Real-Time 3D Head Motion Tracking Using a Camera[1]

Chien-Feng Huang, Tzong-Jer Yang, Ming Ouhyoung

Communication and Multimedia Laboratory
Dept. of Computer Science and Information Engineering
National Taiwan University
1, Roosevelt Rd., Sec. 4, Taipei, 106, Taiwan
Tel: +886-2362-5336 Ext. 421
{ cardy, tjyang, ming}@cmlab.csie.ntu.edu.tw

Abstract

In this paper, a real-time 3D head motion tracking technique is presented. Two issues are addressed: feature tracking, and 3D head motion estimation. In feature tracking, three facial features, two eyes and one nostril, are tracked in a live video sequence under regular office lighting condition. In 3D head motion estimation, knowledge about human head motion is involved in developing a cost function that is then minimized iteratively to find a set of 3D motion parameters. The involvement of head motion knowledge is important, since most optimization techniques are prone to be trapped into incorrect local optimal solutions, if only three points correspondence is given. The recovered 3D motion includes rotations about and translations along three axes. A model-based visual communication system that applies the proposed real-time 3D head motion tracking technique is also illustrated, where a remote 3D head model moves according to a real person's head movement in 3D space. The system performance is above 25 frames/sec on a regular PC with a Pentium-II 400 MHz CPU, with 3D head motion tracking and texture-mapped head model rendering involved.

Keywords

3D head motion estimation, feature tracking, model-based coding.

1. Introduction

The goal of 3D head motion tracking is to recover one's head transformation in 3D space. The transformation includes rotations about and translations along the three axes. In general, there are two different types of approaches: optical flow based tracking and feature based tracking. In this paper, since our focus is on developing real-time systems, we adopt the approach of feature based tracking that has lower computation complexity. Two typical issues are involved: feature tracking and 3D motion estimation.

The real-time constraint is a great challenge for face oriented researches. A good feature tracking method often requires reconstruction or understanding about the 3D geometry behind the scene. For example, epipolar geometry provides a good mean to refine tracking accuracy. However, optical flow based 3D reconstruction processes usually involve expensive computation, and are thus improper for our purposes.

For feature tracking, or image matching equivalently, cross-correlation is often introduced as a measure of similarity between two patterns. A positive and higher correlation indicates that the two patterns are similar in 3D space. However, just one single metric for matching is insufficient for practical applications, and cross-correlation is known to be sensitive to intensity changes [1]. In recent researches, multiple cues from faces are usually combined together to form a set of matching constraints. Useful cues from faces include the geometry relationships of eyes and mouth [2-4], head shape [5-10], face color [11-16], and so on.

When features are associated in a pair, 3D motion can be inferred using either analytical methods or optimization tools. A good review has been presented in [17].

Among different researches, an impressing approach is to include a high-level 3D head model into motion tracking process [5,8,10]. The demonstrations show that a rough 3D head model is very helpful in providing stable and accurate tracking results. Nevertheless, how to produce and calibrate the 3D head model for general public, and how to speed up the large amount of computations are two major issues encountered.

As a result, we propose a real-time motion estimation method that infers one's 3D head motion with only three facial features. The three facial features are two eyes and one nostril. Feature correspondence between two consecutive video frames is automatically established through the proposed feature tracking method, except that the initial locations of the three features are given manually.

This paper is organized as the following. In Section 2, the proposed feature tracking method is first introduced, and an iterative 3D motion estimation method based on human motion knowledge is illustrated in Section 3. A model-based visual communication system adopting the two proposed methods is presented in Section 4. Results and performances are provided in Section 5, and we conclude the paper in Section 6.

[1] Executable programs are available at http://www.cmlab.csie.ntu.edu.tw/~tjyang/research/face.html.

2. Multi-modal Feature Tracking

As mentioned in [18], image matching is an ill-posed problem where no guarantee is given that a solution exists or is unique. In addition, faces appear highly deformable, which make feature tracking even harder. Optical flow may provide good motion information because large amount of flow is generated to suppress noise and incorrect estimations. However, it's difficult to achieve the real-time goal, even only part of selected pixels are calculated [10]. We therefore settle to feature based approaches.

In general, feature tracking also involves facial feature extraction. In this paper, however, we do not address this issue, and initial feature identification is accomplished with user's assistance. Three facial features are selected for their rigidity. These three features are two eyes and one nostril. The initialization is quite simple, where a subject only has to face to the camera and pick the three features manually using a mouse. The identified features are automatically adjusted to nearby pixels with high gradient values.

During feature tracking, three metrics are evaluated in order. The three metrics are *point gradient difference*, *region color intensity difference*, and *normalized cross-correlation*. In the point gradient difference metric, a candidate is considered to have higher probability to be a correct matching if its gradient is similar to that of a feature to be matched. In the region color intensity difference metric, pixel-by-pixel color differences are accumulated, with smaller candidates given higher probability. The last step is to perform normalized cross-correlation over every pairs of matching, and the one with a larger correlation coefficient is considered as a correct matching. In each of the three steps, candidates are sorted according to their corresponding probabilities.

As mentioned in [16], its the author considers the face tracking problem as a search reduction problem. In the above procedure, we are actually performing a search space reduction process, where in each step, possible candidates are re-evaluated and sorted, and only those ones with acceptable high probabilities enter the next cycle.

An alternative metric replacing the normalized cross-correlation is the pixel-by-pixel difference for two binary patterns, where each binary pattern is generated by thresholding gradient values. Gradient computation provides a more stable tracking result, when compared with gray-level cross-correlation that is more sensitive to illumination changes [1]. At this moment, these two metrics are both applied in our system.

3. Iterative 3D Head Motion Estimation with Three Points

In the analysis of [17], in the case of 2D-to-2D correspondences, at least five pairs of points have to be offered to resolve 3D motion. However, as it is also known that a face is less textured in most area, and is deformable. These properties make it difficult to select more than five rigid points on a face. Fortunately, if we have a head model in the 3D space, this problem is reduced to the case of 2D-to-3D correspondence, where the minimal requirement to obtain at least one solution is three points correspondence.

To obtain a 3D head model for motion estimation, we use a simple 3D triangle formed by the three selected features [21], and extend the work with marker-free feature tracking, error handling, and more efficient iteration steps. In this work, a simple 3D head motion tracking method is developed that applies a gradient decent like method to find a set of transformation parameters with six degrees of freedom. Four criteria are developed to guide an objective method toward a solution with minimal errors. Domain knowledge about human head is exploited to develop these four criterions.

This method does not intend to obtain *precise* rotation angles and translation offsets, but to obtain *approximate* ones but with smooth trajectory to avoid jittering effects. This is reasonable, since in visual communication, one needs not to know exactly how many degrees that another person rotates, but to see a natural head movement.

In short, three issues are included in the proposed head motion estimation method, and each issue is described in the following sections:

1. *3D model initialization*;
2. *3D head motion estimation*;
3. *Error Handling*.

3.1 3D Model Initialization

The uniqueness of a feature triangle's 3D position and orientation cannot be guaranteed even when additional information, the triangle's side lengths, is given. In our approach, we assume that the subject's head is *parallel* to the camera's 2D projection plane, and therefore we can uniquely determine the 3D position of the feature triangle by restricting the three vertices of the 3D feature triangle owning the same depth value z. In Figure 1, assuming that the depth z of the 3D feature triangle is of the same for different situations, and the side length l of the 3D feature triangle are also the same for different persons, the camera constant f is thus determined by $f = \frac{L}{l} z$, where L is the measured side length on the 2D projection plane. From our experiments, these assumptions doesn't cause problems for motion estimation. Once the camera constant f is determined, positions of the three vertices of the 3D feature triangle are thus determined according using inverse perspective projection: $x = z\frac{X}{f}$, $y = z\frac{Y}{f}$, where (X, Y) is a 2D known point, and (x, y, z) is its corresponding point in 3D space. In this way, we can obtain a 3D feature triangle easily.

3.2 3D Head Motion Estimation

Human's head motion has certain characteristics:

1. Positions of eyes and the nose are fixed relative to the whole head, so these three features can be considered to satisfy the rigidity constraint.
2. Hhead motions are rotation dominated.
3. Rotation pivot of a head is near to the center of the neck.

Characteristic 1 determines the three facial features for motion estimation, as mentioned above. Characteristic 2 suggests

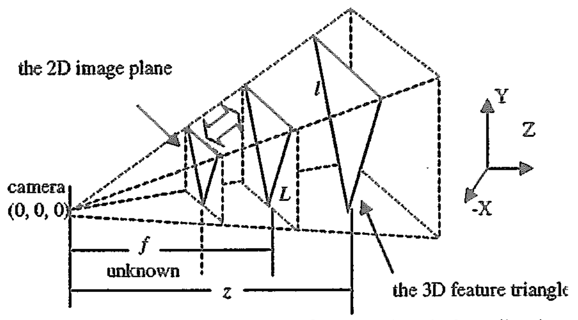

Figure 1. Initialization of the 3D feature triangle by adjusting the distance of the 2D image plane, or .the camera constant *f*.

us to give rotation parameters higher weighting factors, and Characteristic 3 is useful in guiding the iteration process along a correct minimization path. A rotation ambiguity problem can be resolved with a correct rotation pivot. The rotation ambiguity problem can occur because an incorrect rotation pivot is given. For example, two rotations about *x*-axis with rotation angles +θ and -θ will generate the same perspectively projected triangle on a plane. To resolve the problem, the head's rotation pivot is set to be at the center of the neck, and thus a rotation about *x*-axis will produce an offset along *y*-axis, which provides a cue in distinguishing the rotation ambiguities.

To estimation one's 3D head motion, we developed an objective function that measures the distance between the projected 2D positions of the vertices of the 3D feature triangle and the measured positions of the tracked facial features.

Positions of the three triangle vertices p_i^k at time instant *k*, $1 \le i \le 3$, can be computed by applying a transformation on the same three triangle vertices p_i^0 at the beginning:

$$p_i^k = Rp_i^0 + T \tag{1}$$

The translation vector *T* is a 3x1 vector defined by $(t_x, t_y, t_z)^t$. The rotation matrix *R* is represented in terms of Euler angles, where a general rotation is described as a sequence of rotations about three mutually orthogonal coordinate axes that are *x*-, *y*-, and *z*-axes. Let θ_i represents the angle of rotation around the *x*-axis, ψ_i represents the angle of rotation around the *y*-axis, and ϕ_i represents the angle of rotation around the *z*-axis. We have the rotation matrix given by

$$R(\theta_i, \psi_i, \phi_i) = R(\phi_i)R(\psi_i)R(\theta_i), \tag{2}$$

Notice that in equation (1), all rotations are performed on the very first 3D feature triangle, i.e., the one without any transformation. In this way, the recovered rotation parameters represent the amount of rotation angles from the original position.

Four metrics are developed to measure the distance between $P_M^k = (P_{M,1}^k, P_{M,2}^k, P_M^k)$ and $P_A^k = (P_{A,1}^k, P_{A,2}^k, P_{A,3}^k)$, where $P_M^k = (X_M, Y_M)$, and $P_A^k = (X_A, Y_A)$. Note that P_M^k is the 2D projection of the three 3D vertices p_1^k, p_2^k, p_3^k on the 3D feature triangle, and $P_{A,1}^k, P_{A,2}^k, P_{A,3}^k$ are the three tracked feature points on the 2D projection plane. In addition, we represent the recovered 3D rotation angles as $\pi^k = (\theta, \psi, \phi)$ at time instant *k*, and the recovered 3D translation offsets as $\delta^k = (t_x, t_y, t_z)$. The metrics can therefore be defined in terms of p^{k-1}, π^k, δ^k, and P_A^k. In the following, the superscript *k* on the right hand side is omitted for simplicity:

$$e_1(p^{k-1}, \pi^k, \delta^k, P_A^k) = {}_{i=1}^{3} \left((X_{M,i} - X_{A,i})^2 + (Y_{M,i} - Y_{A,i})^2 \right) \tag{3}$$

$$e_2(p^{k-1}, \pi^k, P_A^k) = {}_{i=1}^{3} \left(((X_{M,i} - X_{M,C}) - (X_{A,i} - X_{A,C}))^2 + ((Y_{M,i} - Y_{M,C}) - (Y_{A,i} - Y_{A,C}))^2 \right)$$

$$\text{where } \begin{matrix} X_{M,C} \\ Y_{M,C} \end{matrix} = \frac{1}{3} {}_{i=1}^{3} \begin{matrix} X_{M,i} \\ Y_{M,i} \end{matrix}, \quad \begin{matrix} X_{A,C} \\ Y_{A,C} \end{matrix} = \frac{1}{3} {}_{i=1}^{3} \begin{matrix} X_{A,i} \\ Y_{A,i} \end{matrix} \tag{4}$$

$$e_3(p^{k-1}, \pi^k, P_A^k) = {}_{i,j,k} \left(r(P_{M,i}, P_{M,j}, P_{M,k}) - r(P_{A,i}, P_{A,j}, P_{A,k}) \right)^2,$$

$$\text{where } j = (i \bmod 3) + 1,\ k = (i \bmod 3) + 2,\ 1 \le i$$

$$r(P_i, P_j, P_k) = \frac{(X_i - X_j)^2 + (Y_i - Y_j)^2}{(X_j - X_k)^2 + (Y_j - Y_k)^2} \tag{5}$$

$$e_4(p^{k-1}, \pi^k, P_A^k) = {}_{i,j} \left(s(P_{M,i}, P_{M,j}) - s(P_{A,i}, P_{A,j}) \right)^2,$$

$$\text{where } j = (i \bmod 3) + 1,\ 1 \le i \le 3$$

$$s(P_i, P_j) = \frac{Y_j - Y_i}{X_j - X_i} \tag{6}$$

Two error functions for rotation and translation are thus defined respectively:

$$E_R(p^{k-1}, \pi^k, P_A^k) = w_1 e_1(p^{k-1}, \pi^k, 0, P_A^k) + w_2 e_2 + w_3 e_3 + w_4 e_4 \tag{7}$$

$$E_T(p^{k-1}, \delta^k, P_A^k) = e_1(p^{k-1}, 0, \delta^k, P_A^k) \tag{8}$$

where E_R evaluates the error for rotation parameters π^k, and E_T evaluates the error for translation parameters δ^k. Coefficients w_i's are weighting factors, $1 \le i \le 4$.

It's obvious that E_T only depends on e_1, since the amount of translational offset is the distance between P_M and P_A. The definition of E_R is more complicated, and is better to be described in terms of four metrics.

From perspective projection, translations won't change the triangle's shape on the 2D image plane, but rotations will. Hence, it's important to have some metrics to measure shape differences between two triangles before and after rotations. In other words, metrics to measure shape similarity are necessary.

Metric e_1 measures the vertex distances between P_M and P_A, thus returns the translation offsets required to adjust P_M to approach P_A. The other three metrics e_2, e_3, and e_4 are then designed to measure the shape similarity of P_M and P_A. Metric e_2 also measures vertex distances, but P_M and P_A are translated first so that their gravity centers are aligned together. In metric e_1, if P_M and P_A have the same shape (which means that only translations are necessary) but with different gravity centers , the value of e_1 will be dominated by the distance between their gravity centers. However if we only know e_1, we won't be able to decide whether more rotations or more translations should be applied in the next iteration, since e_1 only tells us that P_M and P_A are not close enough. Rewrite equation (3), we obtain

$$
\begin{aligned}
&e_1\left(p^{k-1},\pi^k,\delta^k,P_A^k\right)\\
&=\sum_{i=1}^{3}\left(\left(\left(\hat{X}_{M,i}+X_{M,C}\right)-\left(\hat{X}_{A,i}+X_{A,C}\right)\right)^2+\left(\left(\hat{Y}_{M,i}+Y_{M,C}\right)-\left(\hat{Y}_{A,i}+Y_{A,C}\right)\right)^2\right)\\
&=\sum_{i=1}^{3}\left(\left(\left(\hat{X}_{M,i}-\hat{X}_{A,i}\right)+\left(X_{M,C}-X_{A,C}\right)\right)^2+\left(\left(\hat{Y}_{M,i}-\hat{Y}_{A,i}\right)+\left(Y_{M,C}-Y_{A,C}\right)\right)^2\right)\\
&\quad\text{where } \begin{matrix}\hat{X}_i\\ \hat{Y}_i\end{matrix} = \begin{matrix}X_i\\ Y_i\end{matrix} - \begin{matrix}X_C\\ Y_C\end{matrix},\quad \begin{matrix}X_C\\ Y_C\end{matrix} \text{ is defined in equation (4)}
\end{aligned}
\tag{9}
$$

$\left(\hat{X} \quad \hat{Y}\right)^t$ is defined relative to the gravity center. From equation (9), we see that the distance between two gravity centers will dominate the value of e_1, if $\left(\hat{X}_{M,i}-\hat{X}_{A,i}\right)$ and $\left(\hat{Y}_{M,i}-\hat{Y}_{A,i}\right)$ are small, which means P_M and P_A are similar in shape. To remove the effect of gravity centers, metric e_2 is defined. If the value of e_2 is small, but the value of e_1 is large, we tend to apply translations in the next iteration, since the shape is quite similar.

Metrics e_3 and e_4 add more constraints on shape similarity by requiring two triangles to have the same edge ratios and edge slopes. These two metrics are theoretically equivalent, but in practice, because of numerical errors and noise, they reinforce each other.

$$E\left(p^{k-1},\pi^k;\delta^k,P_A^k\right)=E_R\left(p^{k-1},\pi^k,P_A^k\right)+E_T\left(p^{k-1},\delta^k,P_A^k\right) \tag{10}$$

The objective function is finally defined as below:

Our task is to minimize $E\left(p^{k-1},\pi^k,\delta^k,P_A^K\right)$, the accumulated sum of errors of the two error functions, with a proposed iterative method.

To minimize the objective function $E\left(p^{k-1},\pi^k,\delta^k,P_A^K\right)$ with unknown parameters π^k and δ^k, we borrow the concept from gradient descent algorithms. It's obviously that our objective function is non-differentiable, and hence gradient descent algorithms cannot be applied.

Given a real valued function $f:\Re^n \to \Re$, the gradient descent algorithm finds the minimal value of $f(x)$ by iteratively updating x at iteration $k+1$,

$$x^{k+1}=x^k-\alpha_k\nabla f\left(x^k\right) \qquad \alpha_k>0 \tag{11}$$

The coefficient α_k is a positive scalar called the *step size*. The step size α_k is a free variable that should be carefully chosen. If the step size is too small, minimization progress would be slow. Therefore we employ a *bold-driver* method to adaptively adjust the value of α_k. Hence equation (11) is rewritten as below:

$$
\begin{aligned}
&x^{k+1}=x^k-\alpha_k\nabla f\left(x^k\right)\\
&\alpha_{k+1}=\begin{matrix}\rho\alpha^k & \rho\ge 1, & \text{if } f\left(x^{k+1}\right)<f\left(x^k\right)\\ \sigma\alpha^k & 0<\sigma<1, & \text{if } f\left(x^{k+1}\right)\ge f\left(x^k\right)\end{matrix}\\
&\alpha_0>0
\end{aligned}
\tag{12}
$$

Because our objective function $E\left(p^{k-1},\pi^k,\delta^k,P_A^K\right)$ is non-differentiable, we cannot obtain the gradient ∇E, so we approach the gradient descent algorithm by evaluating the objective function (10) on twelve possible directions, and choosing the set of parameters π_l^k and δ_l^k that cause the minimal value of $E\left(p^{k-1},\pi_l^k,\delta_l^k,P_A^K\right)$ at iteration l. For simplicity, we represent $E\left(p^{k-1},\pi_l^k,\delta_l^k,P_A^K\right)$ with $E\left(\pi_l,\delta_l\right)$, or more compactly, $E\left(a^l\right)$,

where $a^l=\left(\pi_l^k \quad \delta_l^k\right)^t=\left(\theta \quad \psi \quad \phi \quad t_x, \quad t_y, \quad t_z,\right)^t$.

The twelve directions to evaluate $E\left(a^l\right)$ is defined by

$$
\nabla_1=\begin{matrix}+1\\0\\0\\0\\0\\0\end{matrix},\ \nabla_2=\begin{matrix}-1\\0\\0\\0\\0\\0\end{matrix},\ \nabla_3=\begin{matrix}0\\+1\\0\\0\\0\\0\end{matrix},\ \nabla_4=\begin{matrix}0\\-1\\0\\0\\0\\0\end{matrix},\ \nabla_5=\begin{matrix}0\\0\\+1\\0\\0\\0\end{matrix},\ \nabla_6=\begin{matrix}0\\0\\-1\\0\\0\\0\end{matrix}
$$

$$
\nabla_7=\begin{matrix}0\\0\\0\\+1\\0\\0\end{matrix},\ \nabla_8=\begin{matrix}0\\0\\0\\-1\\0\\0\end{matrix},\ \nabla_9=\begin{matrix}0\\0\\0\\0\\+1\\0\end{matrix},\ \nabla_{10}=\begin{matrix}0\\0\\0\\0\\-1\\0\end{matrix},\ \nabla_{11}=\begin{matrix}0\\0\\0\\0\\0\\+1\end{matrix},\ \nabla_{12}=\begin{matrix}0\\0\\0\\0\\0\\-1\end{matrix}
\tag{13}
$$

At $(l+1)$th iteration, all these twelve directions are evaluated by using

$$
\begin{aligned}
&a^{l+1}=a^l-\alpha_l\nabla_i, \qquad 1\le i\le 6\\
&a^{l+1}=a^l-\beta_l\nabla_i, \qquad 7\le i\le 12\\
&\alpha_{l+1}=\begin{matrix}\rho\alpha_l & \rho\ge 1, & \text{if } E\left(a^{l+1}\right)<E\left(a^l\right)\\ \sigma\alpha_l & 0<\sigma<1, & \text{if } E\left(a^{l+1}\right)\ge E\left(a^l\right)\end{matrix}\\
&\beta_{l+1}=\begin{matrix}\rho\beta_l & \rho\ge 1, & \text{if } E\left(a^{l+1}\right)<E\left(a^l\right)\\ \sigma\beta_l & 0<\sigma<1, & \text{if } E\left(a^{l+1}\right)\ge E\left(a^l\right)\end{matrix}\\
&\alpha_0,\beta_0>0
\end{aligned}
\tag{14}
$$

We now need to find the locally optimal parameter set $\hat{a}^{l+1}$ that best minimizes the objective function $E(a^{l+1})$:

$$\hat{a}^{l+1} = \arg(\min_{j} E(a^{j})) \tag{15}$$

The value of $\hat{a}^{l+1}$ is chosen to be the one with the smallest $E(a^{l+1})$. The iteration stops when either the maximum number N of iterations is reached or the value of $E(\hat{a}^{l+1})$ is smaller than a given error threshold ε.

At this moment, we can successfully recover rotation parameters π^k, and translation parameters δ^k, at time instant k. However the termination criteria of the iteration can be either when the error is small enough, or when the allowed maximum number of iterations is achieved. In the later case, the obtained results are most likely to be incorrect. To recover from incorrect estimations, a prediction algorithm, the *Grey predictor* [19], is thus employed. The Grey predictor applies Grey system theory to predict next parameters from previous historical data. It has been shown that the Grey predictor has a behavior similar to the Kalman-filter based prediction [19].

4. User Assisted Error Recovery

If there exists a point P_i^{k-1} that fails to find its corresponding point on image k, the tracking procedure is stopped, and an error handling procedure is invoked. Matching failures are possible when a feature point is located outside of the searching window W_S. The gray and color differences of all the selected points of interest will be larger than the pre-defined threshold ε_a and ε_c. The matching failures can be detected if no points are returned from the tracking procedure. At this moment, an error handling procedure is invoked.

When a matching failure is detected, a bounding box with two circles is overlapped on the image, as shown in Figure 2. The bounding box gives a rough estimation of the subject's head size and position, and the two circles denote the positions of two eyes. The subject has to move his head so that his two eyes are located within the two circles. Positions of the bounding box and the two circles are derived at the beginning. The size of the bounding box is approximated by the distance L_{eyes} between two eyes. If positions of the two eye are represented by $P_l = (X_l, Y_l)$ and $P_r = (X_r, Y_r)$, we would have $L_{eyes} = \sqrt{(X_r - X_l)^2 + (Y_r - Y_l)^2}$. The dimension of the bounding box is then approximated by $H \times W$, where the height $H = \rho_h \times L_{eyes}$, and the width $W = \rho_w \times L_{eyes}$. The center of the bounding box is at $C = \frac{X_l + X_r}{2}, \frac{Y_l + Y_r}{2} + \frac{\rho_h}{2} - 1 \times L_{eyes}$. As the subject is trying to match his eyes with the two circles, the error handling procedure repeats the tracking procedure.

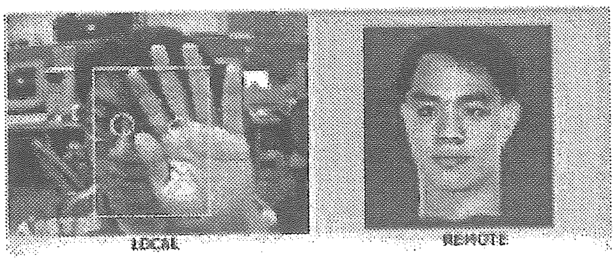

Figure 2. A box is prompted when tracking error occurs. The box is an approximation to the head, and the two circles represent the initial positions of eyes.

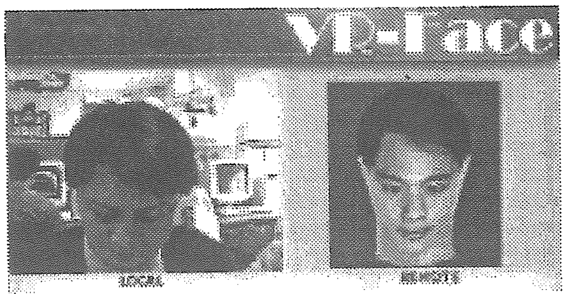

Figure4. A snapshot of the VR-Face system, a model-based visual communication system

In short, the objective of the error handling procedure is to request the subject to return to his original position, and then tries to re-initialize the three feature points automatically. This method is simple but effective. A trained subject can easily re-initialize the three feature points within several seconds.

5. Results

The proposed method has been implemented on Windows 95/98. Without any visual output, the procedure performs more than *250* times/sec on a PC with an Intel Pentium-II 233MHz CPU. This procedure has also been applied to live video. The average overall update rate is over 25 frames/sec with automatic feature tracking. The head model calibration process only requires a user to face to the camera for less than one second. In fact, users are not aware of this calibration process. From experiments, this procedure works for different people and different cameras.

Figure 3 shows a video sequence of the 3-D motion estimation resulted from live video clips. The estimation result is shown as a 3D texture-mapped head model to the right. During our simulation of three minutes at 25 fps, the average orientation error between the left and the right images is less than 2.7 degrees (standard deviation = 2.6), and the average translation error is less than 0.48 cm (standard deviation = 0.45). Part of the above simulation data is plotted in Figure 5.

At this moment, the developed real-time 3D head motion tracking technique has been applied to a model-based visual communication system, termed VR-Face, as shown in Figure 4. For the part of face synthesis, a customized 3D head model has to be first generated, where we use the method proposed by Lee and Thalmann in [20]. When a set of motion vectors is received, the 3D head model is transformed and rendered with texture mapping in real time, as shown in Figure 4. Executable programs are available at http://www.cmlab.csie.ntu.edu.tw/~tjyang/research/facel.html

6. Conclusions

This paper addresses the problem of 3D head motion tracking using a camera, and proposes a real-time motion estimation method that infers one's 3D head motion with only three facial features. The three facial features are two eyes and one nostril. Feature correspondence between two consecutive video frames is automatically established through the proposed feature tracking method, except that the initial locations of the three features are given manually. In addition, a mechanism for error handling is also described. Combining the proposed feature tracking, 3D head motion estimation, and error handling, we propose a framework for global head motion analysis. A model-based visual communication system has been developed based on the proposed 3D head motion tracking technique, together with a real-time face rendering module. The model-based visual communication system is developed on a regular PC with a low-cost camera mounted. In this system, a subject can move his head naturally, and the system will track and estimate the subject's 3D head motion without any markers on the face. A photo-realistic 3D head model rendered with texture-mapping will follow the subject's head motion to tilt, pan, or roll. Performance of the model-based visual communication system can achieve over 25 frames/sec on a PC with an Intel Pentium-II 400 MHz CPU. The system has been demonstrated in several public occasions and the results are reliable.

7. Acknowledgments

This work is part of the project "MPEG-4 Hybrid-Media Virtual Environment" being developed at Communication and Multimedia Laboratory. The project is supported by the National Science Council under grant number NSC 88-2622-E-002-002.

8. References

[1] Roberto Brunelli, Tomaso Poggio, "*Face Recognition: Features versus Templates*," IEEE Trans. On Pattern Analysis and Machine Intelligence, 15(10), Oct. 1993, pp. 1042-1052.

[2] Hans Peter Graf, Tsuhan Chen, Eric Petajan, Eric Cosatto, "*Locating Faces and Facial Parts*," Proc. of the 1st International Conference on Automatic Face and Gesture Recognition, Zurich, Switzerland, 1995, pp. 41-46.

[3] Hans Peter Graf, Eric Cosatto, Dave Gibbon, Michael Kocheisen, "*Multi-Modal System for Locating Heads and Faces*," Proc. of the 2nd International Conference on Automatic Face and Gesture Recognition, Killington, Vermont, USA, 1996, pp. 88-93.

[4] Jochen Heinzmann, Alexander Zelinsky, "*Robust Real-Time Face Tracking and Gesture Recognition*," Proc. of the International Joint Conference on Artifical Intelligence, IJCAI'97, Vol. 2, pp. 1525-1530, Aug. 1997.

[5] Antonio Colmenarez, Ricardo Lopez, Thomas S. Huang, "*3D Model-Based Head Tracking*," Proc. of Visual Communications and Image Processing, San Jose, CA, 1997. Http://troi.ifp.uiuc.edu/~antonio/Papers/ vcip97.ps.gz.

[6] Liang Zhang, "*Tracking a Face for Knowledge-Based Coding of Videophone Sequences*," Signal Processing: Image Communication, 10, 1997, pp. 93-114.

[7] Jorn Ostermann, "*Object-Based Analysis-Synthesis Coding Based on the Source Model of Moving Rigid 3D Objects*," Signal Processing: Image Communication, 6, pp. 143-161, 1994.

[8] Sumit Basu, Irfan Essa, Alex Pentland, "*Motion Regularation for Model-Based Head Tracking*," Technical Report 362, MIT Media Laboratory, Perceptual Computing Section, Jan. 1996. Http://www-white.media.mit.edu/vismod/.

[9] Ricardo Lopez, Antonio Colmenarez, Thomas S. Huang, "*Head and Feature Tracking for Model-based Video Coding*," Proc. of International Workshop on Synthetic-Natural Hybrid Coding and 3-D Imaging, Greece, 1997. Http://troi.ifp.uiuc.edu/~antonio/Papers/snhc97.ps.gz.

[10] Douglas DeCarlo, Dimitris Metaxas, "*The Integration of Optical Flow and Deformable Models with Applications to Human Face Shape and Motion Estimation*," Proc. of Computer Vision and Pattern Recognition (CVPR'96), pp. 231-238, 1996.

[11] Lars-Peter Bala, Kay Talmi, Jin Liu, "*Automatic Detection and Tracking of Faces and Facial Features in Video Sequences*," Picture Coding Symposium 1997, Berlin, Germany, September, 1997.

[12] Eli Saber, A. Murat Tekalp, "Frontal-View Face Detection and Facial Feature Extraction using Color, Shape and Symmetry Based Cost Functions," Pattern Recognition Letters, Vol. 19, 1998, pp. 669-680.

[13] Jie Yang, Alex Waibel, "A Real-Time Face Tracker," Proc. of the 3rd IEEE Workshop on Applications of Computer Vision, Sarasota, Florida, USA, 1996, pp. 142-147 (also in CMU Technical Report CMU-CS-95-210, 1995).

[14] Jie Yang, Weier Lu, Alex Waibel, "*Skin-Color Modeling and Adaptation*," CMU Technical Report CMU-CS-97-146, May, 1997.

[15] Gary R. Bradski, "*Computer Vision Face Tracking for Use in a Perceptual User Interface*," Intel Journal 1998 Q2.

[16] Kentaro Toyama, "*Prolegomena for Robust Face Tracking*," Microsoft Research Technical Report, MSR-TR-98-65, Nov. 1998.

[17] Thomas S. Huang, and Arun N. Netravali, "*Motion and Structure from Feature Correspondences: A Review*," Proceedings of the IEEE, 82(2), pp. 252-268, Feb. 1994.

[18] Christian Heipke, "*Overview of Image Matching Techniques*," Proc. of OEEPE Workshop on the Application of Digital Photogrammetric Workstations, Lausanne, Switzerland, Mar. 4-6, 1996. Http://dgrwww.epfl.ch/PHOT/publicat/wks96/Art_3_1.html.

[19] Jiann-Rong Wu, Ming Ouhyoung, "*Reducing The Latency in Head-Mounted Display by A Novel Prediction Method Using Grey System Theory*," Computer Graphics Forum, 13(3), pp. C503-512, 1994 (also in EUROGRAPHICS'94, Oslo, Norway, 1994).

[20] Won-Sook Lee, Nadia Magnenat Thalmann, "*Head Modeling from Pictures and Morphing in 3D with Image*

Metamorphosis Based on Triangulation," Proc. of CAPTECH'98, Geneva, Switzerland, 1998, pp. 254-267.

[21] Tzong-Jer Yang, Fu-Che Wu, Ming Ouhyoung, "*Real-Time 3D Head Motion Estimation in Facial Image Coding*," Proc. of Multimedia Modeling, Lausanne, Switzerland, Oct. 12-15, 1998, pp. 50-51.

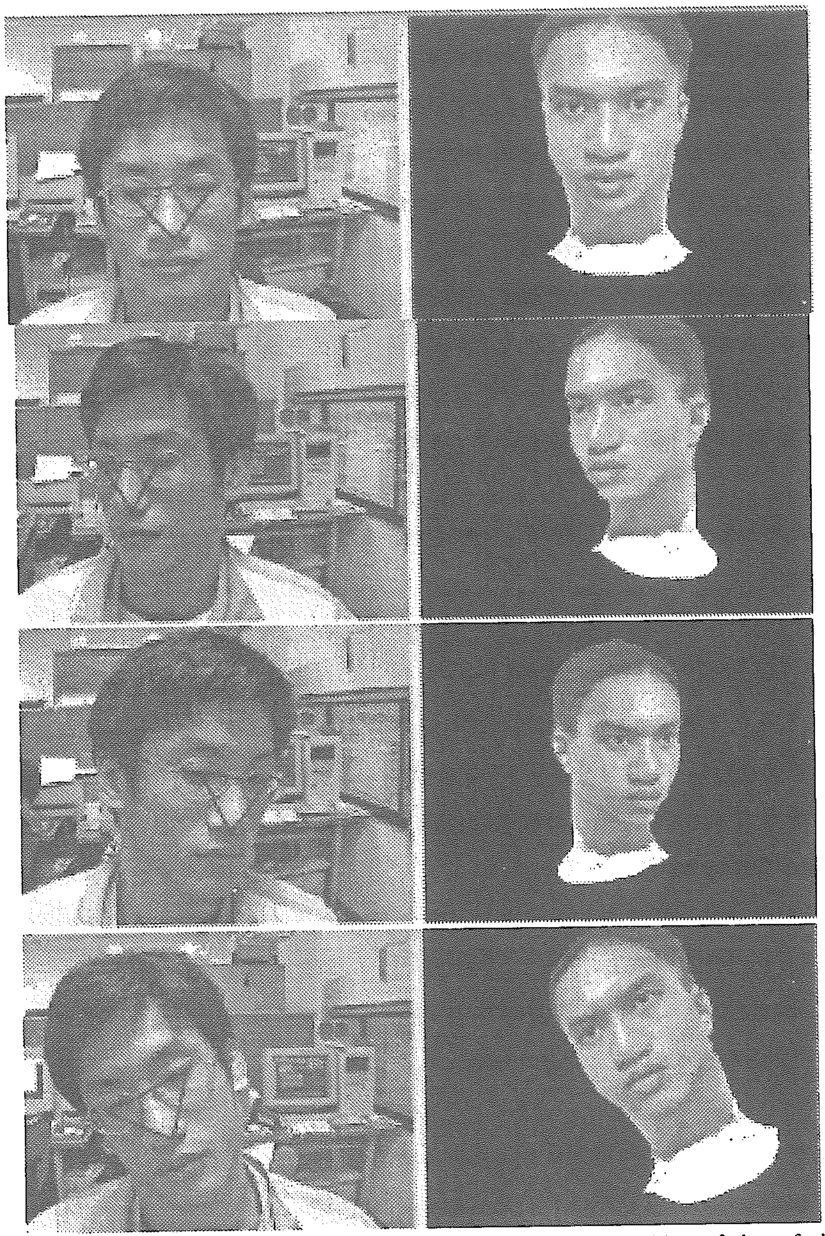

Figure 3. Real-time 3D head motion tracking with automatic tracking of three facial features. The triangle on the face consists of three facial features. A texture-mapped head model to the right is rendered with corresponding orientations and positions.