Global Motion Estimation in Model-Based Image Coding by Tracking Three-Dimensional Contour Feature Points

Soo-Chang Pei, Senior Member, IEEE, Ching-Wen Ko, and Ming-Shing Su

Abstract—Recently a new type of video coding method called model-based image coding has attracted much attention as a potential candidate for low bit-rate visual communication services. This technique reconstructs the facial image with a preknown three-dimensional (3-D) human face model and its received model motion parameters. The parameters of the head motion are mainly divided into two parts: global motion parameters describe the rigid movement of the head, such as rotation and translation, and local motion parameters which deal with the nonrigid movements of facial expressions, such as the opening and closing of the mouth and eyes.

In this paper, we propose a new approach which can estimate the head global motion more robustly and accurately. Comparing with the existing techniques to match only a few key points, here we extract 3-D contour feature points and use chamfer distance matching to estimate head global motion. This can improve and enhance the contour tracking performance greatly.

We also develop another technique called facial normalization transform. It maps the facial region of the current input frame back to the normalized pose of the initial frame. Using this transform, we can analyze facial expressions at the same orientation and fixed region. This simplifies the analysis work a lot. Then, we do our encoding by the clip-and-paste method along with adaptive codebook technique.

In the following, the coder and decoder system are briefly described. Since we mainly focus the work on the analysis and synthesis of the facial portion images, background analysis and bitstream coding technique will not be discussed in this paper.

Index Terms—Contour feature points, generic facial model, global motion estimation, model-based coding.

I. INTRODUCTION

A PART from the information-theoretic coding methods, such as predictive coding, transform coding, vector quantization, etc., model-based coding introduces a new approach for very low bit-rate video communication services. Contrary to conventional coding methods, model-based coding expresses the information using the structural image model, in which it takes into account the three-dimensional (3-D) properties of the scene in some sense.

In this paper, we adopt a simple facial model using about 350 small triangles. This model does not include the complicated microstructures such as eyes, nose, and mouth; these structures are not necessary in the following analysis work. However, these microstructures can be added without changing the algorithms we discuss in this paper.

In [1]–[3], the global motion is estimated by explicit selection of a few key points on the facial image. These methods have a major problem that these key points may be occluded. Another problem is that it is somewhat difficult to extract these key points robustly and automatically. Instead of tracking only a few key points in the current methods, we proposed a new 3-D contour feature points matching technique to estimate the global motion parameters. These 3-D contour feature points are extracted automatically by an edge detector and geometrical two-view triangular measurements. We can estimate the global motion parameters very robustly and accurately by searching the best match between these 3-D contour feature points and the edge pixels in the input image.

In [4]–[6], the facial expressions are handled in each current input frame. They are all challenged to face the problem how to locate the expressive regions, such as eyes and mouth, properly. We propose a novel method, called facial normalization transform, which maps the facial region of the input frame back to the normalized pose of the initial frame. That is to say, we can do our analysis work of facial expressions at the same place regardless of what the global motion is and which coding techniques of facial expressions will be used. In order to demonstrate that facial normalization transform can map the facial region to almost the same fixed place, we implement our analysis work by the clip-and-paste methods that can clip the eyes and mouth region at the same position.

In the following, we describe the establishment of our facial model in Section II. Section III proposes the technique of 3-D contour feature points matching to estimate global motion parameters. In Section IV, we present the facial normalization transform to normalize the input facial image and auto-extract the specific expressive regions. Then Section V briefly describes the coding and decoding algorithms and shows the experimental results. Finally, conclusions are made in Section VI.

II. ESTABLISHMENT OF FACIAL MODEL

The 3-D facial model that we employ is very simple and composed of about 350 small triangles (Fig. 1). This model is mainly for the use of 3-D contour feature points tracking (discussed in Section III) and secondarily for the facial normalization transform (discussed in Section IV). Meanwhile its
unsMOOTH and complicated parts such as eyes, nose, and mouth are difficult to build and can be avoided. To establish our face model, a generic smooth model needs be prepared first.

A. Transformation of the Generic Facial Model with an Affine Mapping

We recast the generic model to fit the particular features of the face in the initial frame, this frame must be a frontal view of the talking person; however, it may not be the very first frame. The deformation is carried out by finding an affine mapping between the two-dimensional (2-D) projection of the generic model and the initial frame. The affine mapping, which maps the point \((x, y)\) to the point \((x', y')\), is expressed by

\[
\begin{align*}
x' &= ax + by + c \\
y' &= dx + ey + f.
\end{align*}
\]

This equation can perform enlargement, reduction, rotation, and translation movement by properly determining the values of six parameters from \(a\) to \(f\). Correspondence points between the model and the initial frame are interactively specified on the monitor screen. In total, six points are specified manually (Fig. 1). The coordinate values of these six points are substituted into the above equation to obtain a series of equations. By solving these equations, the affine mapping coefficients can be determined. The affine mapping with these coefficients is then applied to the \((x, y)\) coordinates of each vertex of the generic model.

This affine mapping is valid in the 2-D domain. Since only 2-D facial images are available, there are no reliable methods to get useful depth information. Fortunately, it does not vary much in proportion of head between \(x\) and \(y\) dimension and \(z\) dimension. Usually \(a, e > b, d\) we consider \(a, e \) parameters as main scalar factors along \(x, y\) dimension. So the scalar factor along the \(z\) dimension may be considered approximately as \(\sqrt{a^2 + e^2}/2\). We simply calculate the \(z'\) position of the vertex by multiplying the generic model depth \(z\) with the factor \(\sqrt{a^2 + e^2}/2\) [7]. If the model is facing almost front, this approximation is quite effective.

B. Determining the Texture Mapping of the Model

Texture mapping can enhance the visual richness of the raster scan images greatly, only requiring a small increase in the computational complexity. Since our adopted model does not include the microstructures, it is much easier to do texture mapping. We assume the curvature of the face is not large. So we can do our texture mapping by simply orthographically mapping the facial image of the initial frame onto the surface of the transformed model (Fig. 1). We take this transformed model with texture mapping as our facial model.

III. GLOBAL MOTION ESTIMATION

In this section, we will derive a direct and robust approach to estimate global motion parameters from two consecutive time frames. There are six degrees of freedom needed for rotation and translation of the model. Assume a point \(p\) on the model is represented by a vector \(s = (x, y, z)^T\), the change in position of this point can be formulated as

\[
s' = Rs + T
\]

where \(R\) is a \(3 \times 3\) rotation matrix and \(T\) is a 3-D translation vector. \(R\) can be represented as a product of three matrices, each corresponding to rotation about one axis. Then

\[
R = R_x \cdot R_y \cdot R_z
\]

where

\[
R_x = \begin{bmatrix} \cos rx & \sin rx & 0 \\ -\sin rx & \cos rx & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & rx & 0 \\ -rx & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
R_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos ry & \sin ry \\ 0 & -\sin ry & \cos ry \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & ry \\ 0 & 1 & 0 \\ -ry & 0 & 1 \end{bmatrix}
\]

\[
R_z = \begin{bmatrix} \cos rz & 0 & \sin rz \\ 0 & 1 & 0 \\ -\sin rz & 0 & \cos rz \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & rz \\ 0 & 1 & 0 \\ rz & 0 & 1 \end{bmatrix}
\]

Assume infinitesimal rotation takes place during the two frames, then we use approximations \(\cos rx \approx 1, \sin rx \approx rx\). \(R\) can be approximately represented as [8]

\[
R \approx \begin{bmatrix} 1 & -rx & ry \\ rz & 1 & -rx \\ -ry & rz & 1 \end{bmatrix}.
\]

Therefore

\[
\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & -rx & ry \\ rz & 1 & -rx \\ -ry & rz & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix}.
\]
Assuming a parallel projection from 3-D space to 2-D plane, then the 2-D coordinates on the 2-D image plane before and after the motion are \((x, y)\) and \((x', y')\), respectively, therefore

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} 1 & -rx & ry \\ rz & 1 & -rx \end{bmatrix} \begin{bmatrix} x \\
y \\
z
\end{bmatrix} + \begin{bmatrix} tx \\
ty
\end{bmatrix}. \quad (9)
\]

Then, the global motion parameters we need to estimate are five parameters in total: \(rx, ry, rz, tx,\) and \(ty\).

For processing the global motion estimation, we must extract the 3-D contour feature points (3-D CFP’s) from both the initial frame and the facial model first. And then we can track these 3-D CFP’s in the following frames to estimate the global motion parameters.

### A. Extracting 3-D CFP’s from the Initial Frame

The 3-D CFP’s are the set of points that represent the contour of the 3-D facial model. We will track them later by using the chamfer distance map to estimate the global motion parameters. This preprocess is shown in Fig. 2 and described as follows.

1) **Specifying the Face Region and Finding the Coordinates \((x, y)\) of 3-D CFP’s:** The 3-D facial model is projected onto a 2-D plane and segmented to two classes. The brighter part is regarded as the face region. By labeling the first or last pixel that changes from black to white or vice versa in each horizontal line of the segmented image [see Fig. 3(a)], the outline points of the face region are extracted as the candidate CFP’s [see Fig. 3(b)].

2) **Finding the Depth of 3-D Candidate CFP’s:** Although we can get the \(z\) coordinate of each candidate CFP by directly interpolating the 3-D wireframe model data from the known \(x, y\) coordinates, if the \((x, y)\) coordinates of CFP’s fall in two or more triangles of the model, the interpolation method will cause 3-D CFP correspondence ambiguity to determine the depth \(z\). This case happens very often, especially for the points at or near the edge of the model. By computer vision approach, we calculate them easily by a simple two-view triangular measurement made with a preset rotation.

To solve the correspondence ambiguity problem and find the depth of 3-D CFP’s, by two-view measurements we rotate the facial model with a small angle about the \(y\) axis, say \(\delta ry\). We will obtain the new coordinates \((x', y')\) in the 2-D projection plane by Step 1) again. The relationship between the original coordinate \((x, y)\) and the new one \((x', y')\) is given as follows:

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta ry \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\
y \\
z
\end{bmatrix}. \quad (10)
\]

or

\[
x' = x + \delta ry \cdot z
\]

\[
y' = y
\]

and solve \(z\), given \(x', y', x, y, \delta ry\), so

\[
z = (x' - x) / \delta ry. \quad (12)
\]
3) Finding the Valid 3-D CFP’s: The coordinates \((x,y,z)\) obtained in Step 2) are the candidate CFP’s. They contain some nonvalid CFP’s due to the occluded edge of this model. We first project these candidate contour feature points onto the 2-D plane and then check if they match the edges of the original image. If the point matches, we preserve it as the valid 3-D CFP. Otherwise, it is a nonvalid CFP and we delete this feature point. For a typical facial model, say Miss American, there are about 150 valid 3-D CFP’s (Fig. 3). We will track these valid 3-D CFP’s in our matching algorithm to estimate the global motion parameters.

B. Two-Threshold Normalized Edge Detection

Before applying our matching algorithm to the following image frames, the edge detection is preprocessed first.

We first use the Sobel operator on the input image. The output of the Sobel operator is normalized by the original corresponding pixel value. Furthermore, we exploit the local histogram of 16\(\times\)16 blocks. The 15% pixels in a block with the greatest gradient values are considered as the real edge points. The details of the edge detection are explained in the following and shown in Fig. 4.

1) First, 5\(\times\)5 median filtering is applied to the input image to suppress noise.

2) The filtered image after Step 1) passes into the Sobel operator.

3) The resultant edge image obtained from Step 2) is normalized by the original corresponding pixel values.

Then \(d(i,j) = s(i,j)/x(i,j)\):

- \(x(i,j)\): pixel values of the original image.
- \(s(i,j)\): pixel values after the Sobel operator.
- \(d(i,j)\): normalized gradients of the edge image.

4) The normalized gradients of pixels are then thresholded by using some predetermined constant

\[ t(i,j) = \begin{cases} 0, & \text{if } d(i,j) < \text{threshold 1} \\ d(i,j), & \text{otherwise.} \end{cases} \]

\(t(i,j)\): candidate pixel value of the edge.

In our experiment, the threshold 1 we choose is 0.2.

5) Finally, for each block, the local histogram statistics are calculated over those selected edge pixels after the first thresholding in Step 4). Define it as \(F\{x\} = \text{prob}(x \leq x)\) then

\[ Y(i,j) = \begin{cases} 1, & \text{if } F\{t(i,j)\} > \text{threshold 2} \\ 0, & \text{otherwise.} \end{cases} \]

In our experiment, the block size is 16\(\times\)16 and the threshold 2 is 0.85.

The algorithm of Sobel edge detection cannot find some unobvious edges for a too small gradient and generate some thick edges for a large gradient (Fig. 5). It is not suitable for matching process because some unobvious edges, especially for skin colors, contain very important features, also thick edges cause some matching ambiguity.
Our edge detection algorithm involves two thresholds. For the first threshold in Step 4), it classifies the possible candidate pixels to become the edge. If \( d(i, j) \), the normalized gradient, is too small, \( \theta(i, j) \), the candidate edge pixel, is set to zero and becomes impossible to be the edge; otherwise, it keeps the original value. In the second threshold, among those selected edge candidates after the first threshold, we select the top 15% of pixels with the greatest gradient values within each 16 x 16 block as the real edge points of the input image.

Our algorithm will generate thin edges for both unobvious and large gradient edges (Fig. 5) because we only select the top 15% for each local block as the edge. Meanwhile, it will not generate the small gradient edges in flat block because we have thresholded them out in Step 4).

C. Matching Algorithm Using Chamfer Distance Transform

Our goal is to find the optimum global motion parameters \((rx, ry, rz, tx, ty)\) from the search space. Then, the projection of the 3-D CFP’s under this parameter set transformation are expected to fall on the edges of the 2-D image or within some tolerant errors. The goodness of this matching algorithm can be checked by the distance map. In the distance map image, each nonedge pixel is given a value that is a measure of the distance to the nearest edge pixel. The edge pixels will get the distance value zero. The chamfer distance [9] is applied here, since it costs less computation and is a reasonably good approximation to the Euclidean distance (Fig. 5). The optimum parameter set is the one that can minimize the sum of chamfer distance values corresponding to the projection of the transformed 3-D CFP’s. We will continue to track these 3-D CFP’s by checking the chamfer distance sum of each input image to estimate the global motion parameters in each frame (Fig. 6). The outline of the matching algorithm is shown in Fig. 7 and described as follows.

1) The edge detection and the distance transform are applied on the current input image.
2) Select one initial parameter set \((rx, ry, rz, tx, ty)\) in the search space.
3) Rotate and shift the total 3-D CFP’s by this parameter set and then project these transformed 3-D CFP’s to 2-D plane.
4) Sum the chamfer distance values corresponding to the projection of transformed 3-D CFP’s; select the parameters set, whose sum of the corresponding chamfer distance values is the smallest, as the optimum one.
5) Continue to process the next input image.

D. Search Space for 3-D Contour Matching

For a typical image sequence, we can choose our search space as follows.

1) Rotation parameters: \(rx, ry, rz\): vary from \(-0.05\) rad to \(+0.05\) rad with step size \(0.01\) rad and have 11 elements in each parameter space.
2) Translation parameters: \(tx, ty\): vary from \(-12\) pixel to \(+12\) pixel with step size 1 pixel, and have 25 elements in each parameter space.

The search in \(rx, ry, rz, tx, ty\) involves five dimension’s spaces and has 831,875 \((= 11^3 \cdot 25^2)\) elements in total for the “full” search scheme. In our experiments, we apply a one-dimensional (1-D) full search on each dimension to find the optimum rotation parameters first. That is, we set \(rx, ry\) to be zero first, and then use full search to find optimum \(rz\); \(rz, ry\) are then found with the similar process. This matching algorithm works fine and tracks the contour very well. After finding the optimum rotation parameters, we use two-step full search to find the translation parameters. Total computation loads for each input image in an image sequence require 4950 (=150 CFP’s \times 11\) step sizes/axis rotation \times 3 axis rotations) \(4 \times 4\) matrix multiplications and several additions to find the optimum global motion parameters if we ignore the computation loads for the edge detection and the distance transform. These operators can be speeded up by implementing them in parallel on some special hardware graphic processors.

E. The Discussions of the Approximation Model in Matching Algorithm

It is not practical in real video communication services to ask the user to input the exact depth information, \(z\) coordinate, or ask the user to buy a very expensive 3-D scanner. In
our system, we calculate the $z$ coordinate of the model approximately as described in Section II-A. However, it costs some estimation errors, especially when the model is rotating.

That means that if we only use a few key points to track and estimate the global motion, it is very easy to lose the tracking due to the model drift. However, we adopt about 150 CFP’s to minimize this effect. Although there exist some errors in these CFPs’ position, it should still get the minimum chamfer distance values in the proper global motion if the symmetric property of error is assumed.

However, there may still exist errors between the estimation results and the real head movement. But, in our algorithm, the translation motion will compensate the small rotation motion estimation error, caused by the depth error. In all, by our match criteria, if we rotate and translate this facial model with the estimated parameters, the contour of the model will match the contour of the face of the image properly. As we know, the whole face area is enclosed by this contour. That is to say, the facial image, regenerated by this model and these parameters, will match the original facial image very properly.

In our experiment, we succeed in tracking the Miss America image sequences from the first frame to the last one, total 140 frames. When we reconstruct the facial image, since we do not encode the background, we directly put the generated facial image over the original input image. These facial images almost perfectly match with the background. And, due to the eyes and mouth also falling within this contour, we can locate their positions accurately by calculating their relative positions on the model. In the next section, we propose a new technique, called facial normalization transform, to map the facial region including the eyes and mouth into the fixed place.

### IV. FACIAL EXPRESSION CODING

There are many articles to deal with coding the facial expressions. No matter what methods they use, they confront with a problem—how to locate the eyes and mouth properly? In this section, we mainly propose another technique, called facial normalization transform, which can map the face region, including eyes and mouth, into the fixed place. With this transform, the analysis work for facial expressions becomes much easier.

After transform, we have a lots of approaches to analyze and encode the facial expressions [10]. For example, we can encode the openness of the eyes and mouth by a horizontal edge detector; or we can adopt the parabolic curves to represent the shapes of the eyes and mouth. For the sake of simplicity, we adopt the clip-and-paste methods along with the adaptive codebook techniques (see Fig. 10).

#### A. Facial Normalization Transform

In the previous section, we obtained the global motion parameters $(rx, ry, rz, tx, ty)$. The original head position in the initial facial image can be transformed to the current head position in the $i$th facial image with the estimated global parameters. We label this transform as $I(rx, ry, rz, tx, ty)$. We may consider the inverse transform $I^{-1}(rx, ry, rz, tx, ty)$ as the normalized transformation, which can transform the current head position in the $i$th facial image to the normalized head position in the initial facial image. The current input facial image, say the $i$th facial image, can be normalized by $I^{-1}$ operator with the estimation global motion parameters $(rx, ry, rz, tx, ty)$ in previous section. Now we describe $I^{-1}$ operator, facial normalization transform, as follows.
Fig. 8. (a) The image after facial normalization transform. (b), (c) Codebooks of the eyes and mouth patterns. The extracted expressive regions are marked by white boxes. We extract these patterns at the same position of the normalized facial images to establish the codebooks.

1) Rotate and shift the facial model with the global motion parameters \((rx, ry, r\tau, tx, ty)\) to the current \(i\)th head position.
2) Project the current \(i\)th facial image onto the surface of the transformed facial model. It can be considered as a texture mapping process.
3) Inverse rotate and shift this 3-D facial model with texture mapping back to the initial normalized head position.
4) Project this facial model to 2-D plane. This is the normalized facial image. (Fig. 8).

B. Clip-and-Paste Method and Adaptive Codebook

By facial normalization transform, not only can we get the eyes and mouth patterns at the same place of the normalized facial image, also the patterns are similar and their sizes are the same. We use the simplest mean square error and threshold method to classify the extracted patterns. Initially, we put the eye or mouth patterns in the original image into the empty codebook. We calculate the square difference errors between the extracted pattern and every pattern in the codebook. If all the square difference errors are greater than a threshold, the eye or mouth shape is considered to be sufficiently different from all the code patterns in the codebook. We add it into the codebook as a new code pattern and send the information to the receiver. Otherwise, we find the most matched pattern, i.e., the minimum square difference errors, and only send the code pattern index number to the receiver.

Using the clip-and-paste method along with the adaptive codebook technique is intuitive and simple, but it suffers a main drawback—artifacts along the boundary of the eyes and mouth boxes (See Fig. 9). We can reduce this effect by adding one or two parameters for color correction and using a small smooth filter on the boundary of the boxes. We mainly focus our work on the facial normalization transform, so we do not address them more deeply here. In fact, we can implement any efficient techniques for facial expression coding along with the facial normalization transform we propose.

In our experiments, there are only 12 eyes patterns and 20 mouths patterns in our codebook for 140 facial images of the Miss American video sequence (Fig. 8). The synthesized image sequence works very well and smoothly.

V. CODING AND DECODING ALGORITHMS

The overall steps for coding and decoding algorithms are shown in Figs. 2, 4, 7, 10, and 11 and summarized as below.

A. Build 3-D Head Model (Fig. 2)
1) Prepare a generic 3-D wireframe head model.
2) Specify six feature points for affine mapping to the initial frame.
Fig. 10. Facial expression coding by extracting patterns from normalized facial image: (a) the normalized facial image by inverting the global motion and (b) extract the eyes and mouth patterns to establish the adaptive codebook.

Fig. 11. Synthesize the image sequences at receiver end: (a) initial frame image, (b) put the desired patterns of the current frame over the initial image, (c) rotate the facial model with the received global motion parameters, and (d) project the facial model to a 2-D plane.

B. Coding of the Initial Frame (Fig. 2)

1) Determine the face region and extract six feature points in the first frame; this step is done manually.
2) Based on the correspondence between the six feature points in the input frame and the generic model, determine an affine mapping between the initial frame and the model.
3) Use the affine mapping found in step 2) to transform the generic model to fit with the initial frame.
4) Texture map the face image in the initial frame onto the transformed model by an orthographic projection. This transformed model with texture mapping is referred to as the facial model.
5) Code the texture map and the affine parameters.
6) Store the eyes and mouth patterns in the boxes as the first entries of the codebook for the eyes and mouth.
7) Extract valid 3-D CFP’s for the facial model from the initial frame; this is accomplished by retaining only those candidate CFP’s that match the edges of the initial frame.

C. Coding of the Following Frames (Figs. 4, 7, and 10)

1) Determine the global motion of the current input frame from the transformed model by finding a set of five motion parameters that best match the CFP’s in the model to the two-threshold edge pixels (Fig. 4) in the current input frame (Fig. 7).
2) Inverse map the input frame back to the normalized pose of the initial frame using the global motion parameters found in Step 1) (Fig. 10).
3) Extract the eye and mouth boxes in the normalized image and compare them to the patterns stored in the codebook (Fig. 10). If no sufficient matches are found, they are stored as new patterns in the codebook.
4) Send the motion parameters, codebook index number, or new pattern to the receiver.
Fig. 12. (a) The original image of the fiftieth frame. (b) The synthesized image of the fiftieth frame. (c) The image after postprocessing to remove the dark stripes around the head. The region enclosed by the dark stripes is the facial region we analyze and reconstruct. They are easily removed by applying a medium filter on them.

D. Decoding and Synthesizing the Image Sequences (Fig. 11)

1) Get the corresponding eye and mouth patterns from the codebook by the received index number or new pattern.
2) Put the current pattern over the initial image at the corresponding position (see Figs. 11 and 9). This facial image will be regarded as the texture mapping image data.
3) Map this texture image data onto the 3-D facial model. Then rotate the facial model with the received global motion parameters.
4) Project this rotated facial model to a 2-D plane. This is the synthesized facial portion image (see Figs. 12 and 13). We put this synthesized facial portion image over the original current input frame as the resultant image.

In our proposed system, we do not model the shoulder, hair, and background of the images. These regions can be compressed very efficiently by the current image coding techniques.

VI. CONCLUSIONS

In this paper, we have proposed two efficient techniques: 3-D contour feature points tracking and facial normalization.
transform. Besides, we also use a two-threshold normalization edge detector to select proper edge points for matching.

With 3-D CFP’s tracking, we can estimate the global motion parameters much more robustly and accurately. Not only because it is hard for all the CFP’s to be occluded, but also because this algorithm can endure the ambiguity in the depth information of the model. This is based on the fact that we have used a lots of CFP’s (150 CFP’s) to reduce the errors and the fact that the face region is enclosed by the contour.

With facial normalization transform, we can map the facial region to the fixed place. In this paper, we encode the facial expressions with clip-and-paste methods along with adaptive codebook techniques. We can avoid the complicated microstructures of the facial model such as eyelid, teeth, mouth, etc. This is much more efficient and robust than the current Facial Action Coding Systems (FACS) [11], [12]. It requires sophisticated facial expression modeling by action units (AU) and reliable methods for analyzing the facial parameters.

A very simple 3-D head model [see Fig. 1(a)] is used and is sufficient enough for our system. The present coder only deals with the analysis and synthesis of the facial portion images, we have not modeled and processed the shoulder, hair, and background of the image; these regions can be compressed and reconstructed by the current image coding techniques very effectively.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for the constructive comments and suggestions.

REFERENCES


Ching-Wen Ko received the B.S. degree from the Department of Communication Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1991 and the M.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, in 1993.

He currently works for Acer Netxus Inc., Hsinchu, as a Software Engineer. His research interests are in high-speed wide and local-area networks.

Ming-Shing Su was born on July 27, 1971. He received the B.S. and M.S. degrees in 1993 and 1995 from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

Since 1995, he has been a Research Assistant at the Institute of Information Science, Academia Sinica, Taipei. His research interests include image analysis, signal processing, and computer graphics.