Head Orientation Prediction: Delta Quaternions Versus Quaternions

Henry Himberg and Yuichi Motai, *Member, IEEE*

***Abstract*—Display lag in simulation environments with helmet-mounted displays causes a loss of immersion that degrades the value of virtual/augmented reality training simulators. Simulators use predictive tracking to compensate for display lag, preparing display updates based on the anticipated head motion. This paper proposes a new method for predicting head orientation using a delta quaternion (DQ)-based extended Kalman filter (EKF) and compares the performance to a quaternion EKF. The proposed framework operates on the change in quaternion between consecutive data frames (the DQ), which avoids the heavy computational burden of the quaternion motion equation. Head velocity is estimated from the DQ by an EKF and then used to predict future head orientation. We have tested the new framework with captured head motion data and compared it with the computationally expensive quaternion filter. Experimental results indicate that the proposed DQ method provides the accuracy of the quaternion method without the heavy computational burden.**

***Index Terms*—Compensation, Kalman filtering, prediction methods, tracking, virtual reality.**

I. Introduction

HEAD tracking is widely used in augmented and virtual reality (AR/VR) simulation environments to control scene rendering in response to head orientation. The perceived latency (lag) between head motion and display response causes a loss of immersion for the user that can result in dizziness in extreme cases [2], [8]–[10], [14], [15], [23]–[25]. In training applications, the user learns to compensate for the display latency of the particular simulator, adjusting head motion to improve performance. This learned behavior compensates for display latency in the simulation environment, but differences between the simulator latency and that of the actual system reduce the value of the training. An effective method of compensating for simulation latency in helmet-mounted display (HMD) simulators is to predict the future orientation of the head. If the head orientation can accurately be predicted, then the simulator can render the next scene before the user moves. Various prediction methods have been proposed for latency compensation [1]–[5], [7], [12], [29], [31], [34], [35], with the Kalman filter receiving considerable attention.

Predictive filtering, including the extended Kalman filter (EKF), particle filters (PFs), and the unscented Kalman filter (UKF), is widely used for latency compensation [1], [2], [5], [7], [26], [33]. The UKF requires additional computation resource without improving performance when compared to the EKF in estimating the quaternion motion [27]. The PF does not provide a significant improvement upon the EKF when used for head motion prediction [7]. We use the EKF in our study of head motion prediction to avoid the additional computational burden of other methods [27], [28].

This paper proposes the delta quaternion (DQ) framework for latency compensation. The DQ framework predicts future head orientation from the change in quaternion orientation between measurements (the DQ). Angular head velocity is estimated from the DQ by an EKF and then combined with the current quaternion measurement to predict future orientation. The DQ differs from other head orientation prediction methods in several ways, including estimation of the DQ instead of the quaternion orientation in the EKF, and decoupling of the prediction interval from the input data rate. Removing the quaternion orientation from the Kalman filter reduces the number of state variables from 7 to 3 in a filter that uses the constant velocity (CV) motion model, which, thus, provides significant savings of computational resources. The decoupled prediction algorithm avoids a reduction in frame rate that is required to accommodate the one-step prediction method used in other approaches.

In the remainder of this paper, we will compare the performance and efficiency of two types of head orientation prediction, i.e., the quaternion (Q) method and the proposed DQ method. In Section II, we present an overview of quaternion orientation and the EKF. Section III discusses the details of each of the frameworks that were examined. Section IV focuses on the computation requirements of each of the frameworks. Section V provides a detailed examination of the measured performance of each of the frameworks using head-tracking data captured with a Polhemus ac magnetic tracker in a simulated VR session. Section VI concludes this paper with a comparison of findings and suggestions for future study.

A. Related work

The authors previously developed two adaptive EKF methods for the prediction of quaternion head motion in a simulation environment [11]. The first method used a fading memory algorithm to modify the EKF-predicted error covariance in response to changes in the filter residual. The proposed algorithm

Manuscript received May 11, 2008; revised December 4, 2008 and February 15, 2009. First published May 29, 2009; current version published November 18, 2009. This work was supported by Polhemus, Inc., Colchester, VT. This paper was recommended by Associate Editor M. Pantic.

H. Himberg is with the Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284 USA, and also with Polhemus, Colchester, VT 05446 USA (e-mail: hhimberg@vcu.edu).

Y. Motai is with the Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: ymotai@vcu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2009.2016571

improved the tracking performance but increased the output noise in some conditions. A second method (R-Adaptive) adaptively modified the measurement covariance to control the output noise level. The R-Adaptive approach provided lowered output noise and improved tracking performance with benign data but had increased RMS error with aggressive head motion.

Kiruluta *et al.* proposed a system that used a Kalman filter to predict head motion from position data [1]. The study compared a constant-acceleration Kalman filter predictor to a polynomial approach. Experimental results showed that the Kalman filter provided good latency compensation for moderate motion but had degraded performance undergoing fast motion. An adaptive version of the Kalman predictor was also studied for applications requiring tracking of fast motion at the cost of throughput delay.

Goddard [4] and Bohg [3] both proposed methods of orientation prediction based on the quaternion motion equation presented by Chou [16]. Each of these methods predicted future orientation as a function of angular head velocity and current head orientation. The large state vector of the quaternion filter (seven state variables) and the nonlinear state equation lead to large matrices in the EKF, which results in a large computational load on the host system.

A head-tracking system was developed by Chang and Cho to control the camera movement in a surveillance system application [5]. The proposed system used image-based head tracking to track an individual in a defined physical space. A Kalman filter was used to improve stability by predicting head position in the image space.

Liang *et al.* developed a Q method of head motion prediction based on Kalman filtering [29]. They based their work on the assumption that the perceived latency was mainly caused by the delay in orientation data. The proposed system predicted the head orientation using a linearized quaternion orientation to break the quaternion into four independent components. Each of the four decoupled components was predicted using a separate Kalman filter. The four predicted components were combined to form a predicted unit quaternion value.

Azuma and Bishop developed a predictive tracking system for an HMD using inertial sensors mounted to the display with Kalman filtering [31]. The system improved the latency in most conditions, as compared to prediction without the inertial sensors or no prediction at all.

A comparison of a Grey-theory-based prediction algorithm, a Kalman filter approach, and an extrapolation method was performed by Wu and Ouhyoung [34]. They found that both the Grey theory method and the Kalman filter significantly improved the performance as compared to extrapolation. The authors stated that the Grey theory method performed equally well as the Kalman filter while having a relatively low computation complexity. The computational demands were not qualitatively compared in the study.

LaViola proposed a latency compensation method based on double exponential smoothing as an alternative to Kalman filter prediction [35]. The proposed algorithm was compared to derivative-free Kalman filters (systems without a velocity or acceleration measurement) and found to provide similar performance with a reduced computation requirement.

A phase lead filter system was proposed by So and Griffin to compensate for delays in HMDs [2]. The study found that phase lead filters significantly improved the head-tracking performance but introduced jitter under some conditions. An additional compensation technique using image deflection was used to compensate for filter jitter.

Zhang *et al.* used an adaptive Kalman filter for human movement tracking in medical rehabilitation [12]. The proposed system uses a Kalman filter to control a camera that captures body movement for future analysis.

Quaternion estimation is often used for attitude control in spacecraft. Ali *et al.* used a system based on DQs to control attitude in the Mars Exploration Rover [38]. The system applies a heading adjustment to the previous attitude to estimate the current orientation. Using the new estimate, the system conducts a series of confirmation tests to determine if the attitude estimate is correct. This system uses a variety of sensors including accelerometers, gyroscopes, wheel odometry, and visual odometry to determine vehicle orientation. Similar to our proposal, this system estimates the change in orientation (DQ) and then corrects based on measurement data. Cheon and Kim used an UKF to estimate the spacecraft attitude with quaternions [22]. This study successfully used magnetometer and gyroscopic data to estimate the quaternion orientation with a UKF.

Marins *et al.* developed an orientation sensor based on a magnetic, angular rate, and gravity (MARG) sensor using Kalman filtering [36]. The study proposed two methods of determining position and orientation from MARG measurement data using Kalman filters. In our study, there is no ability to measure the angular rate; the only measurement available is orientation. Our proposed system directly estimates the angular rate information from quaternion orientation measurement data without additional sensor data. Another study conducted by Sabatini proposed the use of a similar sensor (gyroscope, accelerometer, and magnetometer) to measure the orientation in biomedical applications [37]. This study is very similar to [36] in that it uses rate measurement data to estimate orientation and is specific to the particular sensor type being used. Our study develops angular velocity information from quaternion measurements and is more general in that it is not dependant on any measurement data except the orientation itself.

Attitude control systems develop and control orientation using a Kalman filter with a variety of measurement techniques, including gyroscopes, magnetometers, and accelerometers. These applications are based on the same quaternion motion equations that we use in our work but greatly differ in the application specifics. Orientation measurement devices use angular rate data to estimate orientation using a Kalman filter, although the specifics of the filter design considerably vary with sensor type. These applications differ from our study in that we are using quaternions with a Kalman filter to estimate the angular rate information from an orientation measurement. Our approach is independent of the sensor type used for the measurement. Although we have performed our experiment using the Polhemus tracker, any other method of measuring orientation could be used without loss of performance (assuming similar measurement accuracy).

II. Background on Orientation Prediction

A. EKF

The EKF is a prediction-correction filter used in systems with the following state equation and measurement equation:

$$\hat{x}_k = f(\hat{x}_{k-1}, u, w) \tag{1}$$

$$z_k = h(\hat{x}_k, v_k). \tag{2}$$

The state equation [see (1)] expresses the state x at time k as a function of the state at time $k-1$, an external input u, and process noise w (process noise is defined as any change in state not modeled by the state equation). The measurement equation [see (2)] relates the measurement z at time k to the state at time k and measurement noise v. The process noise and the measurement noise are assumed to be independent Gaussian random variables with zero mean [6], [13], [21], [32]. The EKF equations can be applied to nonlinear systems using a Taylor expansion to linearize the system about the current state, i.e.,

$$x_k \approx \tilde{x}_k + A_k \cdot (x_{k-1} - \hat{x}_{k-1}) + W_k \cdot w_{k-1} \tag{3}$$

$$z_k \approx \tilde{z}_k + H_k \cdot (x_k - \hat{x}_k) + V_k \cdot v_k. \tag{4}$$

The A, W, H and V Jacobian matrices are recomputed each time the filter iterates.

B. Quaternions

Unit quaternions are a commonly used method of orientation representation that avoids the singularities of Euler angles and the stability problems of direction cosine matrices [17]–[20], [26], [30]. A unit quaternion is a 4-D representation of orientation that characterizes orientation as a rotation θ about an axis of rotation defined by the unit vector u, i.e.,

$$q = \left[\cos\left(\frac{\theta}{2}\right) u \cdot \sin\left(\frac{\theta}{2}\right)\right]^T. \tag{5}$$

Quaternions provide a compact efficient method of conducting 3-D rotations. To rotate an object, the orientation q_k of the object is multiplied by the desired change in rotation, i.e., the DQ Δq defined as

$$q_k = \Delta q \cdot q_{k-1}. \tag{6}$$

C. Motion Model

A CV motion model is used for each of the frameworks that were investigated. Both DQ and Q frameworks are based on the change in quaternion being a function of angular velocity. At the high data rate of an ac magnetic tracker, the CV model is a good choice for slow to moderate head motion. The CV model assumes that the angular velocity ω is constant from frame to frame using a white noise acceleration component w and the frame period τ to handle any changes in velocity that may occur, e.g.,

$$\omega_k = \omega_{k-1} + w \cdot \tau. \tag{7}$$

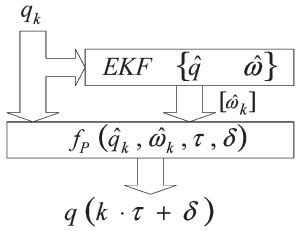

Fig. 1. Q framework as a two-step process that directly estimates future orientation from quaternion orientation measurement data using a Kalman filter and a prediction function (f_P).

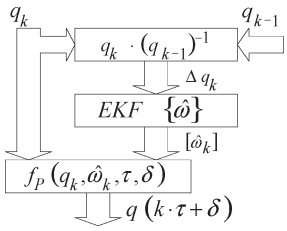

Fig. 2. DQ framework as a three-step process that converts quaternion orientation measurements into DQs. Future orientation is predicted using an EKF and a prediction function (f_P).

III. Filter Design

The Q and DQ frameworks estimate the angular head velocity ω from the measured quaternion orientation q, predicting the future orientation as a function of the estimated head velocity and a user-specified prediction time δ. The Q framework uses an EKF to estimate the current head velocity ω_κ from the quaternion measurement data q_k. Future orientation $q(k\tau + \delta)$ is predicted as a function of the current quaternion measurement q_k, the predicted angular velocity ω_κ, the frame time τ, and the prediction interval δ (Fig. 1). The DQ framework uses a similar process that operates on the DQ between measurements (Fig. 2). The DQ framework first converts the incoming data to DQs (Δq), which are then used by an EKF to estimate the angular head velocity ω. The DQ framework uses the same prediction function as the Q framework, calculating the DQ Δq of the prediction interval and applying it to the current quaternion measurement q_k.

A. Q Framework

The quaternion Kalman filter uses a state vector consisting of the quaternion orientation $\hat{q}$ and the angular velocity vector $\hat{\omega}$, i.e.,

$$\hat{x} = [\hat{q} \quad \hat{\omega}]^T. \tag{8}$$

The state equation

$$f_Q(x, w.\tau) = \begin{bmatrix} q(x, w, \tau) \\ \omega + w \cdot \tau \end{bmatrix} \tag{9}$$

predicts the next state from the current state using the CV model. The measurement equation

$$h(x, v) = \hat{q} + v \tag{10}$$

is linear since the quaternion orientation is included in the state vector.

The relationship between quaternion motion and angular velocity using quaternion multiplication

$$\dot{q} = \Psi \cdot q \tag{11}$$

was presented by Chou [16], where Ψ is the 4×4 element angular velocity quaternion

$$\Psi = \frac{1}{2} \cdot \begin{bmatrix} 0 & -\omega_0 & -\omega_1 & -\omega_2 \\ \omega_0 & 0 & -\omega_2 & \omega_1 \\ \omega_1 & \omega_2 & 0 & -\omega_0 \\ \omega_2 & -\omega_1 & \omega_0 & 0 \end{bmatrix}. \tag{12}$$

As shown in [4], the solution to this differential equation is an exponential function

$$q(t+\tau) = e^{\Psi \cdot \tau} \cdot q(t) \tag{13}$$

which can be solved for the closed discrete form by assuming constant velocity, e.g.,

$$q_k = \Delta q(\omega_k, \tau) \cdot q_{k-1}. \tag{14}$$

The discrete form rotates the current orientation q by a DQ Δq, which is a function of angular velocity ω and time τ. The DQ is computed in its compact four-element column vector form

$$\Delta q(\omega, \tau) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \frac{2 \cdot \omega_0}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \\ \frac{2 \cdot \omega_1}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \\ \frac{2 \cdot \omega_2}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{15}$$

and expanded to a 4×4 matrix for multiplication operations, e.g.,

$$\theta = \tau \cdot \sqrt{\omega^T \cdot \omega}.$$

The predicted angular velocity $\tilde{\omega}$ is generated with the CV motion model as a function of the current angular velocity state $\hat{\omega}$, process noise w, and the frame period τ, i.e.,

$$\tilde{\omega}_k = \hat{\omega}_{k-1} + w \cdot \tau. \tag{16}$$

The predicted quaternion state $\tilde{q}$ is calculated as the product of the DQ (Δq) generated from the predicted angular velocity $\tilde{\omega}$ and the current quaternion state $\hat{q}$, i.e.,

$$\tilde{q}_k = \Delta q(\tilde{\omega}_k, \tau) \cdot \hat{q}_{k-1}. \tag{17}$$

The Kalman filter requires four Jacobian matrices (A, W, H, and V) to be computed each time the filter iterates. The following A matrix contains the partial derivative of the predicted state $\tilde{q}$ with respect to each state variable ($\hat{q}$ and $\hat{\omega}$) and requires three nontrivial partial derivatives:

$$A = \left[\frac{\partial}{\partial \hat{x}} f(x,w)\right]_{w=0} = \begin{bmatrix} \frac{\partial}{\partial \hat{q}}\tilde{q} & \frac{\partial}{\partial \hat{\omega}}\tilde{q} \\ 0 & \frac{\partial}{\partial \hat{\omega}}\tilde{\omega} \end{bmatrix}_{w=0}. \tag{18}$$

The partial derivative of the predicted quaternion $\tilde{q}$ with respect to the quaternion state $\hat{q}$ is the predicted DQ Δq, i.e.,

$$\frac{\partial}{\partial \hat{q}}\tilde{q} = \frac{\partial}{\partial \hat{q}}\left[\Delta q(\tilde{\omega}, \tau) \cdot \hat{q}\right] = \Delta q(\tilde{\omega}, \tau). \tag{19}$$

In this instance, the DQ must be expanded to its full 4×4 matrix format for inclusion in the A matrix.

The partial derivative of the predicted quaternion $\tilde{q}$ with respect to the velocity state $\hat{\omega}$ is calculated as three column vectors

$$\frac{\partial}{\partial \hat{\omega}}\tilde{q} = \begin{bmatrix} \frac{\partial}{\partial \hat{\omega}_0}\tilde{q} & \frac{\partial}{\partial \hat{\omega}_1}\tilde{q} & \frac{\partial}{\partial \hat{\omega}_2}\tilde{q} \end{bmatrix}. \tag{20}$$

Starting with the definition of the predicted quaternion [see (15)], each four-element column vector is the product of the following partial derivative of the predicted DQ with respect to the velocity state $\hat{\omega}$ and the current quaternion state $\hat{q}$:

$$\frac{\partial}{\partial \hat{\omega}_i}\tilde{q} = \left(\frac{\partial}{\partial \hat{\omega}_i}\Delta q(\tilde{\omega}, \tau)\right) \cdot \hat{q}. \tag{21}$$

A generalized form of the partial derivative of the predicted DQ with respect to velocity state $\hat{\omega}$ can be expressed in the following as a function of the predicted DQ $\Delta q(\tilde{\omega}, \tau)$, the predicted angular velocity $\tilde{\omega}$, and the time interval τ:

$$\frac{\partial}{\partial \hat{\omega}_i}\left[\Delta q(\tilde{\omega}, \tau)\right] = \begin{bmatrix} -\frac{\tau}{4} \cdot \Delta\tilde{q}_{i+1} \\ \frac{\tilde{\omega}_0}{\tilde{\Omega}^2} \cdot (\tau \cdot \tilde{\omega}_i \cdot \Delta\tilde{q}_0 - \Delta\tilde{q}_{i+1}) + (\delta_{i,0}) \cdot \frac{1}{\tilde{\omega}_i} \cdot \Delta\tilde{q}_{i+1} \\ \frac{\tilde{\omega}_1}{\tilde{\Omega}^2} \cdot (\tau \cdot \tilde{\omega}_i \cdot \Delta\tilde{q}_0 - \Delta\tilde{q}_{i+1}) + (\delta_{i,1}) \cdot \frac{1}{\tilde{\omega}_i} \cdot \Delta\tilde{q}_{i+1} \\ \frac{\tilde{\omega}_2}{\tilde{\Omega}^2} \cdot (\tau \cdot \tilde{\omega}_i \cdot \Delta\tilde{q}_0 - \Delta\tilde{q}_{i+1}) + (\delta_{i,2}) \cdot \frac{1}{\tilde{\omega}_i} \cdot \Delta\tilde{q}_{i+1} \end{bmatrix}$$

$$\delta_{i,j} = \{dirac\}(i,j) \quad \Omega = \sqrt{\omega^T \cdot \omega}. \tag{22}$$

Finally, the partial derivative of the predicted angular velocity with respect to the velocity state can be obtained by inspection, as follows:

$$\frac{\partial}{\partial \hat{\omega}_i}\tilde{\omega} = I. \tag{23}$$

The following matrix W is the Jacobian of partial derivatives of the predicted state $\tilde{q}$ with respect to the process noise w:

$$W = \left[\frac{\partial}{\partial w} f(x,w)\right]_{w=0} = \left[\frac{\partial}{\partial w}\begin{bmatrix} \tilde{q} \\ \tilde{\omega} \end{bmatrix}\right]_{w=0}. \tag{24}$$

The CV motion model considerably simplifies W, since the predicted velocity $\tilde{\omega}$ is a linear function of the velocity state $\hat{\omega}$ and the process noise w. Closer inspection indicates that it is the product of the partial derivative of the predicted quaternion with respect to velocity state [see (21)] and the time step τ, i.e.,

$$W = \begin{bmatrix} \tau \cdot \frac{\partial}{\partial \tilde{\omega}} q(\tilde{\omega}, \tau) \\ \tau \cdot I \end{bmatrix}. \tag{25}$$

The H matrix is the Jacobian of partial derivatives of the measurement equation [see (10)] with respect to state $\hat{x}$, which

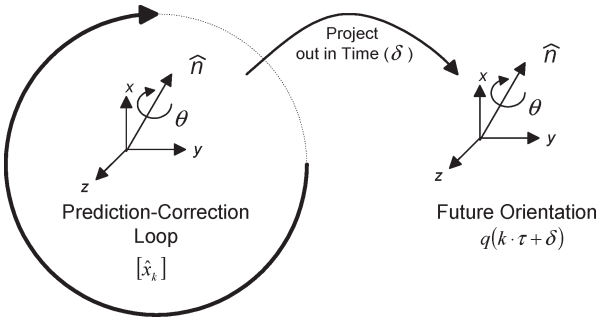

Fig. 3. Prediction-correction loop of the Kalman filter providing an estimate of angular head velocity that is projected across the prediction interval to estimate the change in orientation (Δq) that will occur.

can be derived by inspection as

$$H = \left[\frac{\partial}{\partial \hat{q}} h(x,v) \quad \frac{\partial}{\partial \hat{\omega}} h(x,v)\right]_{v=0} = [I \quad 0]. \tag{26}$$

The V matrix is the Jacobian of partial derivatives of the measurement equation [see (10)] with respect to measurement noise. Since the measurement model is linear, V is a 4×4 identity matrix, i.e.,

$$V = \frac{\partial}{\partial v} h(x,0) = I. \tag{27}$$

The Q framework uses a seven-element state vector to estimate the angular velocity. Close examination of the quaternion filter equation reveals that the DQ Δq is the driving equation of the filter. All information concerning the change in orientation is contained in the DQ, with the quaternion state providing a method or converting the DQ to match the measurement.

B. DQ Framework

The DQ framework removes the quaternion equation from the estimation process by directly converting incoming quaternion data q_k to DQ Δq_k before using the EKF. The EKF now directly predicts the angular head velocity $\tilde{\omega}$ from the DQ Δq_k. The quaternion motion equation [see (14)] is only needed to compute the predicted quaternion $q(k\tau + \delta)$ and is moved outside the Kalman filter into the orientation prediction process (Fig. 3). The DQ framework directly estimates the angular head velocity from a DQ measurement without estimation of the quaternion itself. Eliminating the quaternion orientation from the Kalman filter reduces the state vector from seven elements to three elements when using the CV motion model. The resulting reduction in matrix rank (from 7×7 to 3×3) results in large savings of computational resources while retaining the quaternion motion model [see (14)].

The DQ of the current frame Δq_k represents the change in quaternion between the previous frame at time $k-1$ and the current frame a time k. The DQ is computed as the quaternion product of the current quaternion and the inverse of the previous quaternion, i.e.,

$$\Delta q_k = q_k \cdot (q_{k-1})^{-1}. \tag{28}$$

The DQ Kalman filter uses a three-element state vector

$$x = [\hat{\omega}] \tag{29}$$

containing the average angular velocity.

The CV state equation is now a linear function of the angular velocity state $\hat{\omega}$, the process noise w, and the time interval τ, i.e.,

$$f_{DQ}(x,w) = \hat{\omega} + w \cdot \tau. \tag{30}$$

The measurement model in the DQ Kalman filter must relate the predicted angular head velocity $\tilde{\omega}_k$ to the DQ measurement Δq_k. The equation used for the DQ prediction in the quaternion EKF [see (15)] is used as the measurement equation for the DQ EKF, i.e.,

$$h(x,v) = \Delta q(\omega,\tau) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \frac{2\cdot\omega_0}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \\ \frac{2\cdot\omega_1}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \\ \frac{2\cdot\omega_2}{\|\omega\|} \cdot \sin\left(\frac{\theta}{2}\right) \end{bmatrix} + v. \tag{31}$$

It should be noted that both DQ and Q frameworks compute the difference between the measured and predicted quaternion as a simple subtraction, which is technically not a valid quaternion operation. The small time interval between input data samples minimizes the effect of this compromise

Due to the linear state equation, the DQ A and W matrices are constant and do not have to be computed for each iteration of the Kalman filter, e.g.,

$$A = \frac{\partial}{\partial \hat{\omega}} (\hat{\omega} + w \cdot \tau) = I \tag{32}$$

$$W = \frac{\partial}{\partial w} (\hat{\omega} + w \cdot \tau) = [\tau \cdot I]. \tag{33}$$

The Jacobian H matrix is the partial derivative of the measurement equation $h(x,v)$ with respect to state $\hat{x}$. Since the DQ state vector only contains the angular velocity ω, the H matrix reduces to the partial derivative of the DQ Δq with respect to the velocity state $\hat{\omega}$ as follows:

$$H = \left[\frac{\partial}{\partial \hat{\omega}_0} \Delta q(\tilde{\omega},\tau) \quad \frac{\partial}{\partial \hat{\omega}_1} \Delta q(\tilde{\omega},\tau) \quad \frac{\partial}{\partial \hat{\omega}_2} \Delta q(\tilde{\omega},\tau)\right]. \tag{34}$$

The partial derivative of the DQ with respect to the angular velocity state was derived in the Q filter derivation [see (22)].

The DQ measurement equation is linear with respect to measurement noise v, which reduces V to the identity matrix

$$V = \left[\frac{\partial}{\partial v} [h(x,v)]\right]_{v=0} = I. \tag{35}$$

C. Quaternion Prediction

The proposed prediction method is designed to execute on the Polhemus tracker at frame rates of up to 240 Hz. Each of the frameworks supports a user-specified prediction time (δ) for maximum flexibility. Future orientation is predicted by assuming that the current velocity state $\hat{\omega}$ remains constant

TABLE I
COMPUTATIONAL REQUIREMENTS FOR ONE ITERATION

	Div.	Add.	Multi.	Higher Level Functions	Matrix Inverse
Q	12	1612	2092	3	1 (4x4)
DQ	18	297	438	3	1 (4x4)

throughout the prediction interval (Fig. 3). The future head orientation $q(kt+\delta)$ is estimated as a function of the current quaternion measurement q, the angular velocity ω, the frame time τ, and the prediction interval δ, i.e.,

$$f_P(q,\omega,\tau,\delta)) = \Delta q(\hat{\omega}_k,\delta)\cdot(q_k). \tag{36}$$

The function f_P first computes the DQ that occurs if the angular head velocity ω is constant across the prediction interval δ and then applies it to the current quaternion measurement q_k as

$$q(k\cdot t+\delta) = f_P(q,\omega,\tau,\delta)). \tag{37}$$

IV. COMPARISON OF FILTER DESIGN

Each of the frameworks examined uses a multiple-stage process to predict orientation (Table I). The two frameworks examined have widely varied computational requirements due to the complexity of the system and measurement equations (Table II). Approximately the same number of higher-level function calls (sine, cosine, and square root) and inverse matrix operations are required by each of the frameworks. The higher-level functions are used in the DQ computation, which is common to both frameworks, although it appears in different locations in each algorithm. The single 4×4 matrix inverse operation in each framework occurs in the computation of the Kalman gain and, fortunately, is not effected by the expanded state vector of the Q framework EKF. The Q framework substantially requires more multiplications and additions than the DQ due to the larger state variable. The seven-element state vector of the Q EKF requires three 7×7 matrices (A, A^T and P) in probability covariance calculation. Additionally, the W matrix expands to 7×3, and H expands to 4×7. The expanded matrices of the Q framework are each applied multiple times during the Kalman filter prediction-correction process, which results in a fivefold increase of additions and multiplication for the Q as compared to the DQ.

V. EXPERIMENTAL ANALYSIS

A. Experimental Data

Quaternion head motion data were collected in a simulation of a cockpit VR environment using a Polhemus Liberty ac magnetic tracker operating at a 120-Hz frame rate. The data collection setup consisted of a single Polhemus magnetic sensor mounted on the rear of a headband worn by the test subject. A Polhemus magnetic source was positioned approximately six lampe bergers behind the test subject. There was no effort to control the alignment of the sensor in the source frame. Thirteen individual data sets were collected for this experiment: three sets targeting specific head motion categories (benign, moderate, and aggressive) and ten additional sets containing the full range of motion expected in a VR cockpit simulation session. Each of the 13 data sets consists of 10 000 data frames, which represent 83.33 s of continuous data collection (Figs. 4–7). The three motion-specific data sets (tuning data) will be used for filter tuning and performance analysis under specific types of head motion.

The benign motion data set consists of stationary head orientation with smooth gradual transitions between orientations and is intended to represent targeting and observation activities (Fig. 4). The moderate motion data set includes discrete head orientations with smooth transitions at moderate velocities similar to the visual scanning motion a pilot might use (Fig. 5). The aggressive data set is included to represent high-velocity tracking head movement with rapid starts and stops, as would be experienced when a pilot attempts to find or track a rapidly moving target (Fig. 6).

The ten full-range motion data sets are intended to be representative of typical head motion during a cockpit simulation session and will be used for performance analysis. The data sets contain intervals of benign, moderate, and aggressive motion in pseudorandom order (Fig. 7).

B. Tuning

The Kalman filter uses the process noise covariance and the measurement noise covariance to tune the filter for the targeted application. VR environments are typically custom built in small lots, which leads to a large variation in how the magnetic source and sensor are positioned in the simulation environment. We chose to directly estimate the two covariance parameters from the measured data to allow customization of the filter tuning parameters to each installation. Although this approach does mean our results are specific to the collected data set, the process is easily repeatable in an installation environment and, in fact, could be included in the tracker firmware application. The tuning parameters were directly derived from a composite data set constructed by combining the three tuning data sets (benign, moderate, and aggressive) and two of the full-motion data sets. This approach was chosen to provide an even weighting of the three categories of head motion while including intermediate data types not represented by the three tuning data sets.

C. Measurement Noise Covariance

The DQ and Q filters use different measurement data in the correction phase of the EKF. The DQ filter uses DQ data derived from the quaternion measurement, whereas the Q filter uses the quaternion measurement itself. For this experiment, we decided to estimate the measurement noise as the difference between measurement data and a "denoised" version of the same data. For the Q filter, an estimate of the underlying "noiseless" version of the composite data was created by smoothing with a Gaussian kernel. The smoothed quaternion was then subtracted from the measured quaternion to estimate the measurement noise. The DQ measurement noise is estimated by applying the same technique to DQ data. The DQ measurement data

TABLE II
OVERVIEW OF FRAMEWORK METHODOLOGY

Framework	Pre-Processor	Kalman Filter	Post Processor
Q	None	Estimate angular head velocity to predict the next quaternion value using a single EKF.	Predict future orientation as a function of the current quaternion orientation, head velocity, and the prediction time.
DQ	Convert quaternion orientation to delta quaternion	Estimate head velocity using the delta quaternion as the measurement data.	A delta quaternion estimating the change in orientation across the prediction interval is applied to the current quaternion measurement to predict future orientation.

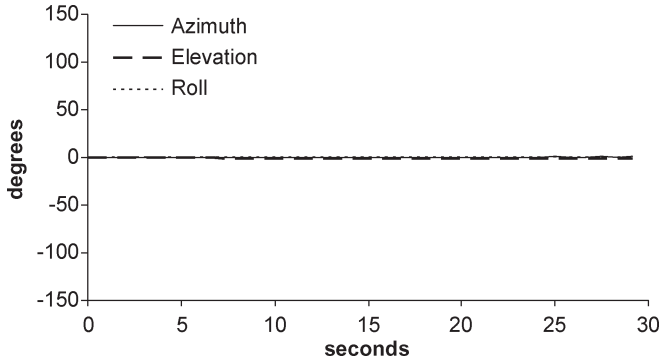

Fig. 4. Benign head motion data representing semi-stationary activities such weapons control (first 30 s shown as Euler angles in degrees).

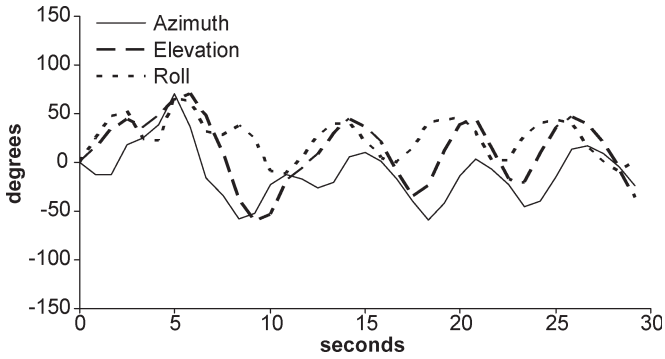

Fig. 5. Moderate head motion data showing smooth but rapid head motion (first 30 s shown as Euler angles in degrees).

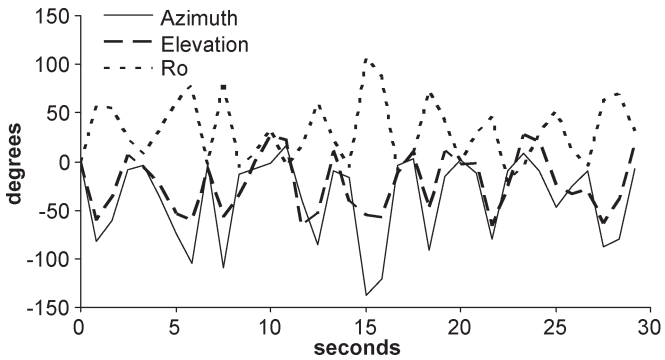

Fig. 6. Aggressive head motion data (first 30 s shown as Euler angles in degrees). Note the very rapid and sometimes erratic motion.

are computed on a frame-by-frame basis from the measured quaternion orientation, whereas the smoothed DQ data are generated from the smoothed data. The measurement noise for the DQ filter is then estimated as the difference between the measured DQ and the smoothed DQ. Using the two variable

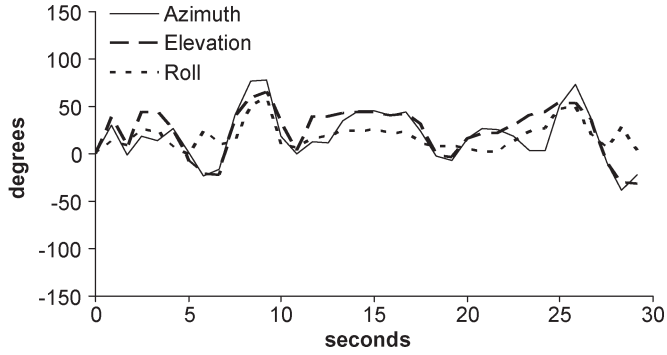

Fig. 7. Full head motion data as a continuous data capture session that includes a complete range of head motion to closely match the simulation session data (first 30 s of a typical example shown as Euler angles).

TABLE III
MEASUREMENT NOISE VARIANCE

	DQ filter	Q filter
var(z_0)	9.74e-13	2.55e-07
var(z_1)	1.21e-08	4.67e-07
var(z_2)	3.99e-09	4.10e-07
var(z_3)	4.13e-09	3.28e-07

quaternion representations (a rotation θ about an axis u), we can compare the measurement data variance of the two filters as

$$z = \begin{bmatrix} \cos(\theta/2) \\ u_0 \sin(\theta/2) \\ u_1 \sin(\theta/2) \\ u_2 \sin(\theta/2) \end{bmatrix}. \tag{38}$$

In Table III, we see that the z_0 component of the measurement data has a much smaller variance for DQ than for Q (9.74e-13 versus 2.55e-07). For this experiment, the change in orientation between frames is small due to the high frame rate (120 Hz), which results in a DQ measurement near the identity quaternion $(q = [1\ 0\ 0\ 0]^{\mathrm{T}})$. The small changes in rotation θ between frames cause an even smaller variation in the z_0 component of the measurement noise, because it is a function of the rotation θ, which has a zero slope for $\theta = 0$. The axis components of the DQ measurement noise (z_1, z_2, and z_3) also have a very small variance due to the influence of the sine function with θ near zero $(\sin(0) = 0)$. The Q measurement is the total rotation of the current orientation from the origin and is typically not representative of a rotation near zero. Accordingly, the Q measurement noise variance is much larger than the associated DQ values, with each of the Q values having a similar magnitude due to the averaging effect of the variance calculation.

TABLE IV
PROCESS NOISE COVARIANCE USED BY BOTH DQ AND Q FRAMEWORKS

	ω_0	ω_1	ω_2
ω_0	5.75	0.23	-0.41
ω_1	0.23	1.12	0.04
ω_2	-0.41	0.04	1.96

D. Process Noise Covariance

The DQ and Q filters both utilize a variation of the angular velocity through the process noise covariance as the driving variable of the Kalman filter prediction step. The Q filter propagates changes in the angular velocity into the quaternion state through application of a DQ (a function of the angular velocity) to the previous quaternion state estimate, whereas the DQ filter uses the DQ itself. In the CV model, the process noise can be modeled as the angular acceleration not related to the measurement noise. In our experiment, the tuning parameters must be derived from measurement data, which raises the issue of how to remove measurement noise from the data before estimating the process noise covariance (Table IV). We used a smoothed version of the full-range data set to provide a "noiseless" quaternion measurement from which to estimate the process noise. A DQ was calculated for each frame of the "noiseless" quaternion, and then the angular velocity was estimated by solving the DQ equation [see (15)] using the Levenberg–Marquardt algorithm (LMA). The process noise was estimated from the angular velocity by applying the CV model to the data on a frame-by-frame basis. The difference between the estimated velocity of a given frame and the previous frame estimate was considered to be process noise.

E. Execution Time

The single-iteration execution time was measured for each of the frameworks in the MathCAD simulation environment. The iteration time was computed as the average time required to process one frame of data. The DQ framework executed a single pass in 520 μs, whereas the Q framework required 921 μs. A tabulation of the number of operations required by each framework (Table I) showed that the DQ provided approximately 80% improvement in the number of additions and multiplications, but our experimental results only showed a 43.5% improvement. The less than expected improvement in execution time using the DQ is the result of the efficiency of the floating-point unit in the simulation host (Pentium 4; 3 GHz). The higher-level functions (sine, cosine, and square root) and the inverse matrix operations occur at the same frequency in both frameworks, which leaves the increased matrix rank of the Q framework as the only difference between the two. Modern floating-point units can generally execute one or more multiply/accumulate (MAC) operations in one instruction cycle, which reduces the improvement in execution speed.

F. Prediction Accuracy

Filter performance was rated by comparing the quaternion prediction to the actual time-shifted data after conversion to Euler angles (azimuth, elevation, and roll). The Euler error for each sample point was computed and then combined to form the RMS average of the compound angle. Error was measured as average error (degrees), overshoot points (as a percentage of total), overshoot average (degrees), and maximum overshoot (degrees). Overshoot was defined as any error exceeding 1.0 degrees of the composite angle.

The two frameworks showed comparable accuracy at the typical prediction time of 50 ms (Table V). When used with benign motion, there essentially was no error in using any of the prediction methods, since the orientation did not appreciably change during the prediction interval. Testing under moderate and aggressive motion illustrates the great improvement in prediction that the Q and DQ filters provide as compared to no prediction. For moderate motion, the overshoot percentage dropped by 35% and approached a 50% improvement for aggressive motion. The Q filter provided better prediction for moderate motion than the DQ filter. Overshoot was higher for DQ than Q during moderate motion (5.22% versus 0.58%), but the average error was not significantly different (0.31 versus 0.33). The maximum overshoot for DQ was approximately twice that of Q for moderate motion (2.69 versus 1.34), whereas the average overshoot was only slightly higher for the DQ. Overall, the performance data at 50 ms suggest that the DQ filter will handle the aggressive behavior better than the Q filter at the cost of performance for moderate motion.

Looking at performance by category as a function of prediction time, we can see that both filters had similar performance but with different profiles (Figs. 8–11). The average error of the DQ for moderate motion was slightly better than the Q values when the prediction time was reduced below 50 ms, but significantly increased above the Q for prediction times greater than 50 ms (Fig. 8). With aggressive data, the average error was lower in DQ than in Q at all the prediction times. The overshoot average (Fig. 9) and the percentage overshoot (Fig. 10) were always lower with Q than with DQ for moderate motion, but DQ was better with aggressive motion.

The maximum overshoot (Fig. 11) showed that Q had better performance during moderate motion but worse performance during aggressive motion.

The DQ filter output is very responsive to changes in angular velocity, since these changes directly impact the DQ, which is applied to the measured quaternion for prediction. The Q filter, however, applies the DQ to the quaternion state, which is dependent on not only the quaternion measurement but also the Kalman gain (per the correction process). Changes in the DQ for the Q filter are not directly reflected in the output, they must propagate through the filter, slowing the response and increasing the error for aggressive motion or increased prediction time. The reduced performance of DQ with moderate motion is primarily due to larger overshoots, since the average error is not significantly different from Q. The improved responsiveness of DQ helps it perform better under aggressive motion but also causes it to suffer from increased overshoot during moderate motion.

The DQ and Q filters were also tested with ten different full-motion data sets to measure the expected performance in a VR simulation environment. All performance measurements of the ten sets were calculated across the combined 100 000-frame

TABLE V
FRAMEWORK PERFORMANCE AT 50 ms OF PREDICTION

Filter	Benign Motion				Moderate Motion				Aggressive Motion			
	Avg. Err (°)	OS (%)	OS Avg. (°)	OS Max (°)	Avg. Err (°)	OS (%)	OS Avg. (°)	OS Max (°)	Avg. Err (°)	OS (%)	OS Avg. (°)	OS Max (°)
DQ	0.04	0.00	0.00	0.00	0.31	5.22	1.46	2.69	1.11	48.04	1.67	5.25
Q	0.05	0.00	0.00	0.00	0.33	0.58	1.17	1.34	1.77	56.61	1.92	7.0
No Pred.	0.04	0.00	0.00	0.00	1.22	39.73	2.51	5.05	3.79	97.95	5.20	15.76

Notes: The following abbreviation are used in Table 5: OS overshoot, ° degrees, and % percent.

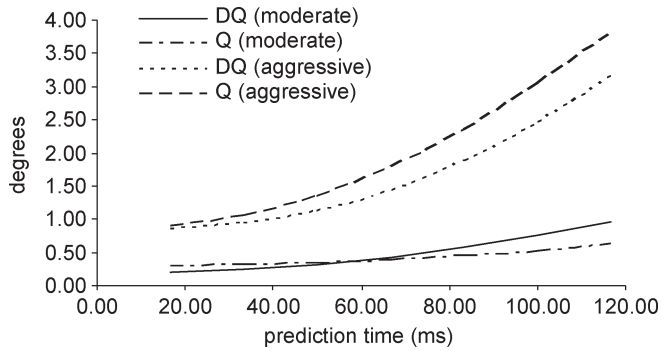

Fig. 8. Average error for moderate and aggressive head motion as a function of prediction time (compound error in degrees). DQ performs better with aggressive motion than Q but is slightly worse when using moderate motion data.

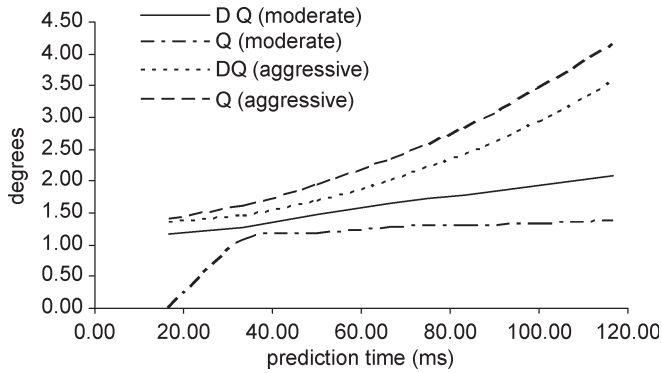

Fig. 9. Average overshoot versus prediction time for moderate and aggressive head motion (total overshoot in degrees). Q performs much better with moderate motion but is much worse with aggressive motion.

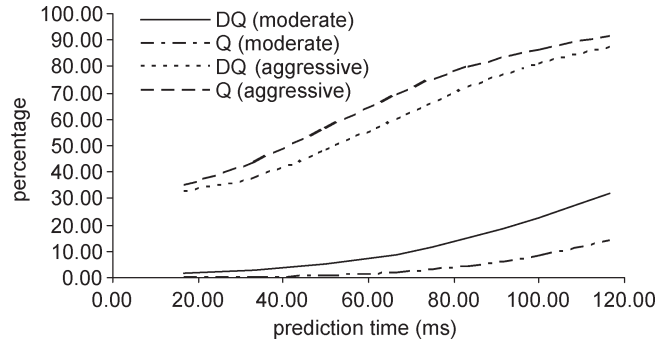

Fig. 10. Percentage overshoot as a function of prediction time for moderate and aggressive head motion (percentage of sample size).

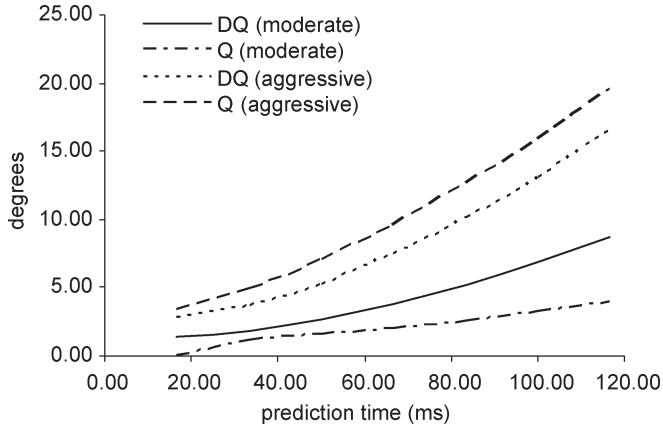

Fig. 11. Maximum overshoot for moderate and aggressive data as a function of prediction time (shown in degrees). Note that Q provides the best performance with moderate motion, but DQ is better for the aggressive case.

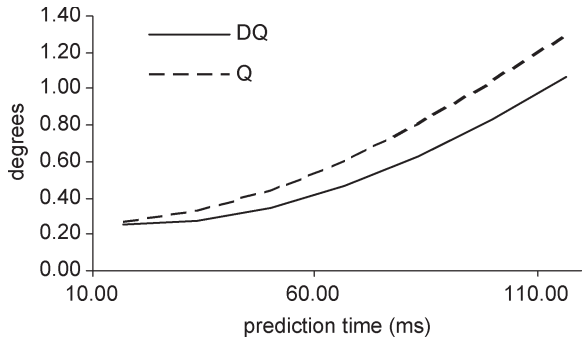

Fig. 12. Average error versus prediction time for full-motion data showing the DQ outperforming the Q framework for all prediction times.

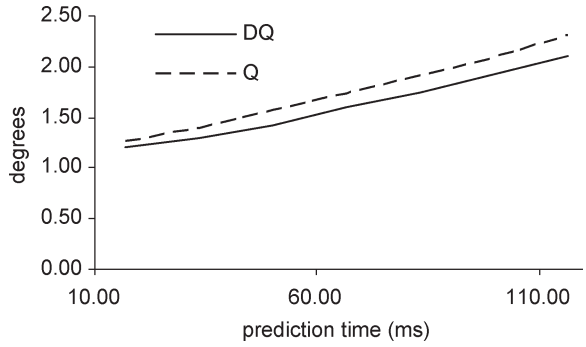

Fig. 13. Average overshoot was marginally better for DQ with the full-motion data.

sample time (13.88 min) to create a profile for the DQ and Q filters as a function of prediction time. The DQ filter provided improved performance in all the error measurements when using full-motion data at any prediction time. The improved aggressive motion performance of DQ allows it to quickly respond to sudden movements, which reduces the average error across the entire simulation interval (Fig. 12). The overshoot average was slightly improved (Fig. 13), whereas the overshoot

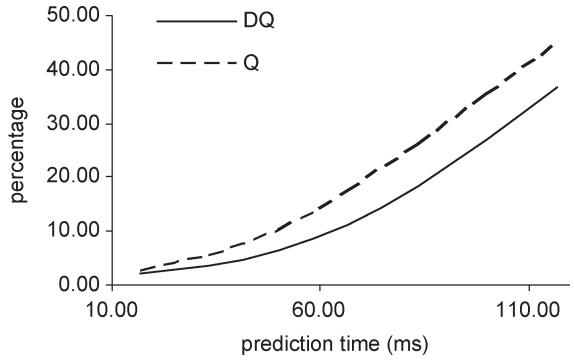

Fig. 14. DQ had a lower overshoot percentage than Q at all prediction times, with almost 10% improvement at 110 ms.

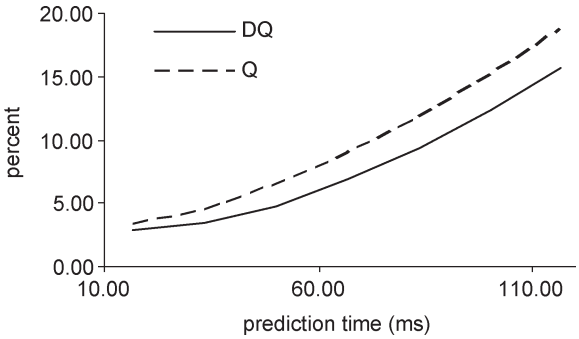

Fig. 15. Maximum overshoot was significantly improved with DQ at all prediction times using full-motion data.

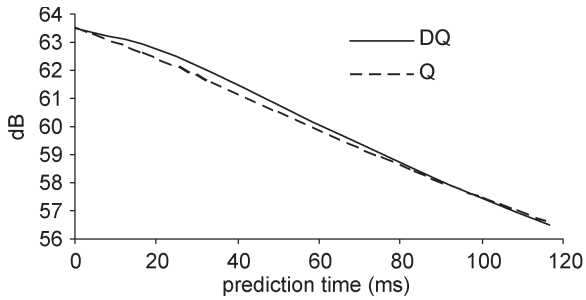

Fig. 16. Output SNR (in decibels) as a function of prediction time (in milliseconds) was nearly identical for the two frameworks (full-motion data).

percentage (Fig. 14) and the maximum overshoot (Fig. 15) were significantly improved. The results for the full-motion simulations suggest that the improved performance of DQ during aggressive motion is a dominant factor in the overall performance of the prediction process.

G. Noise Performance

The prediction process introduces noise into the quaternion data when it projects the current head velocity forward in time. The small changes in the estimated velocity, which are caused by the prediction-correction behavior of the Kalman filter, are amplified by the prediction process. For this experiment, the output noise was estimated as the difference between the output data and a smoothed version of itself, which is expressed in decibels. The expectation was that the output SNR would drop as the prediction time increased. As shown in Fig. 16, the SNR

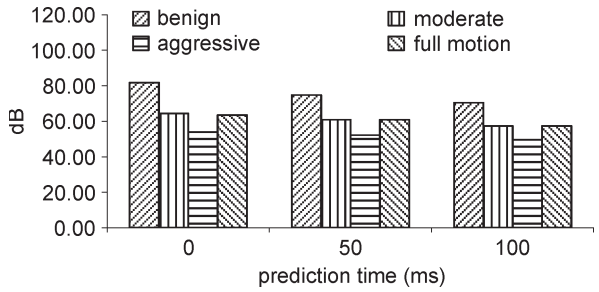

Fig. 17. DQ framework output SNR (in decibels) versus prediction time (in milliseconds) by motion category (benign, moderate, aggressive, and full-range head motion). Note that the full-motion category nearly has the same SNR performance as the moderate motion category.

dropped approximately 7 dB when the prediction was increased from 0 to 120 ms. The two filters displayed nearly identical noise performance with the DQ filter being slightly better than Q (Fig. 16). The DQ displayed increasing output noise as the head motion changed from benign to aggressive (Fig. 17). The full-motion data set provided similar noise performance with the moderate data, suggesting that it is a relatively equal weighting of the three data categories. The 0-ms prediction case indicates that the majority of change is caused by the tracker and not by the prediction algorithm.

VI. Conclusion

In this paper, we have presented a new method of head orientation prediction that estimates the DQ that occurs across a user-specified prediction interval. The proposed method uses a three-step framework to provide a computationally efficient mechanism for predicting the future orientation within the 4-D quaternion space. The DQ framework was compared to the quaternion EKF (Q) filter using head motion data, which represent three individual categories of head motion with prediction intervals varying from 0 to 116 ms. Additional experiments were conducted with data sets representative of head motion in a VR/AR environment. Experimental results show that the DQ approach provides a prediction performance that is similar to quaternion EKF while only requiring a fraction of the computational load.

References

[1] A. Kiruluta, M. Eizenman, and S. Pasupathy, "Predictive head movement tracking using a Kalman filter," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 2, pp. 326–331, Apr. 1997.

[2] R. H. Y. So and M. J. Griffin, "Experimental studies of the use of phase lead filters to compensate lags in head-coupled visual displays," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 26, no. 4, pp. 445–454, Jul. 1996.

[3] J. Bohg, "Real-times structure from motion using Kalman filtering," Ph.D. dissertation, Technische Universität Dresden, Dresden, Germany, 2005.

[4] J. S. Goddard, Jr., "Pose and motion estimation from vision using dual quaternion-based extended Kalman filtering," Ph.D. dissertation, Univ. Tennessee, Knoxville, TN, 1997.

[5] W. C. Chang and C. W. Cho, "Active head tracking using integrated contour and template matching in indoor cluttered environment," in *Proc. IEEE Conf. Syst., Man, Cybern.*, Oct. 11, 2006, vol. 6, pp. 5167–5172.

[6] R. F. Stengel, *Stochastic Optimal Control*. New York: Wiley, 1986.

[7] A. V. Rhijn, R. V. Liere, and J. D. Mulder, "An analysis of orientation prediction and filtering methods for VR/AR," in *Proc. IEEE Virtual Reality Conf.*, 2005, pp. 67–74.

[8] J. R. Wu and M. Ouhyoung, "On latency compensation and its effects on head-motion trajectories in virtual environments," *Vis. Comput.*, vol. 16, no. 2, pp. 79–90, Mar. 2000.
[9] R. H. Y. So, *Lag Compensation by Image Deflection and Prediction: A Review on the Potential Benefits to Virtual Training Applications for Manufacturing Industry*, Hong Kong: Hong Kong Univ. Sci. Technol., unpublished.
[10] R. H. Y. So, W. T. Lo, and A. T. K. Ho, "Effects of navigational speed on motion sickness caused by an immersive virtual environment," *Hum. Factors*, vol. 43, no. 3, pp. 452–461, Fall 2001.
[11] H. Himberg, Y. Motai, and C. Barrios, "R-adaptive Kalman filtering approach to estimate head orientation for driving simulator," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 851–857.
[12] Y. Zhang, H. Hu, and H. Zhou, "Study on adaptive Kalman filtering and algorithms in human movement tracking," in *Proc. IEEE Int. Conf. Inf. Acquisition*, 2005, pp. 11–15.
[13] G. Welch and G. Bishop, "An introduction to the Kalman filter," in *Proc. SIGGRAPH*, 2001, pp. 1–47. Course 8.
[14] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, and J. E. Zacher, "Tolerance of temporal delay in virtual environments," in *Proc. IEEE Virtual Reality Conf.*, 2001, pp. 247–254.
[15] J. Y. Jung, B. D. Adelstein, and S. R. Ellis, "Discriminability of prediction artifacts in a time-delayed virtual environment," in *Proc. IEA/HFES*, 2000, pp. 499–502.
[16] J. C. K. Chou, "Quaternion kinematic and dynamic differential equations," *IEEE Trans. Robot. Autom.*, vol. 8, no. 1, pp. 53–64, Feb. 1992.
[17] E. A. Coutsias and L. Romero, "The quaternions with an application to rigid body dynamics," in *Lecture Notes for Seminar on Dynamic and Stability Theory*. Albuquerque, NM: Dept. Math., Univ. New Mexico, Feb. 1999.
[18] E. A. Coutsias and L. Romero, "The quaternions with an application to rigid body dynamics," Sandia Nat. Lab., Albuquerque, NM, Tech. Rep. SAND2004-0153, 2004.
[19] A. L. Schwab, *Quaternions, Finite Rotation and Euler Parameters*, 2002, unpublished.
[20] M. D. Shuster, "A survey of attitude representations," *J. Astronaut. Sci.*, vol. 41, no. 4, pp. 439–517, Oct.–Dec. 1993.
[21] Y. Bar-Sholon, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation*. New York: Wiley, 2001.
[22] Y.-J. Cheon and J.-H. Kin, "Unscented filtering in a unit quaternion space for spacecraft attitude estimation," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2007, pp. 66–71.
[23] R. Azuma and F. Bishop, *A Frequency-Domain Analysis of Head-Motion Prediction*, NSF/ARPA Sci. Technol. Center Comput. Graph. Visualization. (ARPA contract DABT63-93-C-C048).
[24] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, and J. E. Zacher, "Tolerance of temporal delay in virtual environments," in *Proc. IEEE Virtual Reality Conf.*, 2001, pp. 247–254.
[25] P. M. Jaekl, R. S. Allison, L. R. Harris, U. T. Jasiobedzka, H. Jenkin, M. R. Jenkin, J. E. Zacher, and D. C. Zikovitz, "Perceptual stability during head movement in virtual reality," in *Proc. IEEE Virtual Reality Conf.*, 2002, vol. 4, pp. 149–155.
[26] A. Ude, "Filtering in a unit quaternion space for model-based object tracking," *Robot. Auton. Syst.*, vol. 28, no. 2/3, pp. 163–172, Aug. 1999.
[27] J. LaViola, "A comparison of unscented and extended Kalman filtering for estimating quaternion motion," in *Proc. Amer. Control Conf.*, Jun. 2003, pp. 2435–2440.
[28] F. Ababsa, M. Mallem, and D. Roussel, "Comparison between particle filter approach and Kalman filter-based technique for head tracking in augmented reality systems," in *Proc. IEEE ICRA*, May 2004, pp. 1021–1026.
[29] J. Liang, C. Shaw, and M. Green, "On temporal-spatial realism in the virtual reality environment," in *Proc. 4th Annu. Symp. User Interface Softw. Technol.*, Nov. 1991, pp. 19–25.
[30] S. B. Choe and J. J. Faraway, "Modeling head and hand orientation during motion using quaternions," *SAE Trans.*, vol. 113, no. 1, pp. 186–192, 2004.
[31] R. Azuma and G. Bishop, "Improving static and dynamic registration in a see-through HMD," in *Proc. SIGGRAPH*, 1994, pp. 197–204.
[32] P. S. Maybeck, *Stochastic Models, Estimation and Control*. New York: Academic, 1979.
[33] K. Terada, A. Oba, and A. Ito, "3D human head tracking using hypothesized polygon model," in *Proc. IEEE Conf. Syst., Man, Cybern.*, Oct. 2005, vol. 2, pp. 1396–1401.
[34] J.-R. Wu and M. Ouhyoung, "A 3D tracking experiment on latency and its compensation methods in virtual environments," in *Proc. 8th Annu. ACM Symp. User Interface Softw. Technol.*, Nov. 1994, pp. 41–49.
[35] J. LaViola, "Double exponential smoothing: An alternative to Kalman filter-based predictive tracking," in *Proc. ACM Int. Conf. Series*, 2003, vol. 39, pp. 199–206.
[36] J. Marins, X. Yun, E. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended Kalman filter for quaternions-based orientation using MARG sensors," in *Proc. Int. Conf. Intell. Robots Syst.*, Nov. 2001, vol. 4, pp. 2003–2011.
[37] A. Sabatini, "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 7, pp. 1346–1356, Jul. 2006.
[38] K. Ali, C. Vanelli, J. Biesiadecki, M. Maimone, U. Y. Cheng, A. Martin, and J. Alexander, "Attitude and position estimation on the mars exploration rovers," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2005, vol. 1, pp. 20–27.

Henry Himberg received the B.S. degree in electrical and computer engineering from Clarkson University, Potsdam, NY, in 1982 and the M.S. degree from the University of Vermont, Burlington, VT, in 2005.

He is a graduate student with the Virginia Commonwealth University, Richmond, VA, and is currently a Principal Engineer with Polhemus, Colchester, VT. His research interests include signal processing, position/orientation measurement, and motion prediction.

Yuichi Motai (M'01) received the B.Eng. degree in instrumentation engineering from Keio University, Tokyo, Japan, in 1991, the M.Eng. degree in applied systems science from Kyoto University, Kyoto, Japan, in 1993, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2002.

He is currently an Assistant Professor of electrical and computer engineering with Virginia Commonwealth University, Richmond, VA. His research interests include the broad area of sensory intelligence, particularly in medical imaging, pattern recognition, computer vision, and robotics.