ELSEVIER

PII: S0010-4485(97)00057-2

Computer-Aided Design, Vol. 30, No. 3, pp. 191–198, 1998

Printed in Great Britain
0010-4485/98/$19.00+0.00

Real time motion fairing with unit quaternions

Y C Fang†, C C Hsieh‡, M J Kim§, J J Chang‖ and T C Woo‖*

Though it may be tempting to smooth orientation data by filtering the Euler angles directly, it is noted that smoothed Euler angles do not necessarily yield a smooth motion. This is caused by the difference between the metric in the rotation group and that in the Euclidean space. The quaternions, which Hamilton discovered in 1853, provide a means for representing rotation. A unit quaternion, represented as a hypersphere in $\mathcal{R}^4$, has the same local topology and geometry as the rotation group. It thus provides a means for interpolating orientations. It is possible to achieve smooth rotation by filtering in quaternions the resulting quaternion may no longer be unitized. Fortunately, a unit quaternion curve, which represents the rotation path, can be derived by integrating the exponential map of the angular velocity. Unity of quaternions is thus maintained by filtering angular velocities. A lowpass filter coupled with an adaptive, mediative filter are employed to achieve smooth rotation motion in real time

Keywords: energy minimization, fairing, quaternion, virtual reality

INTRODUCTION

Noise reduction is essential to achieving high quality perception in a virtual reality (VR) environment. Excessive noise results in a ''tremor'' even though the operator stays still. This tremor distracts the operator. It also contributes to the so-called ''cyber sickness'' which is exacerbated by the high speed processors and the high resolution devices employed in VR. Smoothing the VR data is therefore of interest.

A smooth motion comprises two parts: a smooth translation and a smooth rotation. Noise reduction for the translation data is normally formulated as a curve-fairing problem and has received a considerable amount of success[1–3] in the computer aided design community. Progress in dealing with orientation data was impeded by the lack of a suitable representation for orientation. A 3×3 direction cosine matrix represents an orientation by a rotation about a given axis. However, due to the the dependence of the components in the matrix, direct interpolation of each component is futile. *Euler angles* are the three ordered, successive rotations about the three Cartesian axes[4]. It is feasible to interpolate the orientations through these Euler angles; however, in so doing, it suffers from the loss of one degree of freedom when two of the three axes become collinear (the so-called *gimbal lock*[5] effect). Furthermore, linear interpolation of the Euler angles does not provide a linear interpolation of the orientations[6]. An orientation can also be represented by a unit quaternion[7]. Quaternion techniques enable recent advances in the interpolation of orientations[6,8–13]. The unit quaternion space $Sp(1)$ in $\mathcal{R}^4$ has the same local topology and geometry as that of the rotation group $SO(3)$; it permits the linear interpolation between two orientations.

In the VR environment, a real time algorithm for smoothing is desirable. Smoothing by minimizing the integral of a weighted sum of derivatives of the entire curve is a standard technique[14]. But, real time performance can not be expected, though there have been reports on the noise reduction of motion data[15,16]. This is due to the nonlinear behavior of orientations in the Euclidean space. In ref.[17], a Kalman filter is applied to quaternions to achieve smooth orientations. The unity of quaternions after filtering is maintained by explicitly normalizing quaternions. The effect of normalization, however, is not explained.

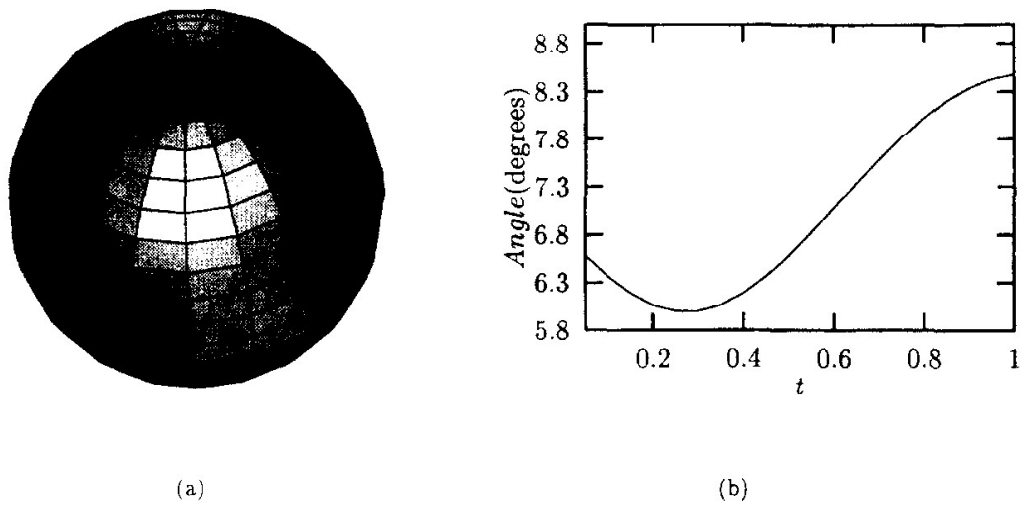
Two kinds of noise are generally encountered in the VR environment[17]: *static* noise, such as sensor noises, and *dynamic* noise, such as the system latency, for example, which only appear while a user is in motion. In this paper, a simple, yet efficient real time algorithm is proposed to deal with these noises in orientation data. The Euler angles, which are obtained directly from an input device, are first converted to unit quaternions. A compact derivation of the angular velocity from Euler angle is then presented. A lowpass filter removes the static error and an adaptive filter is applied to predict future motion to reduce the effect of system latency. Empirical data is provided to demonstrate the effectiveness of this algorithm. The strain energy, a criterion in evaluating the fairness of a curve, is used to compare the algorithm with others.

PRELIMINARY

Orientation

In a VR environment, the scene perceived by the operator through the display device is actively updated according to the changes in the operator's position and orientation. The position and orientation data, which are sampled from

*To whom correspondence should be addressed.
†Department of Industrial and Operations Engineering, University of Michigan, Michigan, USA
‡Department of Industrial Engineering and Management, Chao Yang University of Technology, Taiwan
§Computer Graphics Laboratory, System Engineering Research Institute, Taejon, Korea
‖Department of Industrial Engineering, University of Washington, Seattle, USA



(a) (b)

Figure 1 Nonlinearity in the interpolation of Euler angles: (a) Linear interpolation of two Euler angles, $(1\text{-}t)\boldsymbol{E}_a + t\boldsymbol{E}_b$; (b) Angles between consecutive orientations

certain input devices such as magnetic sensors, trackballs, data glove, etc. usually contain six components (t_x, t_y, t_z, ϕ_3, ϕ_2, ϕ_1). The first three elements* specify the Cartesian coordinates of the sensor, while the last three elements are the Euler angles for the current orientation. The three Euler angles: roll = ϕ_1, pitch = ϕ_2, and yaw = ϕ_3 are the rotation angles about the three orthogonal axes. Let E_z, E_y, E_x be the rotations about z-, y-, and x-axis, respectively, then

$$E_z = \begin{bmatrix} \cos\phi_3 & -\sin\phi_3 & 0 \\ \sin\phi_3 & \cos\phi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$E_y = \begin{bmatrix} \cos\phi_2 & 0 & \sin\phi_2 \\ 0 & 1 & 0 \\ -\sin\phi_2 & 0 & \cos\phi_2 \end{bmatrix}$$

$$E_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_1 & -\sin\phi_1 \\ 0 & \sin\phi_1 & \cos\phi_1 \end{bmatrix}. \quad (1)$$

The *orientation*, E_{zyx}, is the product of the E_i ($i = z, y, x$):

$$E_{zyx} = E_z E_y E_x = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}, \quad (2)$$

where,

$$e_{11} = \cos\phi_2\cos\phi_3$$

$$e_{12} = \sin\phi_1\sin\phi_2\cos\phi_3 - \cos\phi_1\sin\phi_3$$

$$e_{13} = \cos\phi_1\sin\phi_2\cos\phi_3 + \sin\phi_1\sin\phi_3$$

$$e_{21} = \cos\phi_2\sin\phi_3$$

$$e_{22} = \sin\phi_1\sin\phi_2\sin\phi_3 + \cos\phi_1\cos\phi_3$$

$$e_{23} = \cos\phi_1\sin\phi_2\sin\phi_3 - \sin\phi_1\cos\phi_3$$

$$e_{31} = -\sin\phi_2$$

$$e_{32} = \sin\phi_1\cos\phi_2$$

$$e_{33} = \cos\phi_1\cos\phi_2$$

Given two orientations $\boldsymbol{E}_a$ and $\boldsymbol{E}_b$, the in-betweens are found by interpolating the three Euler angles (ϕ_1, ϕ_2, and ϕ_3) independently. However, because of the nonlinearity in the rotation space, the intermediate orientations do not change linearly. *Figure 1a* shows the effect of linear interpolation between two Euler angles: $\boldsymbol{E}_a = (60°,60°,60°)$ and $\boldsymbol{E}_b = (180°,180°,180°)$ and *Figure 1b* shows the angles between consecutive orientations. It can be seen that the rotation angles are not uniformly spaced when the Euler angles are interpolated linearly.

Unit quaternions

A *quaternion* $\boldsymbol{Q}$, a four-dimensional complex number, consists of one real part and three imaginary parts:

$$\boldsymbol{Q} = w + x\hat{i} + y\hat{j} + z\hat{k}\ , w, x, y, z \epsilon \mathcal{R}, \quad (3)$$

which satisfies the multiplication rules[7,18]:

$$\hat{i}^2 = \hat{j}^2 = \hat{k}^2 = -1,$$

$$\hat{i}\hat{j} = \hat{k},\ \hat{j}\hat{i} = -\hat{k},$$

$$jk = i,\ kj = -i,$$

$$\hat{k}\hat{i} = \hat{j},\ \hat{i}\hat{k} = -\hat{j}.$$

The norm $|\boldsymbol{Q}|$ and the inverse $\boldsymbol{Q}^{-1}$ of a quaternion $\boldsymbol{Q}$ are defined as

$$|\boldsymbol{Q}| = w^2 + x^2 + y^2 + z^2$$

$$\boldsymbol{Q}^{-1} = (w - x\hat{i} - y\hat{j} - z\hat{k})/|\boldsymbol{Q}|.$$

A *quaternion* $\boldsymbol{Q}$ can also be represented as an ordered pair $\boldsymbol{Q} = (w, \vec{u})$, where $\vec{u} = (x, y, z) \epsilon \mathcal{R}^3$. Any vector $\vec{v} \epsilon \mathcal{R}^3$ can be easily mapped into the quaternion space:

$$\vec{v} \rightarrow (0, \vec{v}), \quad (4)$$

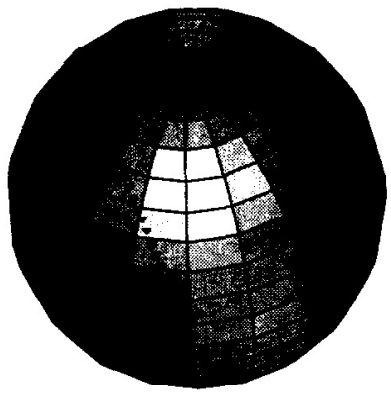
in the same spirit as the Cartesian coordinates are mapped into the homogeneous coordinates. A *unit quaternion*, $\boldsymbol{q}$, is a quaternion of the norm $|\boldsymbol{q}| = 1$, i.e.

$$w^2 + x^2 + y^2 + z^2 = 1.$$

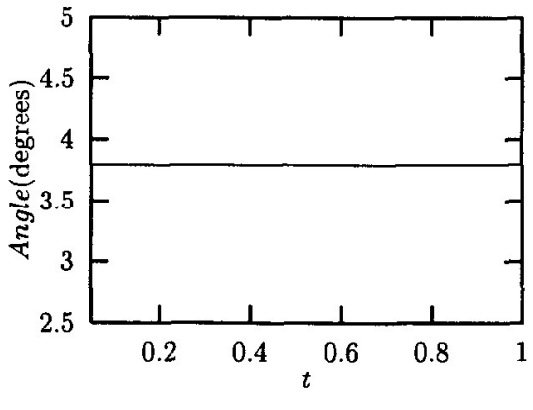
The entire set of unit quaternions has the same structure as a sphere in $\mathcal{R}^4$.

An orientation can be represented as a *rotation vector*

* While it is customary to denote the three Cartesian coordinates by (x,y,z) these symbols will be reserved for three of the four coordinates of a quaternion, as given in the next section.



(a)



(b)

Figure 2 Linearity in the interpolation of quaternions: (a) Linear interpolation of two quaternions, $q_a(q_b{}^{-1}q_b)^t$; (b) Angles between consecutive orientations

$\vec{v} = \theta\hat{\lambda}\epsilon\mathcal{R}^3$ which denotes a rotation $\theta = |\vec{v}|$ about the unit axis $\hat{\lambda} = \vec{v}/|\vec{v}|$. This orientation can also be represented as a *unit* quaternion $\boldsymbol{q}$:

$$\boldsymbol{q} = \left(\cos\frac{\theta}{2},\ \hat{\lambda}\sin\frac{\theta}{2}\right). \tag{5}$$

The mapping from a rotation vector $\vec{v}\epsilon\mathcal{R}^3$ to a unit quaternion $\boldsymbol{q}$ is derived through *quaternion exponential*[7], which is defined as:

$$\exp(\vec{v}) = \left(\cos\frac{\theta}{2}, \hat{\lambda}\sin\frac{\theta}{2}\right) \tag{6}$$

$$= \left(\cos\frac{|\vec{v}|}{2}, \frac{\vec{v}}{|\vec{v}|}\sin\frac{|\vec{v}|}{2}\right).$$

The inverse function of eqn (6), $\log \boldsymbol{q}$, is defined as:

$$\log \boldsymbol{q} = \log(w, \vec{u}) = \begin{cases} 2\tan^{-1}(|\vec{u}|/w)\ \dfrac{\vec{u}}{|\vec{u}|}, & \text{if } w \neq 0 \\ \pi\ \dfrac{\vec{u}}{|\vec{u}|}, & \text{if } w = 0 \end{cases}. \tag{7}$$

Given two orientations represented by unit quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$, it is possible to form a path connecting them by a rotation about a fixed axis. One such rotation paths is the *geodesic G*, also called the *slerp*[6], as enabled by the Euler principal rotation theorem[4], given by

$$G[\boldsymbol{q}_a, \boldsymbol{q}_b](t) = \boldsymbol{q}_a \exp(t\cdot\log(\boldsymbol{q}_a{}^{-1}\boldsymbol{q}_b)),\ 0 \le t \le 1 \tag{8}$$

$$= \boldsymbol{q}_a(\boldsymbol{q}_a{}^{-1}\boldsymbol{q}_b)^t,\ 0 \le t \le 1$$

where $\log(\boldsymbol{q}_a{}^{-1}\boldsymbol{q}_b)$ is the corresponding rotation axis. The geodesic is embedded in the distance metric for the unit quaternion space such that the distance between two unit quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ is defined by

$$\text{dist}(\boldsymbol{q}_a, \boldsymbol{q}_b) = |\log(\boldsymbol{q}_a{}^{-1}\boldsymbol{q}_b)|. \tag{9}$$

Here, the difference between the metric in the rotation space and that in the more familiar Euclidean space is noted.

Conversion between Euler angle and unit quaternion

An orientation can be represented by three ordered rotations about the three orthogonal axes. From eqns (1) and (5), the rotations about these axes are, in unit quaternions:

$$\boldsymbol{q}_{\text{yaw}} = \left(\cos\frac{\phi_3}{2}, \left\{0, 0, \sin\frac{\phi_3}{2}\right\}\right) \tag{10}$$

$$\boldsymbol{q}_{\text{pitch}} = \left(\cos\frac{\phi_2}{2}, \left\{0, \sin\frac{\phi_2}{2}0\right\}\right)$$

$$\boldsymbol{q}_{\text{roll}} = \left(\cos\frac{\phi_1}{2}, \left\{\sin\frac{\phi_1}{2}0, 0\right\}\right).$$

eqn (2) can equivalently be represented by a unit quaternion $\boldsymbol{q}_{\text{Euler}}$:

$$\boldsymbol{q}_{\text{Euler}} = \boldsymbol{q}_{\text{yaw}}\boldsymbol{q}_{\text{pitch}}\boldsymbol{q}_{\text{roll}} = (\mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}), \tag{11}$$

where

$$\mathcal{W} = \cos\frac{\phi_1}{2}\cos\frac{\phi_2}{2}\cos\frac{\phi_3}{2} + \sin\frac{\phi_1}{2}\sin\frac{\phi_2}{2}\sin\frac{\phi_3}{2}$$

$$\mathcal{X} = \sin\frac{\phi_1}{2}\cos\frac{\phi_2}{2}\cos\frac{\phi_3}{2} - \sin\frac{\phi_1}{2}\sin\frac{\phi_2}{2}\sin\frac{\phi_3}{2}$$

$$\mathcal{Y} = \cos\frac{\phi_1}{2}\sin\frac{\phi_2}{2}\cos\frac{\phi_3}{2} + \sin\frac{\phi_1}{2}\cos\frac{\phi_2}{2}\sin\frac{\phi_3}{2}$$

$$\mathcal{Z} = \cos\frac{\phi_1}{2}\cos\frac{\phi_2}{2}\sin\frac{\phi_3}{2} - \sin\frac{\phi_1}{2}\sin\frac{\phi_2}{2}\cos\frac{\phi_3}{2}.$$

The two end points $\boldsymbol{E}_a$ and $\boldsymbol{E}_b$, as in the *Figure 1*, can be converted to quaternions $\boldsymbol{q}_a$ = {0.774519, 0.158494, 0.591506, 0.158494} and $\boldsymbol{q}_b$ = {1,0,0,0}, respectively. *Figure 2* shows the interpolation between $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ using eqn (8). It can be seen that the rotation angles between consecutive orientations are uniform.

ANGULAR VELOCITY AND STRAIN ENERGY

The smoothness of a curve is subjective. Given n points, the ''smoothest'' curve may be a straight line passing through possibly none of them. Indeed, many criteria have been proposed over the years. Here, the smoothness of a curve is measured by its strain energy, i.e. the integral of its curvature. Let $p(s)$ be a space curve; its curvature is defined as

$$\kappa(s) = |p''(s)|,$$

where s is the arc length. Its strain energy is thus given by

$$\varepsilon(p) = \int_p \kappa(s)^2 \mathrm{d}s. \tag{12}$$

It is natural to extend the concept of minimal energy to the smoothing of the unit quaternion curves. Indeed, the strain energy of a unit quaternion curve can be measured by the change in the angular velocity or *angular acceleration*. However, the angular acceleration is no longer the second

derivative of a unit quaternion curve. In the two following subsections, a compact derivation of the angular velocity through the Euler angles is presented. The flaw in adopting eqn (12) is explained and the correct representation of the strain energy for a unit quaternion curve is given.

Angular velocity

Unlike in the Euclidean space, in which the velocity $v(t)$ of a space curve $\boldsymbol{p}(t)$ is its first derivative, the *angular velocity* $\omega(t)$ of a unit quaternion curve $\boldsymbol{q}(t)$ takes on a nonlinear form[19]:

$$\omega(t) = 2\boldsymbol{q}^{-1}(t)\boldsymbol{q}'(t). \tag{13}$$

Conversely, given the angular velocity $\omega(t)$, the corresponding unit quaternion curve $\boldsymbol{q}(t)$ is derived by quaternion integration[7,20]:

$$\boldsymbol{q}(t) = \boldsymbol{q}_0 \prod_0^t \exp(\omega(t)\mathrm{d}t). \tag{14}$$

where $\boldsymbol{q}_0$ is a unit quaternion which represents the initial orientation.

From eqn (13), the angular velocity $\omega(t)$ can be derived as

$$\begin{aligned}\omega(t) &= 2\boldsymbol{q}^{-1}_{\text{Euler}}(t)\boldsymbol{q}'_{\text{Euler}}(t)\\ &= 2(\mathcal{W}, -\mathcal{X}, -\mathcal{Y}, -\mathcal{Z})\\ &\quad \times (\mathcal{W}', \mathcal{X}', \mathcal{Y}', \mathcal{Z}')\\ &= (\omega_0, \omega_1, \omega_2, \omega_3),\end{aligned} \tag{15}$$

where

$$\omega_0 = 0,$$

$$\omega_1 = \phi_1' - \phi_3'\sin\phi_2,$$

$$\omega_2 = \phi_2'\cos\phi_1 + \phi_3'\sin\phi_1\cos\phi_2,$$

$$\omega_3 = -\phi_2'\sin\phi_1 + \phi_3'\cos\phi_1\cos\phi_2.$$

Since ω_0 equals zero, the angular velocity can be rewritten as a vector

$$\omega(t) = (\omega_1, \omega_2, \omega_3) \tag{16}$$

For the discrete case, eqn (16) becomes

$$\omega_i = (\omega_{1i}, \omega_{2i}, \omega_{3i}) \tag{17}$$

where

$$\omega_{1i} = \frac{\phi_{1(i+1)} - \phi_{1i}}{\Delta t} - \frac{\phi_{3(i+1)} - \phi_{3i}}{\Delta t}\sin\phi_{2i}$$

$$\omega_{2i} = \frac{\phi_{2(i+1)} - \phi_{2i}}{\Delta t}\cos\phi_{1i} + \frac{\phi_{3(i+1)} - \phi_{3i}}{\Delta t}\sin\phi_{1i}\cos\phi_{2i}$$

$$\omega_{3i} = -\frac{\phi_{2(i+1)} - \phi_{2i}}{\Delta t}\sin\phi_{1i} + \frac{\phi_{3(i+1)} - \phi_{3i}}{\Delta t}\cos\phi_{1i}\cos\phi_{2i}$$

Δt : sampling interval.

Energy function

Recall that the strain energy for a space curve can be expressed as the integral of its second derivative, viz. Eqn (12). By analogy, the smoothness of a unit quaternion curve $\boldsymbol{q}(t)$ may be expressed in terms of angular acceleration.

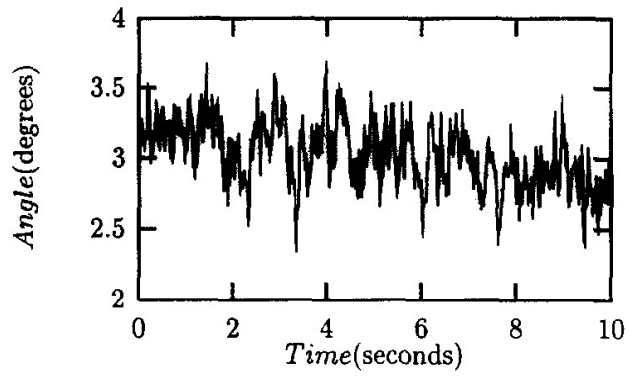


Figure 3 Signal noise

However, the first derivative of a unit quaternion curve is *not* the angular velocity, nor is the second derivative of the quaternion curve the angular acceleration. The energy function $\int_q |q''(t)|^2 \mathrm{d}t$, directly derived from eqn (12), therefore, is not a good measure[14,21].

A more appropriate energy function is obtained by directly integrating the norm of the derivative of the angular velocity[14]:

$$\varepsilon(\boldsymbol{q}) = \int_q |\omega'(t)|^2 \mathrm{d}t. \tag{18}$$

It is clear that $\varepsilon(\boldsymbol{q}) = 0$ if the angular velocity is constant. From eqn (15), it can be seen that for $\omega(t)$ to be constant, ϕ_1 and ϕ_2 need to be constant and ϕ_3 a function of t with an order of at most one. By eqn (11), this results in a unit quaternion curve which is a great circle on the unit hypersphere. For the discrete case, the strain energy is estimated by

$$\sum_{i=1}^{n-2} |\omega_i'|^2 \tag{19}$$

where the first derivative of the angular velocity $\omega'(t)$ can be easily derived from eqn (17):

$$\begin{aligned}\omega_i' &= (\omega_{1i}', \omega_{2i}', \omega_{3i}',)\\ &= \left(\frac{\omega_{1(i+1)} - \omega_{1i}}{\Delta t}, \frac{\omega_{2(i+1)} - \omega_{2i}}{\Delta t}, \frac{\omega_{3(i+1)} - \omega_{3i}}{\Delta t}\right)\end{aligned} \tag{20}$$

FILTERING

Noise removal is normally dealt with by filtering. In this section, a lowpass filter is used to remove sensor noise and an adaptive linear predictor is employed to reduce the noise due to system latency.

Lowpass filter

Figure 3 shows the *pitch* sampled from a fast track† when it is placed on a table. Even though the sensor remains stationary, the rotation angle varies by as much as 1.5°.

A lowpass filter can be implemented by convolving the output signal with a response function. Let s_κ be sampled signal in the time domain. In the frequency domain, the Fourier transform gives:

$$S_n = \sum_{k=0}^{N-1} s_k e^{2\pi i k n/N},$$

† A 3D magnetic sensor built by Polhemus Inc.

where N is the total number of samples. Conversely, the discrete signal S_n in the frequency domain can be convert to the time domain by the inverse Fourier transform:

$$S_k = \frac{1}{N}\sum_{k=0}^{N-1} s_n e^{2\pi ikn/N},$$

Let r_j be a response function, which serves as a lowpass filter, with a duration of N samples. The filtered signal $\hat{s}_k$ can be obtained by finding the convolution of s_k and r_j in the time domain[22],

$$\hat{s}_k = (r \times s)_k \equiv \sum_{j=-N/2+1}^{N/2} s_{k-j}\, r_j \tag{21}$$

or in frequency domain,

$$\hat{s}_k = (r \times s)_k \equiv SCF^{-1}(S_n R_n), \tag{22}$$

where R_n is the discrete Fourier transform of r_j, and $\mathcal{F}^{-1}$ is the inverse Fourier transform operator.

Adaptive filter

Other than sensor noise, perception lag is an important factor which affects the visual quality in a VR environment. Perception lag is caused by the latency inherent in such system devices as the tracking sensor and the image generation hardware. To minimize latency, a forward prediction filter can be applied. Linear predictors, such as the Wiener filter, work well for *stationary* signals which only involve white noise. However, if the correlation between noise and signal is high, other alternatives maybe required[23]. Some higher order predictors, such as the Kalman filter, have been applied in refs[16,17,24]. However, the parameters for the higher order filter are hard to obtain[16]. In addition, although a higher order filter gives more accurate prediction, it involves the risk of further amplifying the sensor noise[25].

An adaptive linear predictor is used here. Due to the recursive nature of adaptive filtering, changes in the incoming signal are taken into account, thus producing rather accurate results.

Let $\hat{s}^1, \hat{s}^2, \ldots, \hat{s}^n$ represent n samples of the lowpass filtered signal. The signal $\hat{s}^{n+1}$ estimated by forward linear prediction of order m is

$$\hat{s}^{n+1} = \sum_{k=1}^{m} a_m^k \hat{s}^{n-k+1} \tag{23}$$

where a_m^k, $k = 1, \ldots, m$ are prediction coefficients. The stability of the prediction coefficients is important. For an inappropriately chosen a_m^k, the filter can have an exponential growth. A stable estimation method is due to Burg[26]. It has the property that the prediction coefficients estimated are guaranteed to be less than one, thus preventing the undesirable growth[27]. The algorithm of finding a_m^k using the Burg method is given as follows:

Initialization
Calculate reflection coefficient k^1 and the initial forward (e_f) and backward (e_b) errors:

$$k_1 = \frac{2\sum_{j=2}^{m} x^j x^{j-1}}{\sum_{j=2}^{m} |x^j|^2 + |x^{j-1}|^2}, \quad e_{f0}^j = x^j, \quad e_{b0}^i = x^{j-1}.$$

History
For $1 \le i \le n$

(1) Calculate the prediction errors, for $i + 1 \le j \le m$

$$e_{fi}^j = e_{f(i-1)}^j - k_i e_{b(i-1)}^{j-1},$$

$$e_{bi}^j = e_{b(i-1)}^{j-1} - k_i e_{f(i-1)}^j.$$

(2) Calculate the reflection coefficients k_i by

$$k_i = \frac{2\sum_{j=i+1}^{m} e_{f(i-1)}^j e_{b(i-1)}^{j-1}}{\sum_{j=i+1}^{m} |e_{f(i-1)}^j|^2 + |e_{b(i-1)}^{j-1}|^2}.$$

(3) Calculate the components of prediction coefficients by

$$d_{ii} = k_i.$$

For $1 \le j \le i-1$,

$$d_{ji} = d_{j(i-1)} - k_i d_{(i-j)(i-1)}.$$

Prediction
Calculate the prediction coefficients, for $1 \le j \le m$

$$a_j = \sum_{i=1}^{n} d_{ji}$$

The reflection coefficients k_i is updated

$$k_i^{n+1} = k_i^n + \frac{1}{D_i^{n+1}}[e_{fi}^{n+1} e_{b(i-1)}^n + e_{f(i-1)}^{n+1} e_{bi}^{n+1}],$$

where

$$D_i^{n+1} = D_i^n + |e_{f(i-1)}^{n+1}|^2 + |e_{b(i-1)}^n|^2.$$

Data smoothing

It is tempting to apply the filters immediately. Let $\mathcal{L}$ and $\mathcal{A}$ be the lowpass filter and the adaptive prediction, respectively. One could formulate the filtering process as

$$\tilde{q}(t) = \mathcal{A}\{\mathcal{L}[q(t)]\},$$

Unfortunately, $\tilde{q}(t)$ thus obtained is not necessarily a *unit* quaternion curve.

As direct filtering on a unit quaternion curve is not feasible, the filtering on the *angular velocity* of the unit quaternion curve is examined. For a quaternion curve to be smooth, it is necessary that the changes in its angular velocity be "small". Let $\omega(t)$ be the angular velocity; the filtering on the angular velocity is written as

$$\tilde{\omega}(t) = \mathcal{A}\{\mathcal{L}[\omega(t)]\}, \tag{24}$$

Recall from eqn (14) that a unit quaternion can be obtained by integrating over the given angular velocities. A smooth unit quaternion curve can then be represented as

$$\tilde{q}(t) = q_0 \prod_0^t \exp(\tilde{\omega}(t)\mathrm{d}t). \tag{25}$$

Since the multiplication of the unit quaternions is still a unit quaternion[6], the result of eqn (25) is guaranteed to be a unit

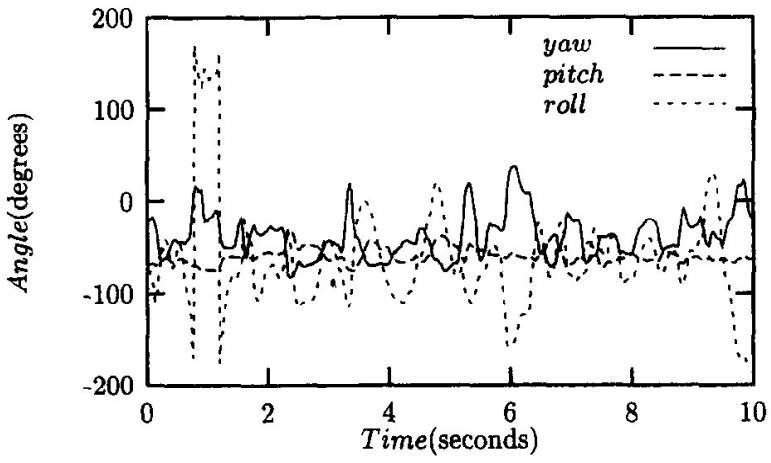


Figure 4 Original sampled data (yaw, pitch, roll)

quaternion curve. For discrete data, eqn (25) can be rewritten as

$$\tilde{q}_i = q_0 \prod_{i=1}^{n-1} \exp(\tilde{\omega}_i \Delta t). \tag{26}$$

IMPLEMENTATION

Figure 4 shows a set of yaw, pitch, and roll angles sampled from a fasttrack at 60 Hz. As ''smooth'' as the angles in *Figure 4* look, their first derivatives (or angular velocities ω_i) are not. *Figure 5a* shows ω_1, the x component of the angular velocity. With this data, we intentionally take the ''incorrect'' approach by filtering the Euler angles. The (x component of the) angular velocity thus obtained is denoted by $\underline{\omega}_i$. With the same data, we apply $\mathcal{A}$ and $\mathcal{L}$ to the angular velocity ω_i and obtain $\tilde{\omega}_i$ *Figure 5b* contrasts the x components: $\underline{\omega}_1$ (incorrect, dashed) and $\tilde{\omega}_1$ (proposed, solid).

Table 1 contrasts $\underline{\omega}$ to $\tilde{\omega}$ from the point of view of the strain energy in eqn (18). ε denotes the strain energy before the filtering; $\underline{\varepsilon}$ and $\tilde{\varepsilon}$ the strain energy after the filtering, but with the following difference: $\underline{\varepsilon}$ is obtained by taking the first derivative of the angular velocity $\underline{\omega}$ by linear interpolation of the Euler angles; $\tilde{\varepsilon}$ on the other hand, is obtained from $\tilde{\omega}$. Three kinds of motions (slow, medium, and quick) are sampled, in addition to the equipment being stationary. In the stationary mode, the improvement of $\tilde{\varepsilon}$ over $\underline{\varepsilon}$ is about 6.2%. (The lower the energy, the smoother the trajectory is.) The improvement is more dramatic for a ''slow'' motion:

Table 1 Strain energy comparison for different filtering schemes

Motion	ε	$\underline{\varepsilon}$	$\tilde{\varepsilon}$
Stationary	249.49	7.10	6.37
Slow	4967.72	807.11	183.33
Medium	35876.11	3386.08	930.75
Quick	74041.90	3761.85	1831.96

the user wearing the equipment moves about at approximating 1 ft/sec, rotating at around 5°/sec. The strain energy before the filtering is 4967.72; drops to 807.11 for filtered Euler angles and 183.33 for filtered angular velocities. The energy becomes necessarily higher as the motion progresses from ''slow'' to ''quick''; the signal correspondingly becomes less distinguishable from noise. Nevertheless, comparing $\underline{\varepsilon}$ to $\tilde{\varepsilon}$, the last two rows of *Table 1* indicate a 3:1 reduction in energy (hence a corresponding increase in smoothness).

The rotation path generated by the various schemes are shown and compared in *Figure 6*. *Figure 6a* is the rotation path converted directly from sampled Euler angles. *Figure 6c* results from filtering the Euler angles. To be noted is that the path in c deviates from the original a. The result of filtering on the quaternions and then normalizing to maintain unity is shown in *Figure 6e*. It retains the geometry of the intended path more faithfully, yet ''noise'' is still present. *Figure 6g* shows the the result of filtering in angular velocities. As the path is ''smooth'', fidelity to the original data is high.

SUMMARY

A rotation smoothing technique has been presented in this paper. It provides an efficient way of achieving a smooth rotation without involving a non-linear optimization which may not be achievable in real time applications such as that for noise filtering of VR equipment.

ACKNOWLEDGEMENTS

Raw data in *Figures 3–5* was provided by Dr. Woodrow Barfield, University of Washington. Insightful comments by the reviewers are gratefully acknowledged.

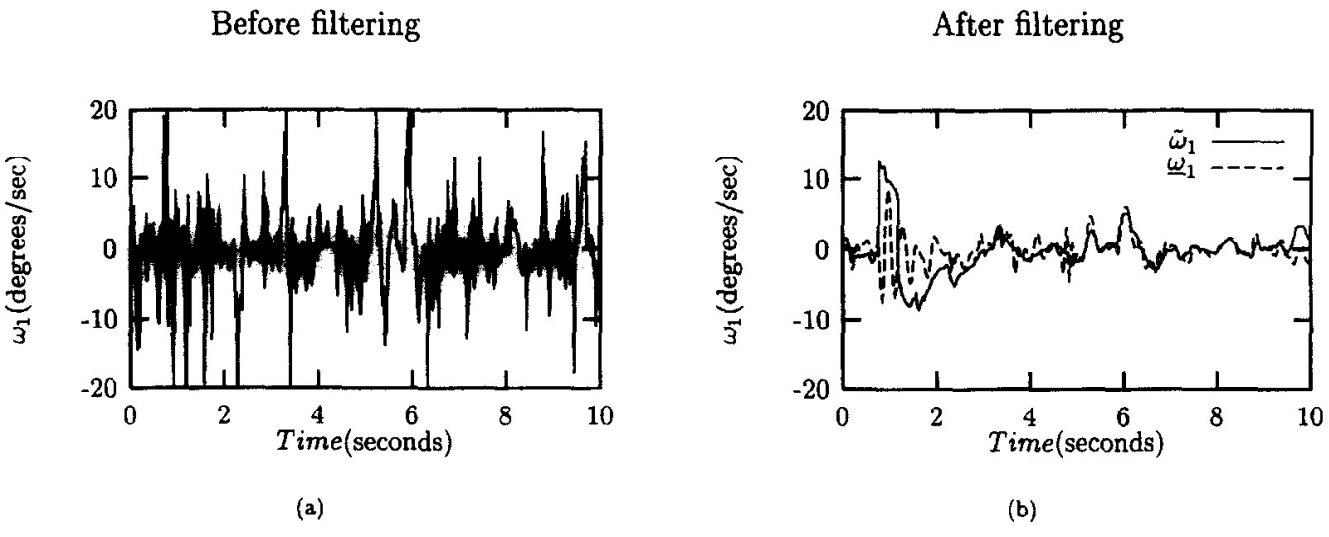


Figure 5 Empirical data: (a) x component of the angular velocity vector; (b)$\tilde{\omega}_1$ and $\underline{\omega}_1$

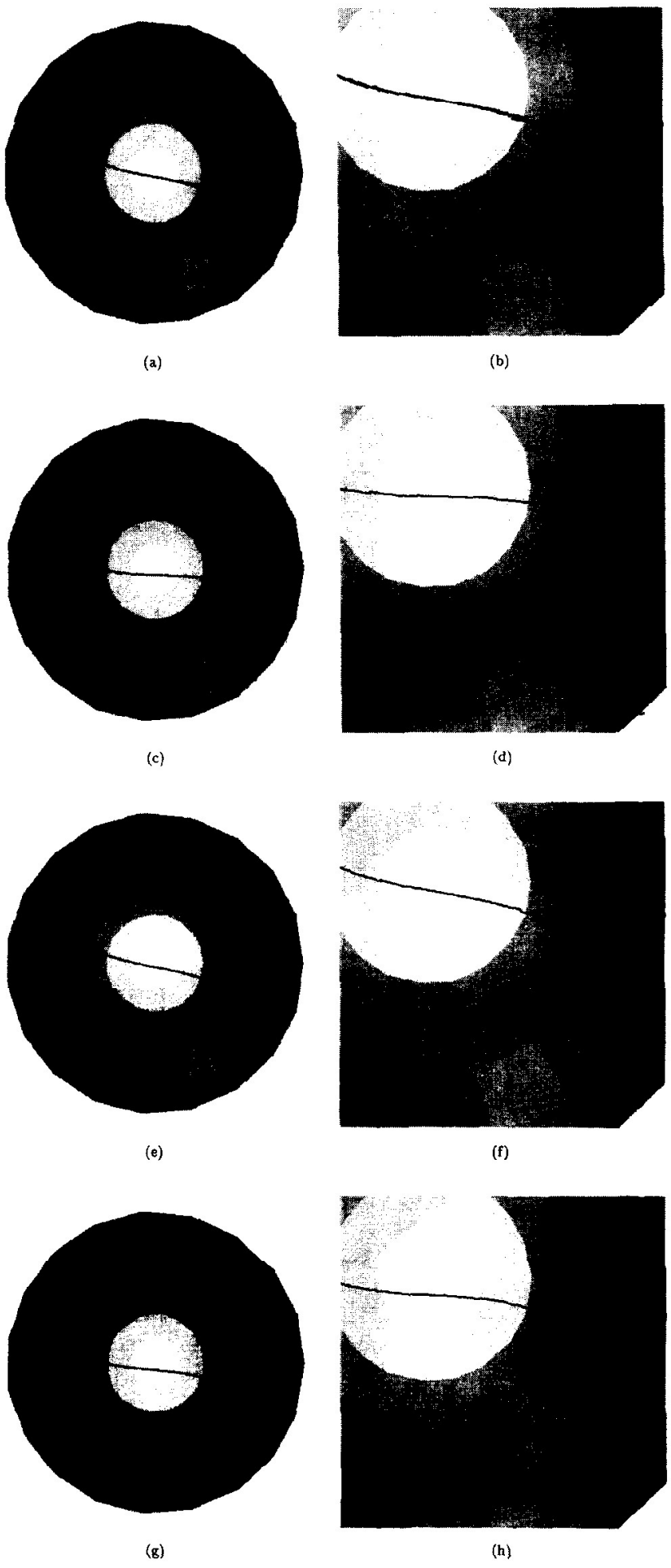


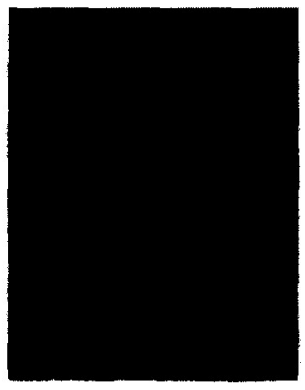
Figure 6 Comparison of rotation path between different filtering schemes: (a) Sampled data; (b) Magnification of (a); (c) Filtered Euler angles; (d) Magnification of (c); (e) Filtered in quaternions and then normalized; (f) Magnification of (e); (g) Filtered angular velocity; (h) Magnification of (g)

REFERENCES

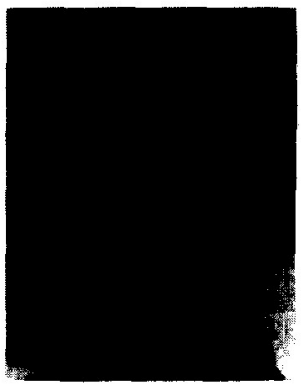
1. Choi. B., *Geometric Modeling for CAD/CAM*. Elsevier Science Publishing, New York, 1991.
2. Farin, G., Rein, G., Sapidis, N. and Worsey, A., Fairing cubic B-spline curves. *Computer Aided Geometric Design*, 1987, **4**, 91–103.
3. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 2nd edn., Academic Press, New York, 1990.
4. Euler, L., Du movement de rotation des corps solides autour d'un axe variable. In: *Opera Omnia*, Ser. secunda, vol. 8, Orell Flisli Thriei, Lausanne, 1758.
5. Kane, T., Likins, P. and Levinson, D., *Spacecraft Dynamics*. McGraw–Hill, New York, 1983.
6. Shoemake, K., Animating rotation with quaternion curves. In *Proceedings SIGGRAPH '85*, vol. 19, 1985, pp. 245–254.
7. Hamilton, W., *Elements of Quaternions*, Vol. I-II. Chelsea Publishing Company, Chelsea, 1969.
8. Juttler, B., Visualization of moving objects using dual quaternion curves. *Computer and Graphics*, 1994, **18**(3), 315–326.
9. Kim, M. J., Kim, M.S., Shin, S., A C^2-continuous B-spline quaternion curve interpolating a given sequence of solid orientations, In *Proceedings of Computer Animation '95*, Geneva, Switzerland, 1995, pp. 72–81.
10. Kim, M., Hsieh, C., Wang, M., Wang, C., Fang, Y. and Woo, T.C., Noise smoothing for VR equipment in the quaternion space. In *Proceedings of the Symposium on Virtual Reality in Manufacturing Research and Education*, Chicago, October 1996.
11. Kim, M., Nam, K., Interpolating solid orientations with circular blending quaternion curves, In *Proceedings of the Communicating with Virtual Worlds*, Lausanne, Switzerland, 1993, pp. 258–69.
12. Lake, R., Green, M., Dynamic motion control of an articulated figure using quaternion curves, In *Proceedings of the Second International Conference on Computer-Aided Design and Computer Graphics*, Hangzhou, China, 1991, pp. 37–44.
13. Nielson, G. and Heiland, R., Animated rotations using quaternions and splines on a 4D sphere. *Programming and Computer Software*, 1992, **18**(4), 145–154.
14. Lee, J., Shin, Y., Motion Fairing, In *Proceedings of Computer Animation 1996*, Geneva, Switzerland, 1996, pp. 136–143.
15. Friedmann, M., Starner, T. and Pentland, A., Synchronization in Virtual Realities. *Presence*, 1992, **1**(1), 139–144.
16. Srinvas, K. and Reddy, V., Interlaced Kalman filtering of 3D angular motion based on Euler's nonlinear equations. *IEEE Transactions on Aerospace and Electronic Systems*, 1994, **30 No 1**, 174–184.
17. Azuma, R., Bishop, G., Improving static and dynamic registration in an optical seethrough HMD, In *Proceedings SISSGRAPG '94*, vol. 28, 1994, pp. 197–204.
18. Curtis, M., *Matrix Groups*. Springer, 1972.
19. Junkins, J. and Turner, J., *Optimal Spacecraft Rotational Manueuvers*. Elsevier, 1986.
20. Kim, M.J., Kim, M.S., Shin, S., A general construction scheme for unit quaternion curves with simple high order derivatives, In *Proceedings SISSGRAPH '95*, vol. 29, 1995, pp. 369–376.
21. Barr, A., Currin, B., Gabriel, S., Hughes, J., Smooth interpolation of orientations with angular velocity constraints, In *Proceedings SISSGRAPH '92*, vol. 26, 1992, pp. 313–320.
22. Press, W., Teukolsky, S., Vetterling, W. and Flannery, B., *Numerical Recipes in C*. Cambridge University Press, 1992.
23. Ghazisaedy, M., Adamczyk, D., Sandin, D., Kenyon, R., Defanti, T., Ultrasonic calibration of magnetic tracker in a virtual reality space, In *Proceedings VRAIS '95*, 1995, pp. 179–188.
24. Azuma, R., Bishop, G., A frequency-domain analysis of head–motion prediction, In *Proceedings SISSGRAPH '95*, vol. 29, 1995, pp. 401–408.
25. Deering, M., High resolution virtual reality, In *Proceedings SISSGRAPH '92*, vol. 26, 1992, pp. 195–202.
26. Burg, J., Maximum entropy spectral analysis. PhD Thesis, Standford University, 1975.
27. Bellanger, M., *Adaptive Filters and Signal Analysis*. Marcel Dekker, New York, 1987.

Ying-Che Fang is currently a PhD candidate in the Department of Industrial and Operations Engineering at the Univesity of Michigan. He received his BS degiee in civil engineering and management science from National Chillo Tung Universy at Hsinshu, Taiwan in 1989 and his MS degree in manufacturing engineering from Syracuse University in 1993. His primary research interests are in the areas of CAD/CAM computational geometry, and geometric modeling.

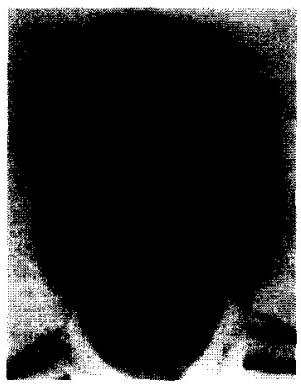
Chung-Chi Hsieh is an assistant professor at ChaoYang University of Technology. He received a BS in industrial engineering and a BS in electrical engineering from National Tsing Hua University (Taiwan) in 1990. He completed his Ph.D. in Industrial Operations Engineering at University of Michigan in 1997. His research interests are in computational geometry, computer graphics, tolerancing, and inspection in manufacturing processes.



Jen-Chien J. Chang is currently a graduate student in the Department of Industrial Engineering at the University of Washington. He received his BS degree in computer science and mathematics from the University of Washington in 1995. His research interests are in CAD/CAM, computer graphics and force feedback design in ergonomics.



Myoung-Jun Kim is with the Computer Graphics Laboratory, Systems Engineering Research institute, in Taejon, Korea. After receiving his PhD in computer science from the Korea Advanced Institute of Science and Technology in 1996, he spent a year at the University of Washington. There, he was instrumental in two projects: Beaconing BarCode and Virtual Holography. Dr. Kim has broad interests beyond quaternions, geometry, and graphics. He has extensive industrial experience with tele-communication software. He can operate machine tools from mould and die making. His hobbies include synthetic musical instruments and boating.



Tony C. Woo is professor of industrial engineering, adjunct professor of mechanical engineering, and holder of the John M. Fluke Distinguished Chair in Manufacturing Engineering at the University of Washington. His prior associations were with the University of Michigan and the National Science Foundation. His research and teaching is in computational geometry for design and manufacturing. He received his education in electrical engineering at the University of Illinois.