N1260, Mar. 1996.
[11] P. Ekman and W. V. Friesen, *Facial Action Coding System.* Palo Alto, CA: Consulting Psychologists Press, 1977.
[12] P. Kalra, A. Mangili, N. M. Thalmann, and D. Thalmann, "Smile: A multilayered facial animation system," in *Proc. IFIP Conf. Modeling in Computer Graphics,* 1991, pp. 189–198.
[13] F. I. Parke, "Computer generated animation of faces," in *Proc. ACM Annual Conf.,* Aug. 1972, pp. 451–457.
[14] ——, "Parameterized models for facial animation—revisited," in *State of the Art in Facial Animation, ACM SIGGRAPH 89 Course Notes,* 1989, pp. 43–56.
[15] ——, "Parameterized models for facial animation," *IEEE Comput. Graph. Appl. Mag.,* vol. 2, no. 9, pp. 61–68, Nov. 1982.
[16] N. M. Thalmann, N. E. Primeau, and D. Thalmann, "Abstract muscle action procedures for human face animation," *Visual Comput.,* vol. 3, no. 5, pp. 290–297, Mar. 1988.
[17] Y. Lee, D. Terzopoulos, and K. Waters, "Realistic modeling for facial animation," in *Proc. SIGGRAPH'95,* 1995, pp. 55–62.
[18] S. M. Platt, "Animating facial expressions," in *Proc. SIGGRAPH'81,* Aug. 1981, pp. 245–252.
[19] D. Terzopoulos and K. Waters, "Analysis and synthesis of facial image sequences using physical and anatomical models," in *Proc. 3rd Int. Conf. Comput. Vision (ICCV'90),* Osaka, Japan, Dec. 1990, pp. 727–732.
[20] K. Waters, "A muscle model for animating three-dimensional facial expression," in *Proc. SIGGRAPH'87,* July 1987, pp. 17–24.
[21] 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.
[22] L. Williams, "Performance-driven facial animation," in *Proc. SIGGRAPH'90,* Aug. 1990, pp. 235–242.
[23] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," in *Proc. SIGGRAPH 98,* 1998, pp. 55–66.
[24] H. Li, P. Roivainen, and R. Forchheimer, "3-D motion estimation in model-based facial image coding," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 15, pp. 545–555, June 1993.
[25] I. Essa and A. Pentland, "Coding, analysis, interpretation and recognition of facial expressions," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 19, pp. 757–763, July 1997.
[26] 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.
[27] H. Tao, R. Lopez, and T. S. Huang, "Facial feature tracking under varying pose using Bayesian nets," in *Proc. Int. Workshop Coding Tech. for Very Low Bit-Rate Video (VLBV'97),* Linkoping, Sweden, July 1997, pp. 65–68.
[28] "Text for CD 14496-1 systems," ISO/IEC JTC1/SC29/WG11 N1901, Nov. 1997.
[29] "FAP interpolation table (FIT)," ISO/IEC JTC1/SC29/WG11 MPEG97/M2599, Aug. 1997.
[30] 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.
[31] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[32] K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[33] H. Tao, H. Chen, and T. S. Huang, "Analysis and compression of facial animation parameter set (FAP's)," in *Proc. IEEE Workshop Multimedia Signal Process (MMSP'97),* June 1997, pp. 245–250.

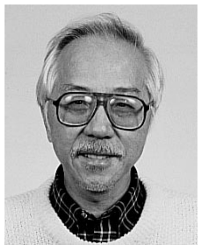
Hai Tao (S'94–M'99) received the B.S. and M.S. degrees in engineering from Tsinghua University, Beijing, China, in 1991 and 1993, respectively. He received the M.S. degree from Mississippi State University, Mississippi State, MO, in 1994 and the Ph.D. degree from the University of Illinois at Urbana-Champaign in 1998, both in electrical engineering.

Since November 1998, he has been a Member of Technical Staff in the Vision Technologies Laboratory, Sarnoff Corp., Princeton, NJ. His research interests cover a broad range in image and video processing, computer graphics, computer vision, and pattern recognition. He is the senior author of 20 technical papers and one book chapter. He has received three patents.

Homer H. Chen (S'83–M'86), for a photograph and biography, see this issue, p. 207.

Wei Wu received the B.S. degree from Shanghai Jiao Tong University, China, and the M.S. degree from the University of Washington, Seattle, both in electrical engineering.

Since 1994, he has been with the Signal Processing and Multimedia Systems Department, Rockwell Science Center, Thousand Oaks, CA, as a Member of Technical Staff. His areas of work include video compression (MPEG, H263), multimedia system integration, and image processing.



Thomas S. Huang (S'61–M'63–SM'76–F'79) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, China, and the M.S. and Sc.D. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge.

He was on the Faculty of the Department of Electrical Engineering at MIT from 1963 to 1973. He was on the Faculty of the School of Electrical Engineering and Director of the Laboratory for Information and Signal Processing at Purdue University, West Lafayette, IN, from 1973 to 1980. In 1980, he joined the University of Illinois at Urbana-Champaign, where he is now William L. Everitt Distinguished Professor of Electrical and Computer Engineering, a Research Professor in the Coordinated Science Laboratory, and Head of the Image Formation and Processing Group at the Beckman Institute for Advanced Science and Technology. During his sabbatical leaves, he has worked at the MIT Lincoln Laboratory, the IBM T. J. Watson Research Center, and the Rheinishes Landes Museum in Bonn, Germany. He has held Visiting Professor positions at the Swiss Institutes of Technology in Zurich and Lausanne, the University of Hannover, Germany, INRS-Telecommunications of the University of Quebec, Montreal, Canada, and the University of Tokyo, Japan. He has served as a Consultant to numerous industrial firms and government agencies both in the United States and abroad. His professional interests lie in the broad area of information technology, especially the transmission and processing of multidimensional signals. He has published 12 books and more than 300 papers in network theory, digital filtering, image processing, and computer vision. He is a Founding Editor of the *International Journal Computer Vision, Graphics, and Image Processing* and Editor of the Springer Series in Information Sciences, published by Springer Verlag.

Dr. Huang is a Fellow of the International Association of Pattern Recognition and the Optical Society of American. He has received a Guggenheim Fellowship, an A. V. Humboldt Foundation Senior U.S. Scientist Award, and a Fellowship from the Japan Association for the Promotion of Science. He received the IEEE Acoustics, Speech, and Signal Processing Society's Technical Achievement Award in 1987 and the Society Award in 1991.