Hands-free navigation in VR environments by tracking the head

Sing Bing Kang

Cambridge Research Laboratory
Technical Report Series

CRL 97/1

March 1997

Cambridge Research Laboratory

The Cambridge Research Laboratory was founded in 1987 to advance the state of the art in both core computing and human-computer interaction, and to use the knowledge so gained to support the Company's corporate objectives. We believe this is best accomplished through interconnected pursuits in technology creation, advanced systems engineering, and business development. We are actively investigating scalable computing; mobile computing; vision-based human and scene sensing; speech interaction; computer-animated synthetic persona; intelligent information appliances; and the capture, coding, storage, indexing, retrieval, decoding, and rendering of multimedia data. We recognize and embrace a technology creation model which is characterized by three major phases:

Freedom: The life blood of the Laboratory comes from the observations and imaginations of our research staff. It is here that challenging research problems are uncovered (through discussions with customers, through interactions with others in the Corporation, through other professional interactions, through reading, and the like) or that new ideas are born. For any such problem or idea, this phase culminates in the nucleation of a project team around a well articulated central research question and the outlining of a research plan.

Focus: Once a team is formed, we aggressively pursue the creation of new technology based on the plan. This may involve direct collaboration with other technical professionals inside and outside the Corporation. This phase culminates in the demonstrable creation of new technology which may take any of a number of forms - a journal article, a technical talk, a working prototype, a patent application, or some combination of these. The research team is typically augmented with other resident professionals—engineering and business development—who work as integral members of the core team to prepare preliminary plans for how best to leverage this new knowledge, either through internal transfer of technology or through other means.

Follow-through: We actively pursue taking the best technologies to the marketplace. For those opportunities which are not immediately transferred internally and where the team has identified a significant opportunity, the business development and engineering staff will lead early-stage commercial development, often in conjunction with members of the research staff. While the value to the Corporation of taking these new ideas to the market is clear, it also has a significant positive impact on our future research work by providing the means to understand intimately the problems and opportunities in the market and to more fully exercise our ideas and concepts in real-world settings.

Throughout this process, communicating our understanding is a critical part of what we do, and participating in the larger technical community—through the publication of refereed journal articles and the presentation of our ideas at conferences–is essential. Our technical report series supports and facilitates broad and early dissemination of our work. We welcome your feedback on its effectiveness.

Robert A. Iannucci, Ph.D.
Director

Hands-free navigation in VR environments by tracking the head

Sing Bing Kang

March 1997

Abstract

Conventional methods of navigating within a virtual reality environment involve the use of interfaces such as keyboards, hand-held input devices such as joysticks, mice, and trackballs, and hand-worn data gloves. While these devices are mostly adequate, they are rather obstrusive and require some amount of training to use. Researchers have begun investigation into interfaces that have the capability to interpret human gestures visually.

In this document, we describe an approach used to navigate virtual reality environments by tracking the pose (translation and orientation) of the user's face. This "hands-free" navigation is simple, intuitive, and unobstrusive. It requires only commercially available products such as a camera and an image digitizer. The pose of the face is determined by warping a reference face image to minimize intensity difference between the warped reference face image and the current face image. This is more robust because all pixels in the face are used, in contrast to detecting only selected facial features. In addition, the proposed approach does not require a geometric model of the face.

CRL Technical reports are available on the CRL's web page at
http://www.crl.research.digital.com.

Digital Equipment Corporation
Cambridge Research Laboratory
One Kendall Square, Building 700
Cambridge, Massachusetts 02139

Contents

List of Figures

1 Introduction

Conventional methods of navigating within a virtual reality environment involve the use of interfaces such as keyboards, hand-held input devices such as joysticks, mice, and trackballs, and hand-worn data gloves. While these devices are mostly adequate, they are rather obstrusive and require some amount of training to use. In addition, because of the constant physical use and manipulation, they either have limited life or require some degree of maintenance. Researchers have begun investigation into *natural* interfaces that are intuitively simple and unobstrusive to the user. By natural, we mean communication by way of human gestures and/or speech.

The proposed approach is designed to address the problem of navigating within a virtual environment without the use of keyboards, hand-held input devices, or data gloves. The approach that we take is to track the pose (i.e., translation and orientation) of the face and use that information to move and orient the virtual environment accordingly. This method of navigating the virtual environment is very intuitive and easy. Part of the novelty of the approach is the tracking of the entire face image without the use of any geometric face model. The simplicity of the approach allows fast tracking without using specialized image processing boards. The speed of face tracking currently achieved is 4 frames per second on a DEC AlphaStation 600 with an image patch size of 98x80 pixels.

1.1 Relevant work

Approaches to controlling interaction in a virtual environment have been mostly limited to using hand gestures for games (using a hand worn device such as a Mattel glove) or for manipulating virtual objects using a dataglove (e.g., [18]).

Other work that are relevant to this approach relate to face tracking. In [3], the full face is tracked using a detailed face model that relies on image intensity values, deformable model dynamics, and optical flow. This representation can be used to track facial expressions. Due to its complexity, processing between frames is reported to take 3 seconds each on a 200 MHz SGI machine. Initialization of the face model on the real image involves manually marking face locations, and takes 2 minutes on the same SGI machine.

In [5], the face model in the form of a 3-D mesh is used. However, the emphasis of this work is to recognize facial expressions, and it assumes that there is no facial global translation or rotation.

Work reported in [6, 8] are probably the most relevant to the approach. However, they require detection of specific facial features and ratios of distances between facial features. In [6], the gaze direction is estimated from the locations of specific features of the face, namely eye corners, tip of the nose, and corners of the mouth. As described in the paper, these features are manually chosen. In [8], 3-D head orientation is estimated by tracking five points on the face (four at the eye corners and one at the tip of the nose). Again the facial features are selected by hand.

Also relevant is [14], which describes a real-time (20 frames per second) facial feature tracking system based on template matching. The system includes the DataCube real-time image processing equipment. The face and mouth areas are extracted using color histogramming while the eyes are tracked using sequential template matching. An application cited is the visual mouse, which emulates the functionality of the physical mouse through eye position (cursor movement) and mouth shape change (clicking operation). Again this system tracks specific features of the face; it is not clear if this form of tracking (sequential) is stable over time and whether reliable face orientation can be derived from so few features.

In their work on facial image coding, Li *et al.* [10] use a 3-D planar polygonized face model and assume 3-D affine motion of points. They track the motion of the face model (both local and global) using optic flow to estimate the facial action units (based on the facial action coding system, or FACS [4]). A feedback loop scheme is employed to minimize the error between the synthetically generated face image based on motion estimates and the true face image. However, they have to estimate the depth of the face, assumed segmented out, in the scene. The feature node points of the face model are manually adjusted to initially fit the face in the image. No timing results were given in their paper.

Azarbayejani et al.'s [1] system tracks manually picked points on the head, and based on recursive structure from motion estimates and Extended Kalman filtering, determine the 3-D pose of the head. The cited translation and rotation error, with respect to the Polhemus tracker estimates, are 1.67 cm and 2.4° respectively. The frame rate achieved is 10 frames per second. Their system requires local point feature trackers.

The approach described in [19] is that of block-based template matching. The idea is to take many image samples of faces (152 images of 22 people), partition the images into chunks of blocks (each of which is 5x7 pixels), and compute statistics of the intensity and strength of edges within each block. The results are then used as a template to determine the existence of a face in an image as well as its orientation. No timing results are given. In comparison, the initial steps of sampling faces and performing statistical analysis of the samples are not required in this approach. In addition, for the work described in [19], the orientation of the face is determined by interpolating between known sampled face orientation. The approach measures directly the face orientation without any interpolation scheme.

1.2 The "hands-off" navigation approach

Our concept of "hands-off" navigation is depicted in Figure 1. The camera is mounted in front and above the user; it views the user's face from a tilted angle. The system requires a reference face image, and this image is captured for initialization. The reference face image is that of the user in a neutral pose, where he/she is facing directly ahead below the camera. To determine the head translation and orientation, the face tracker warps the reference face image to minimize the difference between the warped reference face image and the current face image. This is equivalent to using the reference face image as a globally deformable template. The warping matrix transformation is then decomposed to yield the face translation and orientation. Subsequently, the view point of the 3-D virtual environment changes accordingly.

The software is written in C, and is run in a DEC AlphaStation 600. The 3-D virtual environment viewer used for our system is VRweb, which is originally developed by the Institute for Information Processing and Computer Supported New Media (IICM), Graz University of Technology, in Austria[1]. We customized it to allow TCP/IP communication with the face tracker.

In navigating a virtual environment, it is very likely that the user would not want to rotate the scene about the viewing direction. Hence we adopt this convenient assumption, and disable control of rotational motion about the viewing axis (i.e., rotation about t_z vector in Figure 1).

This research is done in connection with the Smart Kiosk project at Cambridge Research Lab, Digital Equipment Corp. [20]. The Smart Kiosk can be considered as an enhanced version of the Automatic Teller Machine, with the added capability of being able to interact with the user through body tracking, and gesture and speech recognition. The "hands-off" capability would enable the Smart Kiosk to allow the user to navigate the local virtual environment as an information dispensing appliance. The local virtual environment could be created using either direct range data or stereo from multiple real images [9].

1.3 Organization of document

We first review the most general global motion tracking, namely full 2-D perspective tracking in Section 2. Here we also describe how 2-D motion matrix is decomposed directly into various motion parameters such as translation, magnification, and skew. Since the face is assumed to be relatively far away from the camera, we can assume an affine model rather than a full 2-D perspective model. Section 4 describes how the motion parameters derived from Section 2 can be used to extract head translation and orientation. Results are shown in Section 6, followed by

[1]The web site for the VRweb browser is http://www.iicm.edu/vrweb.

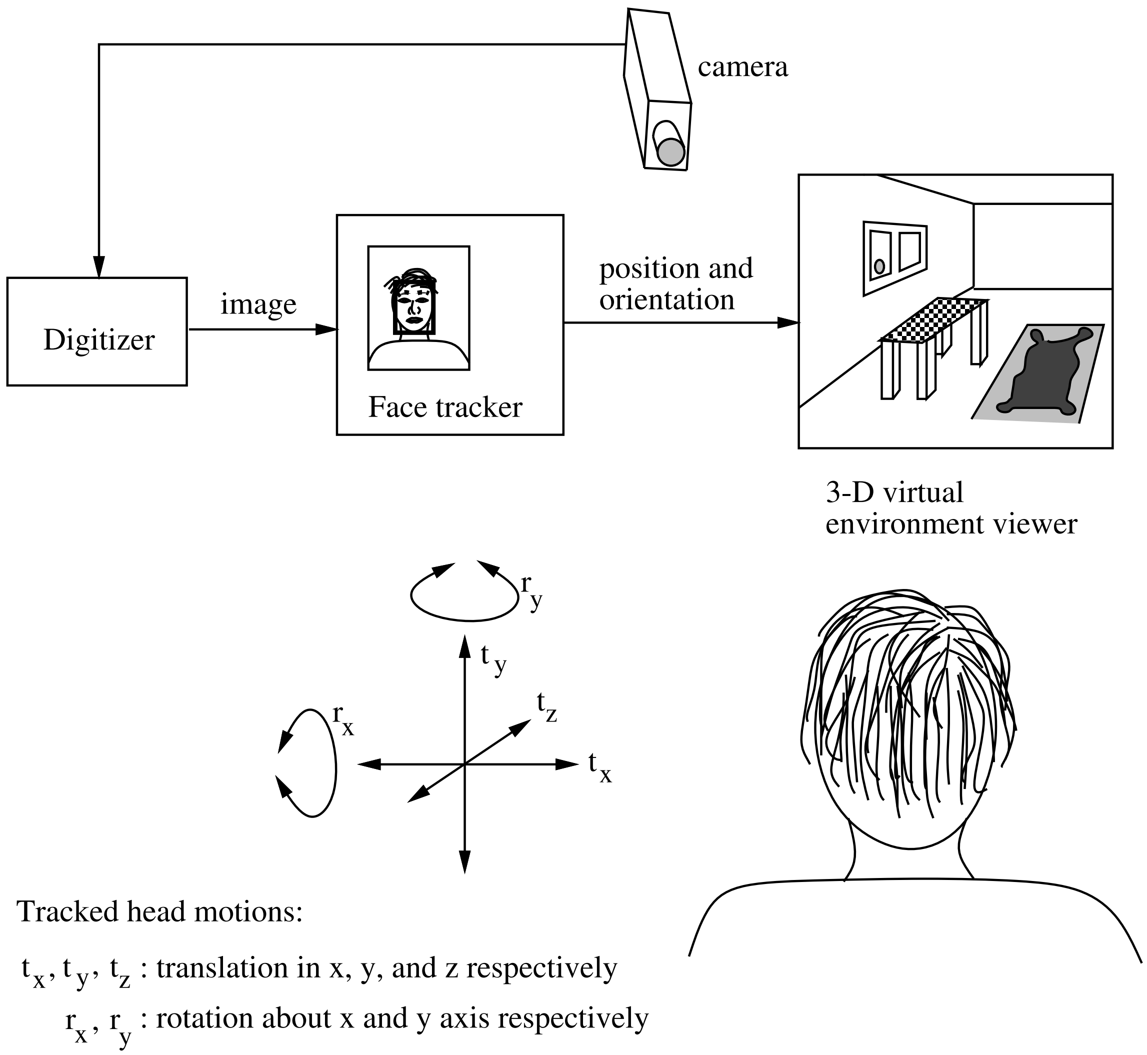


Figure 1: Concept of hands-free navigation.

discussion on the implication of the method in Section 7. Finally, we summarize the approach in Section 8.

2 2-D perspective tracking

The approach in tracking is the same as the approach for image registration described in [13, 16], i.e., to directly minimize the discrepancy in intensities between pairs of images after applying the transformation we are recovering. This has the advantage of not requiring any easily identifiable feature points, and of being statistically optimal once we are in the vicinity of the true solution [17]. The technique minimizes the sum of the squared intensity errors

$$E = \sum_i [I'(x'_i, y'_i) - I(x_i, y_i)]^2 = \sum_i e_i^2 \tag{1}$$

subject to the constraint

$$x'_i = \frac{m_{00}x_i + m_{01}y_i + m_{02}}{m_{20}x_i + m_{21}y_i + 1}, \; y'_i = \frac{m_{10}x_i + m_{11}y_i + m_{12}}{m_{20}x_i + m_{21}y_i + 1}. \tag{2}$$

The objective function E is applied over the region of interest.

The minimization of E is done using the Levenberg-Marquardt iterative non-linear minimization algorithm [12]. This algorithm requires the computation of the partial derivatives of e_i with respect to the unknown motion parameters $\{m_{00} \ldots m_{21}\}$. These are straightforward to compute, i.e.,

$$\frac{\partial e_i}{\partial m_0} = \frac{x_i}{D_i}\frac{\partial I'}{\partial x'}, \quad \ldots \quad , \frac{\partial e_i}{\partial m_7} = -\frac{y_i}{D_i}\left(x'_i\frac{\partial I'}{\partial x'} + y'_i\frac{\partial I'}{\partial y'}\right), \tag{3}$$

where D_i is the denominator in (2), and $(\partial I'/\partial x', \partial I'/\partial y')$ is the image intensity gradient of I' at (x'_i, y'_i). From these partials, the Levenberg-Marquardt algorithm computes an approximate Hessian matrix A and the weighted gradient vector $\mathbf{b}$ with components

$$a_{kl} = 2\sum_i \frac{\partial e_i}{\partial m_k}\frac{\partial e_i}{\partial m_l}, \qquad b_k = -2\sum_i e_i \frac{\partial e_i}{\partial m_k}, \tag{4}$$

and then updates the motion parameter estimate $\mathbf{m}$ by an amount $\Delta\mathbf{m} = (A + \mu I)^{-1}\mathbf{b}$, where μ is a time-varying stabilization parameter [12]. The advantage of using Levenberg-Marquardt over straightforward gradient descent is that it converges in fewer iterations.

To enable the tracker to be more tolerant of larger displacements, we employ a hierarchical scheme where coarse estimates are first found at coarse resolutions to be refined at higher resolutions. In our implementation, we can specify an arbitrary number of resolution levels and iteration at each level. We set the number of resolutions to be 3 and the number of iterations per level to be 3.

2.1 Decomposition of full 2-D perspective matrix

Given the full 2-D perspective matrix, we can decompose it into the following warping parameters:

- center point or displacement t_x and t_y (in x and y directions respectively)
- rotation angle θ_I (about the viewing axis)
- zoom factor ζ
- aspect ratio a
- skew factor s
- pinch parameters ξ_x and ξ_y (in x and y directions respectively)

Define

$$r_x = \zeta a \;\text{ and }\; r_y = \frac{\zeta}{a}, \tag{5}$$

and let $s_\theta = \sin\theta_I$ and $c_\theta = \cos\theta_I$.

The 2-D perspective matrix (which is first scaled such that $m_{22} = 1$) can be decomposed as follows:

$$\begin{aligned}
\begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & 1 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_x & 0 & 0 \\ 0 & r_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \\
&\quad \begin{pmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \xi_x & \xi_y & 1 \end{pmatrix} \\
&= \begin{pmatrix} c_\theta r_x + \xi_x t_x & c_\theta s r_x - s_\theta r_y + \xi_y t_x & t_x \\ s_\theta r_x + \xi_x t_y & s_\theta s r_x + c_\theta r_y + \xi_y t_y & t_y \\ \xi_x & \xi_y & 1 \end{pmatrix}
\end{aligned} \tag{6}$$

The recovery of the full 2-D perspective matrix from two images is generally relatively unstable if these images are small or have little intensity variation. From experiments, this has shown to be true, even with face images as large as about 100×100. As a result, we use instead an approximation of the 2-D perspective model, namely the 2-D affine model.

For the 2-D affine case, we set $m_{20} = m_{21} = 0$ and $\xi_x = \xi_y = 0$, giving

$$\begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_\theta r_x & c_\theta s r_x - s_\theta r_y & t_x \\ s_\theta r_x & s_\theta s r_x + c_\theta r_y & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

Next, we describe the affine camera, and show the cases when 2-D affine transformation is valid.

3 Affine camera

The 2-D affine transformation of the image is applicable for the *affine* camera. The affine camera has the projection matrix of the form

$$\mathrm{T}_{affine} = \begin{pmatrix} T_{00} & T_{01} & T_{02} & T_{03} \\ T_{10} & T_{11} & T_{12} & T_{13} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} M & \mathbf{m} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \tag{8}$$

If $\mathbf{p}$ is the 3-D point in space and $\mathbf{u}$ is the corresponding affine projection, then

$$\mathbf{u} = M\mathbf{p} + \mathbf{m} \tag{9}$$

In the affine camera model, all epipolar lines are parallel, and the epipoles are located at infinity in the image planes. This camera model is a generalization of the scaled orthographic (also known as weak perspective or paraperspective) camera model, which can be used as a good approximation if the change in relative object depth is small compared to its distance to the camera. For a fuller description of the affine and scaled orthographic camera models, the reader is encouraged to refer to the Appendix of [11].

As shown in [16], full 2-D perspective image transformation can only be used in cases of planar surfaces using the perspective camera model and rotation about the camera optic center. The 2-D affine image transformation can be used only in the cases of planar surfaces and *translation* under the affine camera model. To see this, we use the derivation similar to [15].

Let 3-D point $\mathbf{p}$ be a point on a planar patch whose unit normal is $\hat{\mathbf{n}}$. Let also $\hat{\mathbf{n}}_{\perp,1}$ and $\hat{\mathbf{n}}_{\perp,2}$ be the other two unit vectors that, with $\hat{\mathbf{n}}$, form the orthonormal bases of $\Re^3$. Thus $\mathbf{p}$ can be specified as

$$\mathbf{p} = \alpha\hat{\mathbf{n}}_{\perp,1} + \beta\hat{\mathbf{n}}_{\perp,2} + \lambda\hat{\mathbf{n}} \tag{10}$$

Note that $\mathbf{p}$ lies on the plane whose equation is $\mathbf{p} \cdot \hat{\mathbf{n}} = \lambda$, λ being a constant.

Let also R be the 3×3 rotation matrix such that

$$R\hat{\mathbf{n}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} ; R\hat{\mathbf{n}}_{\perp,1} = \begin{pmatrix} \mathbf{g}_1 \\ 0 \end{pmatrix} ; R\hat{\mathbf{n}}_{\perp,2} = \begin{pmatrix} \mathbf{g}_2 \\ 0 \end{pmatrix} \tag{11}$$

From (9),

$$\begin{aligned} \mathbf{u} &= MR^{-1}R\mathbf{p} + \mathbf{m} \\ &= M_R \begin{pmatrix} \alpha\mathbf{g}_1 + \beta\mathbf{g}_2 \\ \lambda \end{pmatrix} + \mathbf{m} \end{aligned} \tag{12}$$

where $M_R = MR^{-1}$. We now partition M_R as $(B|\mathbf{b})$, after which we can rewrite (12) as

$$\begin{aligned} \mathbf{u} &= B(\alpha\mathbf{g}_1 + \beta\mathbf{g}_2) + \lambda\mathbf{b} + \mathbf{m} \\ &= B(\alpha\mathbf{g}_1 + \beta\mathbf{g}_2) + \mathbf{b}_\lambda \end{aligned} \tag{13}$$

with $\mathbf{b}_\lambda = \lambda\mathbf{b} + \mathbf{m}$. Note that the only variables on the right hand side of (13) that depend on 3-D point location on the plane are α and β.

Similarly, for another affine camera, we have

$$\begin{aligned} \mathbf{u}' &= B'(\alpha\mathbf{g}_1 + \beta\mathbf{g}_2) + \lambda\mathbf{b}' + \mathbf{m}' \\ &= B'(\alpha\mathbf{g}_1 + \beta\mathbf{g}_2) + \mathbf{b}'_\lambda \end{aligned} \tag{14}$$

Eliminating $(\alpha\mathbf{g}_1 + \beta\mathbf{g}_2)$ from (13) and (14) yields

$$\mathbf{u}' = \Gamma\mathbf{u} + \epsilon \tag{15}$$

where $\Gamma = B'B^{-1}$ and $\epsilon = \mathbf{b}'_\lambda - \Gamma\mathbf{b}_\lambda$. Hence $\mathbf{u}'$ is an affine transformation of $\mathbf{u}$ for points on a plane.

Showing the applicability of the 2-D affine transformation under translation is trivial. If the 3-D translation is $\Delta\mathbf{p}$, then

$$\mathbf{u}' = M(\mathbf{p} + \Delta\mathbf{p}) + \mathbf{m} = \mathbf{u} + M\Delta\mathbf{p} \tag{16}$$

In this case, the image transform is a translation as well.

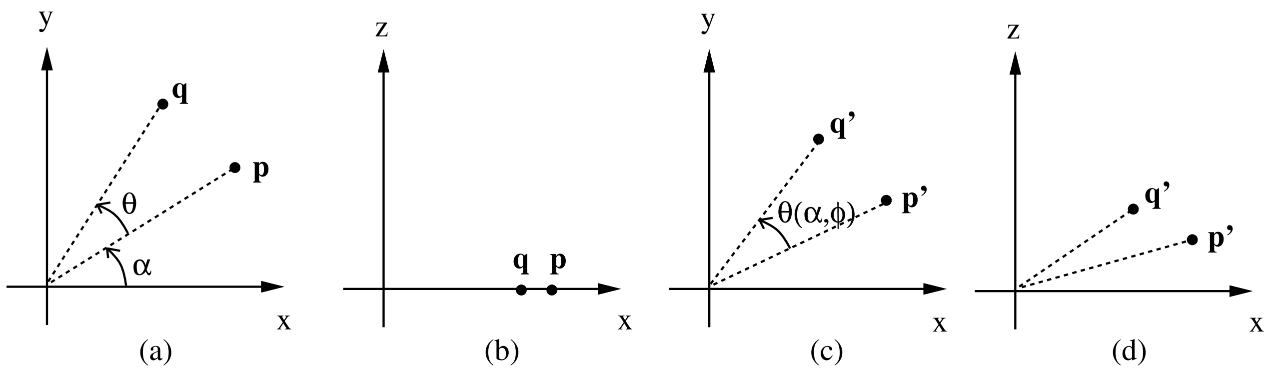


Figure 2: Effect of tilt (τ) on perceived rotation: (a) $\tau = \pi/2$ (top view), and (b) τ between 0 and $\pi/2$ (view from an angle above).

4 Using affine tracking to determine limited head pose

We have shown in the preceding section that 2-D affine transformation is valid for planar surfaces with an affine camera model. This is a good approximation if the face is far away enough from the camera. This has the effect of rendering the relative depth change in the face to be insignificant relative to the distance of the face from the camera. The face can then be approximated as a planar surface.

We capitalize on the decomposition of the 2-D affine matrix to determine head pose (location and orientation). However, in navigating a virtual environment, it is very likely that the user would not want to rotate the scene about the viewing direction. Hence we adopt this convenient assumption, and disable control of rotational motion about the viewing axis.

To keep the camera relatively unobstrusive to the user, it is better to position it higher up above the monitor and allow it to track the user's head from that location at a tilted angle. This location has a convenient side-effect; head rotations to either side map result in rotations about the viewing axis, which can be easily obtained from the affine matrix decomposition.

To see this, we consider viewing the head from the top (Figure 2(a)) and viewing the head at a tilted angle (Figure 2(b)). We assume that perspective effects are negligible. The point $\mathbf{p}$ has rotated by an angle θ to $\mathbf{q}$. Seen from an angle, the corresponding points are $\mathbf{p}'$ and $\mathbf{q}'$, and the perceived rotation angle is $\theta(\alpha, \tau)$; α being the original angle subtended by $\mathbf{p}$ with respect to the x-axis and τ is the tilt angle about the x-axis.

Without loss of generality, we can assume that both $\mathbf{p}$ and $\mathbf{q}$ are unit distance from the origin.

Hence we get

$$\mathbf{p} = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}; \mathbf{q} = \begin{pmatrix} \cos(\alpha+\theta) \\ \sin(\alpha+\theta) \end{pmatrix}; \mathbf{p}' = \begin{pmatrix} \cos\alpha \\ \sin\alpha\sin\tau \end{pmatrix}; \mathbf{q}' = \begin{pmatrix} \cos(\alpha+\theta) \\ \sin(\alpha+\theta)\sin\tau \end{pmatrix} \quad (17)$$

From (17), we can easily recover $\theta(\alpha, \tau)$:

$$\theta(\alpha,\tau) = \cos^{-1}\left(\frac{\cos\alpha\cos(\alpha+\theta) + \sin\alpha\sin(\alpha+\theta)\sin^2\tau}{\sqrt{\left(\cos^2(\alpha+\theta) + \sin^2(\alpha+\theta)\sin^2\tau\right)\left(\cos^2\alpha + \sin^2\alpha\sin^2\tau\right)}}\right) \quad (18)$$

For the case where the starting head pose is with the head facing horizontally below the camera (which we also assume), i.e., $\alpha = \pi/2$, from (18) we get

$$\theta(\frac{\pi}{2}, \tau) = \theta_I = \cos^{-1}\left(\frac{\cos\theta\sin\tau}{\sqrt{\sin^2\theta + \cos^2\theta\sin^2\tau}}\right) \quad (19)$$

To track the location of the head, we simply track the center of the affine patch (given by t_x and t_y). Motion in the forward/backward direction is given by the amount of zoom ζ. Due to the camera tilt, moving the head ahead has the undesirable effect of giving an image displacement in the y direction as well. The fix is to disable all other motion while zooming is detected.

If we know the tilt τ (from (22)), the true head rotation is then

$$\theta = \tan^{-1}(\tan\theta_I \sin\tau) \quad (20)$$

Finally, the head tilt is determined from the amount of y magnification r_y; because the camera is situated at a vertical angle with respect to the head, tilting up to face the camera results in larger r_y (y extent is larger than usual, hence greater than 1), while tilting the head down has the opposite effect. In the absence of all other motion parameters, the apparent facial height is (see Figure 3)

$$\Delta y = r_y \Delta y_0 = r_y \Delta y_{true} \cos\tau_0 \quad (21)$$

Hence the face tilt angle with respect to the camera is given by

$$\tau = \cos^{-1}\left(\frac{\Delta y}{\Delta y_{true}}\right) = \cos^{-1}(r_y \cos\tau_0) \quad (22)$$

To determine τ_0, we can apply a "calibration" technique in which the user tilts his/her head up and down once. The system keeps track of the maximum value of r_y, say $r_{y,max}$. Then

$$\tau_0 = \cos^{-1}\left(\frac{1}{r_{y,max}}\right) \quad (23)$$

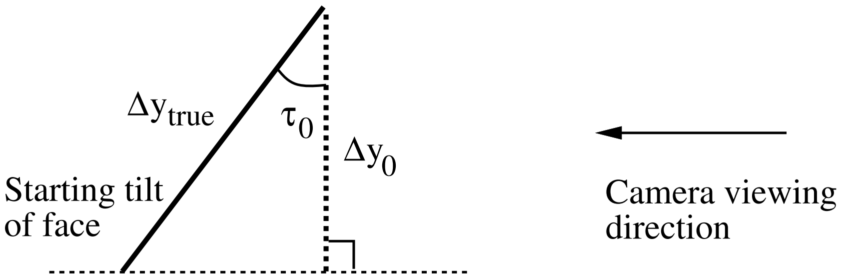


Figure 3: Starting tilt τ_0 of face (represented as planar patch) relative to the camera viewing direction. Δy_{true} is the true facial height while Δy_0 is the initial apparent facial height. Again we assume insignificant perspective effects.

We are interested in the face tilt angle with respect to the environment rather than with respect to the camera. Hence, the actual tilt angle used to control the orientation of the virtual environment is the displaced tilt angle given by

$$\tau' = \tau - \tau_0 = \cos^{-1}\left(r_y \cos\tau_0\right) - \tau_0 \tag{24}$$

5 Controlling the view

Even though we are able to extract the 5 pose parameters of the face, we are faced with the problem of using them to control the viewing of the virtual reality environment. One simple way would be to directly use the pose parameters to determine the absolute position and orientation of the viewpoint. However, this limits the viewpoint selection to the pose that the face can assume within the camera viewing space.

The alternative is to control the viewpoint incrementally, i.e., by changing the viewpoint of the virtual reality environment in direct response to the *change* in the pose of the face relative to the previous pose. To indicate continuous movement within the virtual reality environment beyond what the absolute pose of the face is able to convey, we added the ability for the face tracker to detect and respond to the lingering deviation from the reference pose. For example, if the user is interested in rotating to the left continuously, he/she would rotate his/her head to the left and maintain that posture. The system would respond by first turning the viewpoint of the virtual scene. However, because it detected the same deviated face posture longer than a preset time threshold (2 seconds in our case), it continues to rotate the viewpoint of the virtual scene in the same manner until the head posture changes.

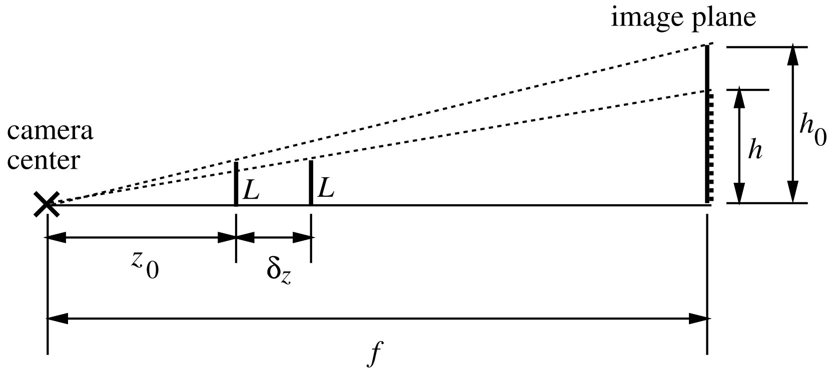


Figure 4: Effect of moving along the camera axis. f is the camera focal length, L is the length of the object, z_0 is the reference location, δ_z is the change in object location, and h_0 and h are the projected images lengths (in pixels) of the object at positions z_0 and $(z_0 + \delta_z)$ respectively.

To minimize the possibility of sudden jumps in consecutive viewpoints, we employ a simple Kalman-based filter to smooth the motion trajectory (see, for example, [7]).

While the orientation angles of tilt and pan can be directly used, we still need the translational scaling factors in x, y, and z. These are dependent on the relative scaling of the virtual environment. However, converting amount of zoom ζ (see Section 2.1) to change in depth z is less direct. From Figure 4, let z_0 be the reference depth location of the face. If the face has moved by δ_z, then by similarity of triangles, we have

$$\frac{L}{z_0} = \frac{h_0}{f} \text{ and } \frac{L}{z_0 + \delta_z} = \frac{h}{f}, \tag{25}$$

with $h = \zeta h_0$. Thus,

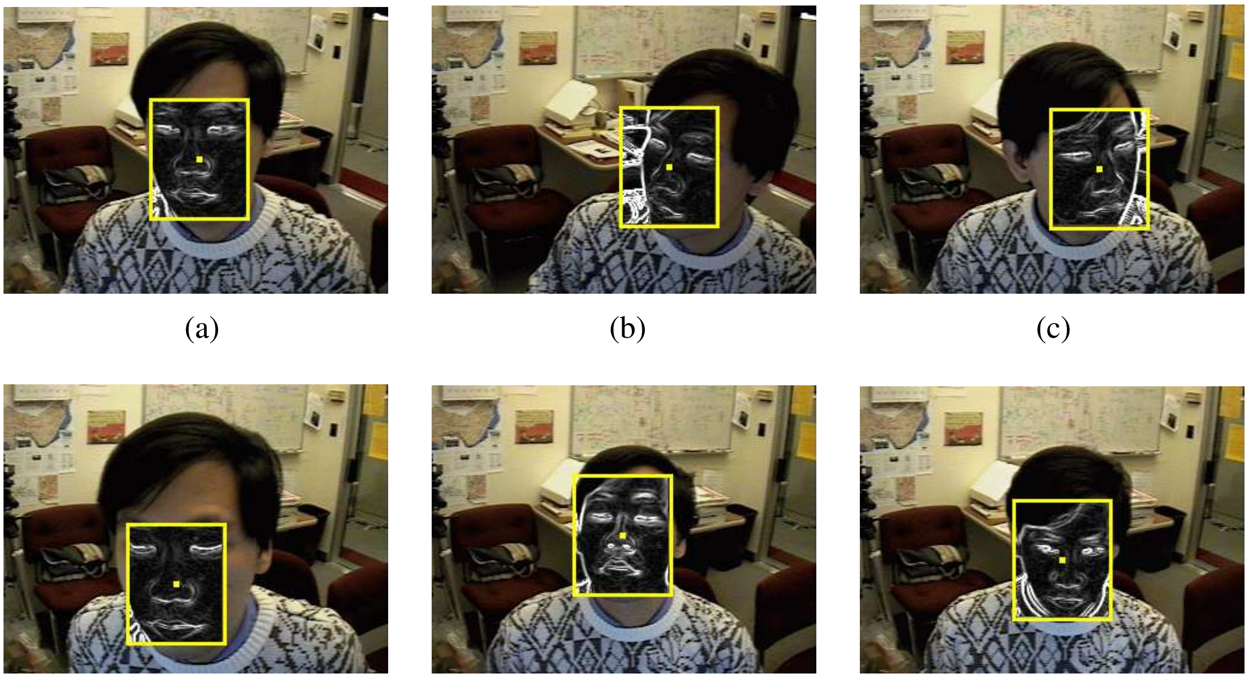
$$\frac{L}{z_0 + \delta_z} = \zeta \frac{h_0}{f} = \zeta \frac{L}{z_0} \tag{26}$$

from which

$$\delta_z = z_0 \left(\frac{1}{\zeta} - 1 \right) \tag{27}$$

6 Results

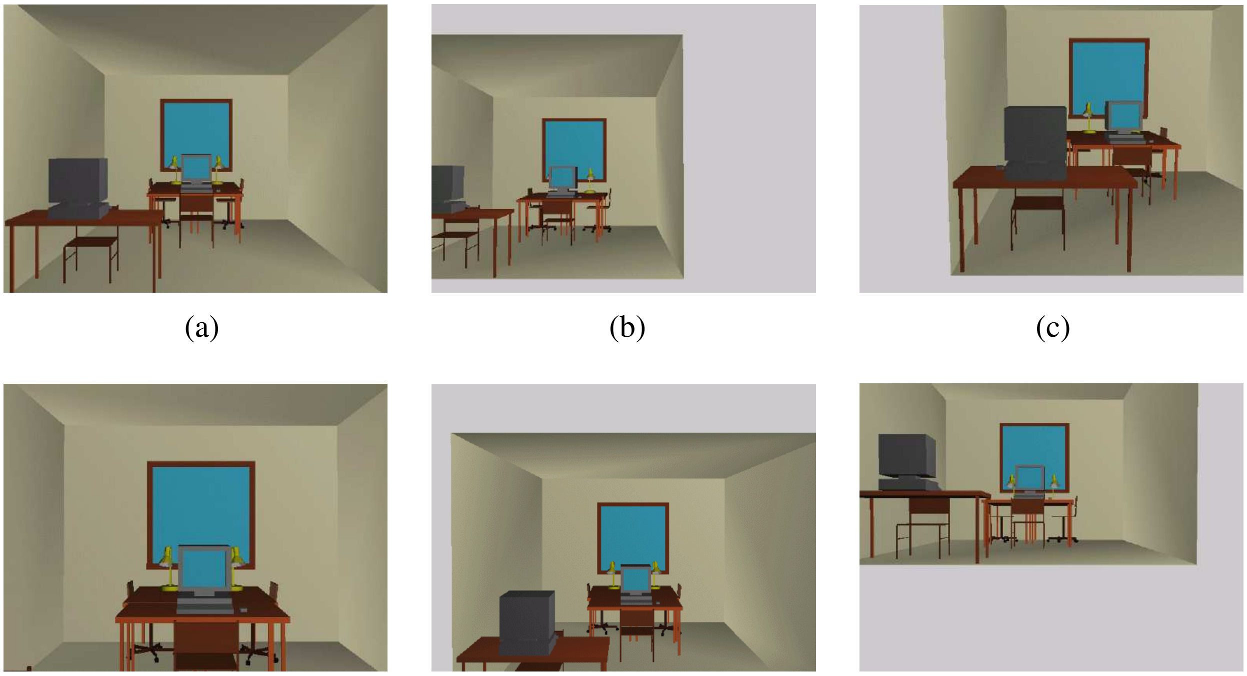
The speed of face (affine) tracking currently achieved is 4 frames per second on a DEC AlphaStation 600 with an image patch size of 98x80 pixels. The number of hierarchical levels is set to 3, and the number of iterations in image registration per level is also set to 3.



(a) (b) (c)

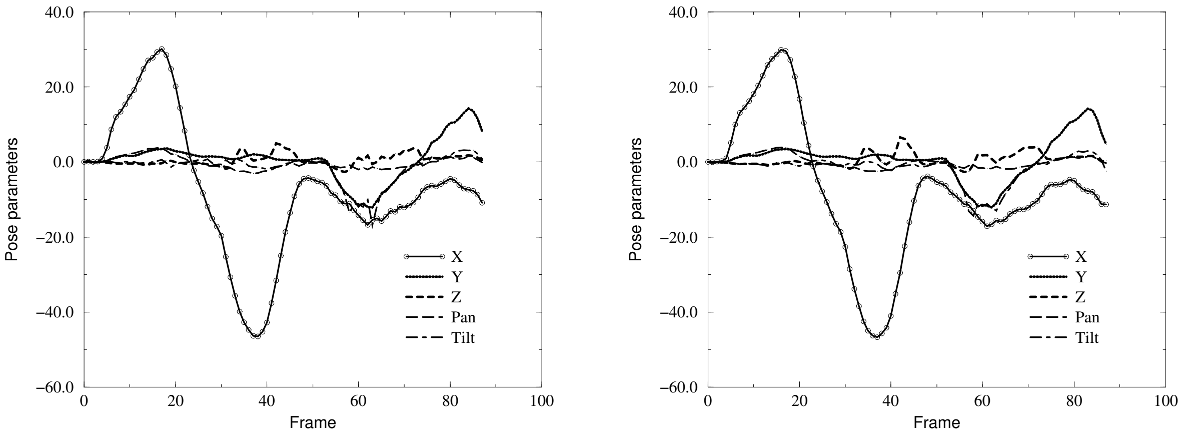
(d) (e) (f)

Figure 5: Face tracking results: (a) Reference face; (b) Face turned to the left of camera; (c) Face turned to the right of camera; (d) Face moved closer to the camera; (e) Face turned upwards; (f) Face turned downwards. The monochrome subimage of the face is the computed edge strength image.



(a) (b) (c) (d) (e) (f)

Figure 6: Response by viewer to changes in face pose (see Figure 5: (a) Initial view; (b) View rotated to the left; (c) View rotated to the right; (d) View moved closer; (e) View rotated upwards; (f) View rotated downwards.



(a) (b)

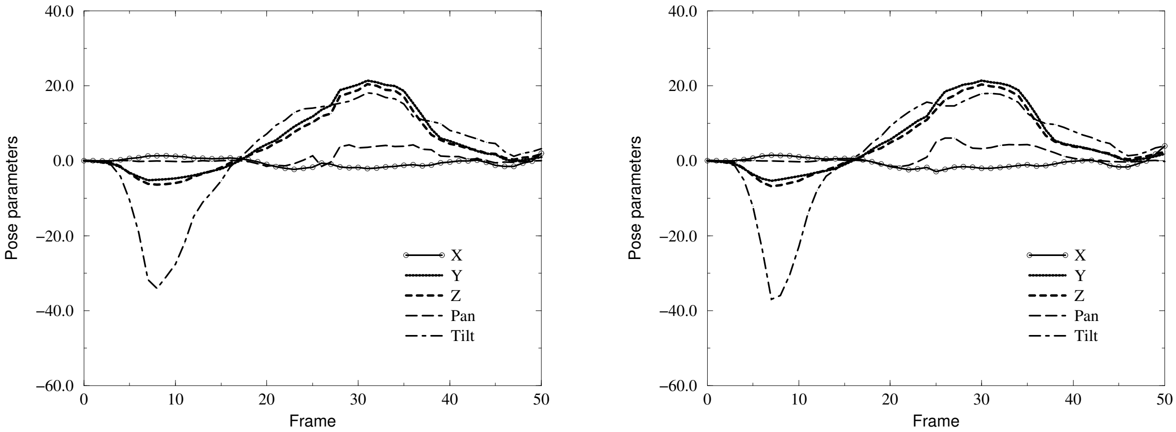
Figure 7: Pose parameter against time for head motion in predominantly X and then Y image plane directions. (a) and (b) are the unfiltered and filtered versions respectively.

Examples of plots of X, Y, Z, Pan, and Tilt against the frame number (both before filtering and after filtering) are shown in Figures 7-9. X and Y are given in pixels, Z is the scaled change in depth as given in (27), and Pan and Tilt are in degrees. As can been seen, the Kalman filter has the effect of smoothing the pose parameters.

7 Discussion

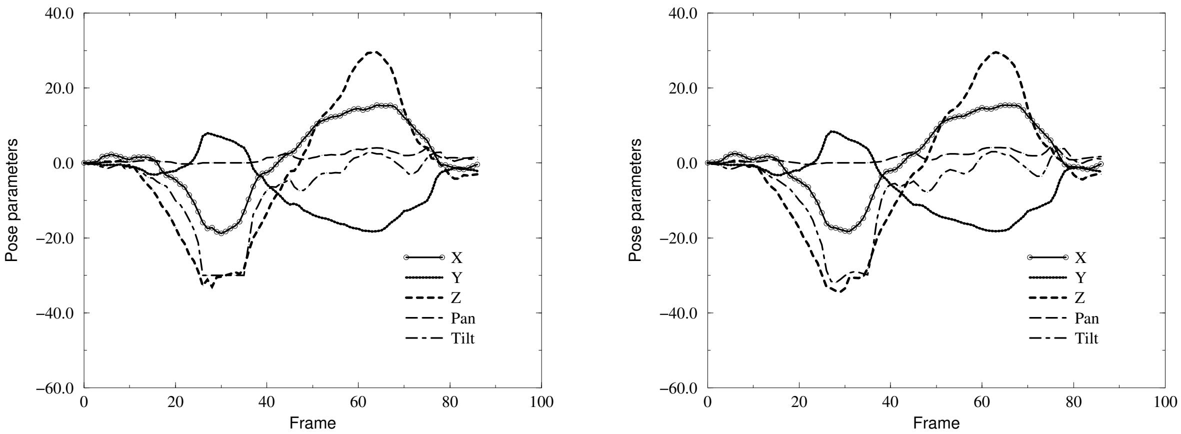
The face is obviously not planar, and the most robust approach of tracking the face would probably be to employ a properly initialized full 3-D face model on the image with a perspective camera model, as used by [3]. However, the face model initialization is not trivial, nor is it fast. The method described in this article sacrifices some accuracy for ease of initialization and speed of tracking by making the basic assumption that the face is far enough from the camera. In our application of navigating a virtual environment, absolute accuracy is not necessary. This distance assumption allows the scaled orthography camera model to be used, and the face to be assumed relatively planar.

Given the assumptions of significant distance from the camera and object planarity, there is also the issue of the camera model. Using 2-D full perspective to extract global image motion



(a) (b)

Figure 8: Pose parameter against time for predominantly head tilting motion. (a) and (b) are the unfiltered and filtered versions respectively.



(a) (b)

Figure 9: Pose parameter against time for head motion that includes moving towards and away from the camera. (a) and (b) are the unfiltered and filtered versions respectively.

is more appropriate but more unstable, while using pure 2-D rigid transform lacks the important parameters of scaling and skewing. As shown by our results, the 2-D affine model works well and is a good compromise.

Depending on the speed of head motion, the tracker pose output trajectory can be a little jerky. This is due to image noise and limited speed of convergence in image registration. To reduce the severity of this problem, we use the Kalman filter; it does help to produce a smoother trajectory of navigation.

8 Summary

In summary, the key features of our approach to “hands-off” navigation in virtual reality environments are:

- Only one camera is used, and its calibration is not necessary.

- Initialization is done by just taking a snapshot of the face in a neutral pose (facing directly ahead below the camera). This reference face snapshot is used to track the pose of the face.

- Tracking and pose determination is done using all the pixels in the image patch. This is considerably more robust than tracking specific small local features as is usually done [6, 8, 14]. To reduce dependency of the approach to illumination, we use the *edge strength* image rather than the direct intensity image. A comparative study of face recognition approaches seems to favor using templates as compared to using specific geometrical face features [2].

  At any given instant of time, the reference face image that has been taken during the initialization step is warped so as to minimize its intensity difference with the current face image. The warping transformation matrix is then decomposed to yield both position and orientation of the face. This is equivalent to deformable template matching with global motion.

- A geometric face model is *not* required. This is a major departure from most existing methods of tracking the face [3, 5, 6, 8, 14, 19]. As such, there are no complicated initialization steps required of this approach.

- Facial pose tracking is simplified by the observation that most people would usually not navigate a virtual environment by rotating about the viewing axis. The viewing camera is mounted in front of and above the user’s head at a tilted angle so as to be unobstrusive.

- The face, rather than the articulated hand, is being tracked; tracking the articulated hand and recognizing hand gestures is considerably more complicated and possibly error-prone.

We have also shown some results of virtual reality environment navigation by tracking the head.

Acknowledgments

The Kalman Filter code used in the system was implemented by Maria Loughlin.

References

[1] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland. Visually controlled graphics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):602–605, June 1993.

[2] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, October 1993.

[3] D. DeCarlo and D. Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pages 231–238, San Francisco, CA, June 1996. IEEE Computer Society.

[4] P. Ekman and W. V. Friesen. *Manual for the Facial Action Coding System*. Palo Alto: Consulting Psychologists, 1977.

[5] I. A. Essa and A. Pentland. A vision system for observing and extracting facial motion parameters. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 76–83, Seattle, Washington, 1994.

[6] A. Gee and R. Cipolla. Estimating gaze from a single view of a face. In *12th IAPR International Conference on Pattern Recognition*, volume I, pages 758–760, Jerusalem, Israel, 1994.

[7] A. Gelb, editor. *Applied Optimal Estimation*. The MIT Press, 1974.

[8] T. Horprasert, Y. Yacoob, and L. S. Davis. Computing 3-d head orientation from a monocular image sequence. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 242–247, Killington, VT, 1996.

[9] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *Proc.s IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 364–370, June 1996.

[10] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):545–555, June 1993.

[11] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[13] J. Rehg and A. Witkin. Visual tracking with deformation models. In *IEEE International Conference on Robotics and Automation*, pages 844–850, Sacramento, California, April 1991. IEEE Computer Society Press.

[14] H. Sako, M. Whitehouse, A. Smith, and A. Sutherland. Real-time facial-feature tracking based on matching techniques and its application. In *12th IAPR International Conference on Pattern Recognition*, volume II, pages 320–324, Jerusalem, Israel, 1994.

[15] L. S. Shapiro, A. P. Zisserman, and M. Brady. Motion from point matches using affine epipolar geometry. Technical Report OUEL 1994/93, University of Oxford, Robotics Research Group, 1993.

[16] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Workshop on Applications of Computer Vision (WACV'94)*, pages 44–53, Sarasota, Florida, December 1994. IEEE Computer Society.

[17] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 194–201, Seattle, Washington, June 1994. IEEE Computer Society.

[18] T. Takahashi and H. Ogata. Robotic assembly operation based on task-level teaching in virtual reality. In *IEEE International Conference on Robotics and Automation*, pages 1083–1088, Nice, France, 1992.

[19] T. Tsukamoto, C.-W. Lee, and S. Tsuji. Detection and pose estimation of human face with synthesized image models. In *12th IAPR International Conference on Pattern Recognition*, volume I, pages 754–757, Jerusalem, Israel, 1994.

[20] K. Waters, J. Rehg, M. Loughlin, S. B. Kang, and D. Terzopoulos. Visual sensing of humans for active public interfaces. In *Workshop on Computer Vision in Man-machine Interfaces*, Cambridge, UK, April 1996.

Hands-free navigation in VR environments by tracking the head

Sing Bing Kang

CRL 97/1

March 1997