A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion

Joseph J. LaViola Jr.
Brown University Technology Center
for Advanced Scientific Computing and Visualization
PO Box 1910, Providence, RI, 02912, USA
Email:jjl@cs.brown.edu

***Abstract*— The unscented Kalman filter is a superior alternative to the extended Kalman filter for a variety of estimation and control problems. However, its effectiveness for improving human motion tracking for virtual reality applications in the presence of noisy data has been unexplored. In this paper, we present an empirical study comparing the performance of unscented and extended Kalman filtering for improving human head and hand tracking. Specifically, we examine human head and hand orientation motion signals, represented with quaternions, which are critical for correct viewing perspectives in virtual reality. Our experimental results and analysis indicate that unscented Kalman filtering performs equivalently with extended Kalman filtering. However, the additional computational overhead of the unscented Kalman filter and quasi-linear nature of the quaternion dynamics lead to the conclusion that the extended Kalman filter is a better choice for estimating quaternion motion in virtual reality applications.**

Keywords: extended Kalman filtering, unscented Kalman filtering, human motion tracking, quaternions, virtual reality

I. Introduction

Accurate human motion tracking is a critical component in any virtual reality (VR) application [1]. Having real time head and hand motion information enables the computer to draw images in the correct perspective. Unfortunately, tracking systems suffer from noise and small distortions causing incorrect viewing perspectives. To handle these imperfections, filtering is often applied to the tracked data so the VR application can obtain more accurate estimates of the user's motion.

The Kalman filter (KF) is a popular choice for estimating user motion in VR applications[2][3][4]. Since position information is linear, standard Kalman filtering can be easily applied to the tracking problem without much difficulty. However, human pose information also contains nonlinear orientation data, requiring a modification to the KF. The extended Kalman filter (EKF) provides this modification by linearizing all nonlinear models (i.e., process and measurement models) so the traditional KF can be applied[5].

Unfortunately, the EKF has two important potential drawbacks. First, the derivation of the Jacobian matrices, the linear approximators to the nonlinear functions, can be complex causing implementation difficulties. Second, these linearizations can lead to filter instability if the timestep intervals are not sufficiently small[6].

To address these limitations, Julier and Uhlmann developed the unscented Kalman filter (UKF)[7]. The UKF operates on the premise that it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function. Instead of linearizing using Jacobian matrices, the UKF using a deterministic sampling approach to capture the mean and covariance estimates with a minimal set of sample points. The UKF is a powerful nonlinear estimation technique and has been shown to be a superior alternative to the EKF in a variety of applications including state estimation for road vehicle navigation[8], parameter estimation for time series modeling[9], and neural network training[10]. The UKF is also effective in certain types of visual contour hand tracking[11][12]. However, these systems dealt mostly with tracking position and did not take orientation into account.

Although the UKF has been applied to a wide range of estimation problems, to the best of our knowledge there has been no attempt to use it to improve human head or hand orientation tracking. Therefore, in this paper, we explore the potential benefits of the UKF over the more traditional EKF in human orientation estimation. We describe the results of an experimental study which examines the estimation accuracy of the EKF and UKF on both head and hand orientation represented with quaternions. Quaternions are a common way to represent rotations in tracking, robotics, and mechanical engineering because they are compact and avoid gimbal lock[13]. The results of our study indicate that, although the EKF and UKF have equivalent performance, the additional computational overhead of the UKF and the quasi-linear nature of the quaternion dynamics makes the EKF a more appropriate choice for orientation estimation in VR applications.

The remainder of this paper is organized as follows. In the next two sections, we describe the algorithmic details of the EKF and UKF formulations used in our study. Section IV describes our experimental methodology and setup. Section V presents the experimental results and discusses their significance. Section VI concludes the paper.

II. Extended Kalman Filtering

The extended Kalman filter is a set of mathematical equations which uses an underlying process model to make an estimate of the current state of a system and then corrects the

estimate using any available sensor measurements. Using this predictor-corrector mechanism, it approximates an optimal estimate due to the linearization of the process and measurement models[14]. To describe all the details of the EKF is beyond the scope of this paper. Therefore, we present a more algorithmic description omitting some theoretical considerations. More details on the EKF can be found in [15][16].

The process model we use is an orientation/angular velocity (OV) model defined by

$$f = \frac{dq}{dt} = \frac{1}{2} q\omega, \tag{1}$$

where q is the current quaternion and ω is a pure vector quaternion representing angular velocity. We use a single EKF, where the state vector at time k is defined by

$$\hat{\mathbf{x}}_k = [q_x, q_y, q_z, q_w, \omega_0, \omega_1, \omega_2]^T. \tag{2}$$

Given the state vector at step $k-1$, we first perform the prediction step by finding the *a priori* state estimate $\hat{\mathbf{x}}_k^-$ by integrating equation 1 through time by Δt (i.e., 1.0 divided by the current sampling rate) using a 4th Order Runge-Kutta scheme.

Then, we find the *a priori* estimate of the error covariance matrix

$$\mathbf{P}_k^- = \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T + \mathbf{Q}_k, \tag{3}$$

where $\mathbf{Q}_k$ is the process noise covariance, $\mathbf{P}_{k-1}$ is the *a posteriori* estimate of the error covariance, and $\mathbf{\Phi}_k$ is an approximation to the fundamental matrix calculated by taking the Taylor expansion of $\mathbf{\Phi}(\mathbf{t})$ around the system dynamics matrix

$$\mathbf{F}_{k,[i,j]} = \frac{\partial f_{(i)}}{\partial x_{(j)}}(\hat{\mathbf{x}}_k^-), \tag{4}$$

a Jacobian matrix which linearizes the process function f, and then substituting Δt for $\mathbf{t}$[1].

After the prediction step, the correction step calculates the *a posteriori* state estimate using

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-), \tag{5}$$

where $\mathbf{K}_k$ is the Kalman gain or blending factor and $\mathbf{H}_k$ is the measurement matrix used to combine the measurement vector $\mathbf{z}_k$, obtained from the tracking device, with $\hat{\mathbf{x}}_k^-$. The Kalman gain is computed using

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1}, \tag{6}$$

where $\mathbf{R}$ is the measurement noise covariance, and the measurement matrix is calculated using

$$\mathbf{H}_{k,[i,j]} = \frac{\partial h_{(i)}}{\partial x_{(j)}}(\mathbf{x}_k^-), \tag{7}$$

a Jacobian matrix that linearizes around the nonlinear measurement function h. In our case, h is quaternion normalization defined by

$$h = \frac{q}{\sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}} \tag{8}$$

for the quaternion in $\hat{\mathbf{x}}_k^-$. Finally, we compute the *a posteriori* estimate of the error covariance using

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-. \tag{9}$$

Note that after we calculate the *a posteriori* state estimate, the quaternion is renormalized ensuring it is on the unit sphere, making it a valid rotation.

A. EKF Parameters and Initialization

The EKF has two parameters, $\mathbf{Q}_k$ and $\mathbf{R}$, which represent the process noise covariance and the measurement noise covariance. $\mathbf{R}$ is determined empirically and accounts for the uncertainty in the tracking data. Setting these matrices properly goes a long way toward making the filters robust. We determine $\mathbf{Q}_k$ using the continuous process noise matrix $\tilde{\mathbf{Q}}$ which assumes that the process noise always enters the process model on the highest derivative[16]. Therefore,

$$\mathbf{Q}_k = \Phi_s \int_0^{\Delta t} \mathbf{\Phi}(\tau) \tilde{\mathbf{Q}} \mathbf{\Phi}(\tau)^T \, dt, \tag{10}$$

where Φ_s is a scaling parameter which acts as a confidence value for how sure we are that the process model is an accurate description of the the true motion dynamics.

The EKF also needs to be initialized on startup. The quaternion in the state vector at time 0 is simply set to the first observation in the motion sequence and the angular velocity components are set to 0. The *a priori* estimate of the error covariance and the elements in the these matrices are set to 0 for the off-diagonal entries and to relatively large numbers in the diagonal entries. For our implementation, the quaternion variance diagonals are set to 1 and the angular velocity variances are set to 100.

III. Unscented Kalman Filtering

The basic premise behind the unscented Kalman filter is it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function. Instead of linearizing using Jacobian matrices, the UKF uses a deterministic sampling approach to capture the mean and covariance estimates with a minimal set of sample points[9]. As with the EKF, we present an algorithmic description of the UKF omitting some theoretical considerations. More details can be found in [7][6][18].

Given the state vector at step $k-1$ (we use the same state vector as in equation 2, we compute a collection of sigma

[1]Note that from a theoretical perspective, the EKF calculates $\mathbf{F}_k$ each time f is evaluated. In the 4th order Runge-Kutta routine, f is evaluated 8 times[17], meaning that $\mathbf{F}_k$ should be a product of 8 intermediate Jacobian evaluations. In our formulation, we only evaluate $\mathbf{F}_k$ once from the output of the Runge-Kutta routine. Although this approach deviates slightly from the definition of the EKF, we find it faster, less complex, and works just as well for our applications.

points, stored in the columns of the $L \times (2L+1)$ sigma point matrix $\mathcal{X}_{k-1}$ where L is the dimension of the state vector. In our case, $L = 7$ so $\mathcal{X}_{k-1}$ is a 7×15 matrix. The columns of $\mathcal{X}_{k-1}$ are computed by

$$\begin{aligned}(\mathcal{X}_{k-1})_0 &= \hat{\mathbf{x}}_{k-1} \\ (\mathcal{X}_{k-1})_i &= \hat{\mathbf{x}}_{k-1} + \left(\sqrt{(L+\lambda)\mathbf{P}_{k-1}}\right)_i, i = 1 \dots L \\ (\mathcal{X}_{k-1})_i &= \hat{\mathbf{x}}_{k-1} - \left(\sqrt{(L+\lambda)\mathbf{P}_{k-1}}\right)_{i-L}, i = L+1 \dots 2L,\end{aligned} \tag{11}$$

where $\left(\sqrt{(L+\lambda)\mathbf{P}_{k-1}}\right)_i$ is the ith column of the matrix square root and λ is defined by

$$\lambda = \alpha^2(L+\kappa) - L, \tag{12}$$

where α is a scaling parameter which determines the spread of the sigma points and κ is a secondary scaling parameter. Note that we assume $\left(\sqrt{(L+\lambda)\mathbf{P}_{k-1}}\right)_i$ is symmetric and positive definite which allows us to find the square root using a Cholesky decomposition.

Once $\mathcal{X}_{k-1}$ computed, we perform the prediction step by first propagating each column of $\mathcal{X}_{k-1}$ through time by Δt using

$$(\mathcal{X}_k)_i = f((\mathcal{X}_{k-1})_i), i = 0 \dots 2L, \tag{13}$$

where f is differential equation defined in equation 1. In our formulation, since $L = 7$, we perform 15 4th order Runge-Kutta integrations.

With $(\mathcal{X}_k)_i$ calculated, the *a priori* state estimate is

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} (\mathcal{X}_k)_i, \tag{14}$$

where $W_i^{(m)}$ are weights defined by

$$\begin{aligned}W_0^{(m)} &= \frac{\lambda}{(L+\lambda)} \\ W_i^{(m)} &= \frac{1}{2(L+\lambda)}, i = 1 \dots 2L.\end{aligned} \tag{15}$$

As the last part of the prediction step, we calculate the *a priori* error covariance with

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} \left[(\mathcal{X}_k)_i - \hat{\mathbf{x}}_k^-\right] \left[(\mathcal{X}_k)_i - \hat{\mathbf{x}}_k^-\right]^T + \mathbf{Q}_k, \tag{16}$$

where $\mathbf{Q}_k$ is once again the process error covariance matrix, and the weights are defined by

$$\begin{aligned}W_0^{(c)} &= \frac{\lambda}{(L+\lambda)} + (1 - \alpha^2 + \beta) \\ W_i^{(c)} &= \frac{1}{2(L+\lambda)}, i = 1 \dots 2L.\end{aligned} \tag{17}$$

Note that β is a parameter used to incorporate any prior knowledge about the distribution of $\mathbf{x}$.

To compute the correction step, we first must transform the columns of $\mathcal{X}_k$ through the measurement function. Therefore, let

$$(\mathcal{Z}_k)_i = h((\mathcal{X}_k)_i), i = 0 \dots 2L \tag{18}$$

$$\hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} (\mathcal{Y}_k)_i. \tag{19}$$

h is the same quaternion normalization function found in equation 8.

With the transformed state vector $\hat{\mathbf{z}}_k^-$, we compute the *a posteriori* state estimate using

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k^-), \tag{20}$$

where $\mathbf{K}_k$ is once again Kalman gain. In the UKF formulation, $\mathbf{K}_k$ is defined by

$$\mathbf{K}_k = \mathbf{P}_{\hat{\mathbf{x}}_k \hat{\mathbf{z}}_k} \mathbf{P}_{\hat{\mathbf{z}}_k \hat{\mathbf{z}}_k}^{-1}, \tag{21}$$

where

$$\mathbf{P}_{\hat{\mathbf{z}}_k \hat{\mathbf{z}}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[(\mathcal{Z}_k)_i - \hat{\mathbf{z}}_k^-\right] \left[(\mathcal{Z}_k)_i - \hat{\mathbf{z}}_k^-\right]^T + \mathbf{R} \tag{22}$$

$$\mathbf{P}_{\hat{\mathbf{x}}_k \hat{\mathbf{z}}_k} = \sum_{i=0}^{2L} W_i^{(c)} \left[(\mathcal{X}_k)_i - \hat{\mathbf{x}}_k^-\right] \left[(\mathcal{Z}_k)_i - \hat{\mathbf{z}}_k^-\right]^T. \tag{23}$$

Note that as with the EKF, $\mathbf{R}$ is the measurement noise covariance matrix. Finally, the last calculation in the correction step is to compute the *a posteriori* estimate of the error covariance given by

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{\hat{\mathbf{z}}_k \hat{\mathbf{z}}_k} \mathbf{K}_k^T. \tag{24}$$

As with the EKF, we renormalize the state vector's quaternion to make sure it is on the unit sphere, making it a valid rotation.

A. UKF Parameters and Initialization

$\mathbf{Q}_k$, $\mathbf{R}$, α, β, and κ are the five parameters used in the UKF. We determine, $\mathbf{R}$, α, β, and κ empirically and use the formulation described in Section II.A to find $\mathbf{Q}_k$. More details on our choice for determining $\mathbf{Q}_k$ can be found in Section V. The UKF is initialized in the same way as the EKF, using the same values for the state vector and error covariance matrix upon startup.

IV. Experimental Study

To compare the performance of the EKF and UKF algorithms described in sections II and III, we conducted an experiment to determine which filtering algorithm is preferable for improving human orientation tracking in virtual reality systems.

A. Experimental Setup

Two datasets (one head and one hand) were used in our study to represent common orientation dynamics found in our virtual reality applications. Each dataset consists of unit length quaternions running about 20 seconds in length. The orientation sequences were captured using an Intersense IS900 tracking system, a hybrid inertial/ultrasonic tracking device. The head orientation dataset, denoted HEAD and shown in Figure 1, is an example of a user rotating her head to view images on three orthogonal display screens. The hand orientation dataset, denoted by HAND and shown in Figure 2, is an example of a user rotating his hand to navigate through the virtual world.

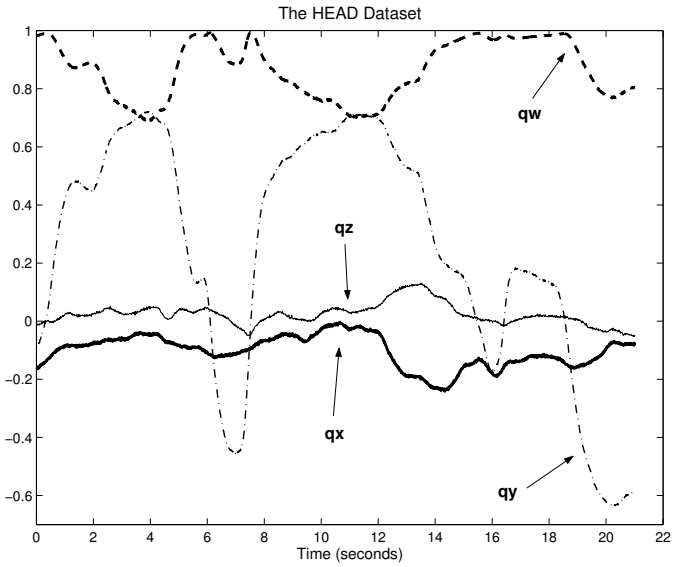

Fig. 1. The four signals that make up the quaternion sequence for the HEAD dataset. The values for each quaternion component are unitless.

In the experiment, the datasets were tested with sampling rates of 25, 80, and 215Hz giving us three different test scenarios for each dataset. These sampling rates were chosen because VR tracking systems are commonly run at these rates. We use a small Monte Carlo simulation on each test scenario since we have random Gaussian noise added to the motion signals, which is used to simulate jittery tracking data. A constant value of 5e-6 was set for the random noise variance providing noise added to the motion signals with a Gaussian distributed range of ± 1.19 degrees. All tests were run on a AMD Athelon XP 1800+ with 512Mb of main memory.

B. Evaluation Method

To determine how well the EKF and UKF algorithms are performing, we need comparison data. Comparing estimated output with reported user orientations is problematic since these records have noise and small distortions associated with them. Thus, any comparison with the recorded data would count tracking error with the estimation error. We obtain the "ground truth" datasets by passing them through a zero phase shift filter to remove high frequency noise. We determine the

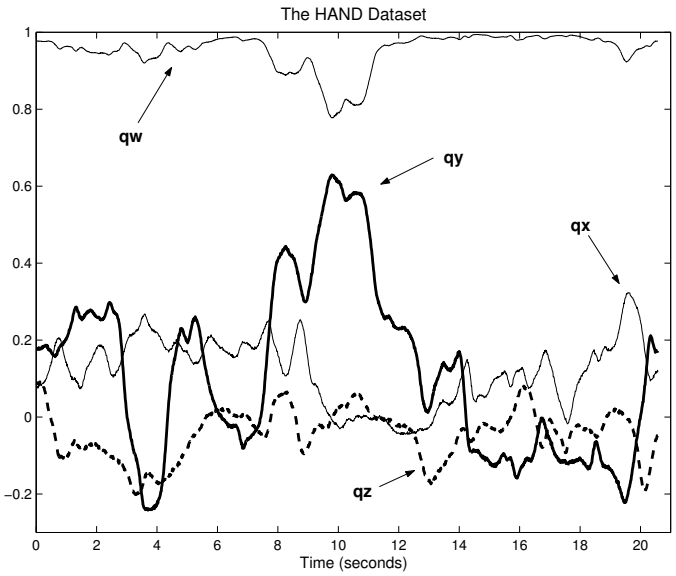

Fig. 2. The four signals that make up the quaternion sequence for the HAND dataset. The values for each quaternion component are unitless.

lowpass and highpass filter parameters by examining each signal's power spectrum. Depending on the particular dataset, the lowpass/highpass pairs were anywhere between 1/3 and 2/4 Hz. This cleaning step gives us the truth datasets we need to test against and makes it easy to add noise of known characteristics for simulating jittery tracking data. With the truth datasets, we can calculate the root mean square error (RMS) for each test and take the average over the Monte Carlo simulation runs. For truth and estimated quaternions, q_{t_i} and q_{e_i}, RMS is defined by

$$RMS_q = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} e_i^2}, \tag{25}$$

where RMS_q is in degrees and

$$e_i = \frac{2(180)}{\pi} \arccos\left((q_{t_i}(q_{e_i})^{-1})_w\right). \tag{26}$$

C. EKF and UKF Parameters

For the EKF and UKF algorithms, we needed to determine the $\mathbf{R}$ and $\mathbf{Q_k}$ covariance matrices. Since we know the variance of the Gaussian white noise we are injecting into the motion signals, we set the off-diagonal entries of $\mathbf{R}$ to 0 and set the diagonal entries to be the value of the noise variance value (5e-6 in this case). Thus we are making the assumption that our measurement noise is based on the variability of a stationary tracker. As shown in Section II.A, we calculate the $\mathbf{Q_k}$ matrix using equation 10 leaving Φ_s as our free parameter. The search routines ran over different integer values for Φ_s and we found 1 to be a good choice for the HEAD dataset and 2 for the HAND dataset. For the UKF, we also needed to set the α, β, and κ parameters. After running a number of tests, we found that 1,0, and 0 were appropriate for these

parameters. See Section V for a discussion on our parameter choices.

V. Results and Discussion

Tables I and II show the RMS errors for the HEAD and HAND datasets across the different sampling rates. These results show that the EKF and UKF have roughly the same error in all cases. Note that we also include the RMS error for doing no filtering at all to show that both the EKF and UKF improve tracking accuracy at sampling rates of 80 and 215Hz.

RMS Results for the HEAD Dataset			
	EKF	UKF	NONE
25Hz:	0.398973	0.431781	0.43614
80Hz:	0.297994	0.304475	0.4195
215Hz:	0.222815	0.228121	0.422076

TABLE I

THE RMS ERROR RESULTS (IN DEGREES) FOR THREE DIFFERENT SAMPLING RATES ON THE HEAD DATASET. THE DATA SHOWS THE EKF AND UKF HAVE ROUGHLY THE SAME ERROR WHEN ESTIMATING QUATERNIONS AND IMPROVE ACCURACY OVER NO FILTERING AT ALL.

RMS Results for the HAND Dataset			
	EKF	UKF	NONE
25Hz:	0.381808	0.386876	0.400181
80Hz:	0.308092	0.302395	0.389521
215Hz:	0.226587	0.23002	0.386043

TABLE II

THE RMS ERROR RESULTS (IN DEGREES) FOR THREE DIFFERENT SAMPLING RATES ON THE HAND DATASET. THE DATA SHOWS THE EKF AND UKF HAVE ROUGHLY THE SAME ERROR WHEN ESTIMATING QUATERNIONS AND IMPROVE ACCURACY OVER NO FILTERING AT ALL.

The tests that were run at 25Hz show there is only a slight improvement in the EKF and UKF's estimation performance for both the HEAD and HAND datasets. These numbers indicate that sampling rates of 25Hz are probably not high enough for applying filtering algorithms to quaternion motion data. However, more work is needed to verify this claim.

Figures 3 and 4 show the state errors from the EKF and UKF filters for the quaternion components in the HEAD dataset sampled at 80Hz. These graphs are representative of the component wise error in our test scenarios and show that, on a component level, the accuracy of the EKF and UKF are roughly the same. From this data and the data in Tables I and II, it is difficult to make a decision about which estimation algorithm is the better choice. Therefore, we need to examine the algorithms in greater detail.

Using the test scenarios, we recorded the running times for each algorithm. On average, the EKF algorithm took 266.13 microseconds per estimate while the UKF algorithm took 3,294.2 microseconds per estimate. The reason the UKF algorithm takes significantly longer to make an estimate is because it has to handle all the sigma points. In our implementation, the

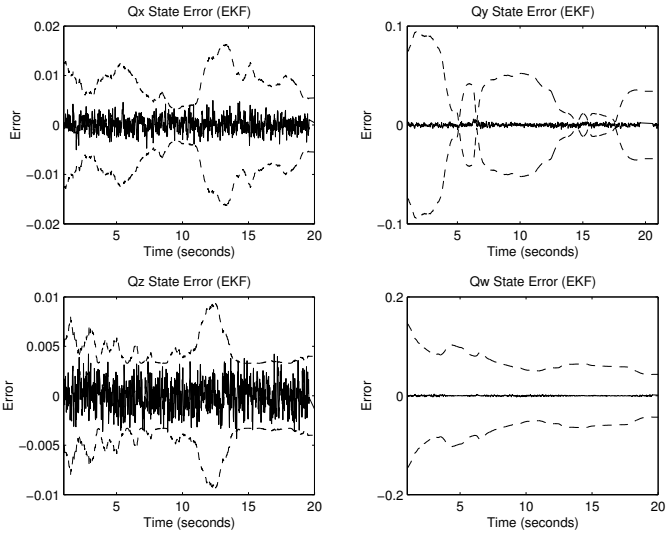

Fig. 3. State errors from the EKF for the four quaternion components in the HEAD dataset sampled at 80Hz. The solid lines represent the errors while the dashed lines show the 3 standard deviation bounds. The component estimates are unitless.

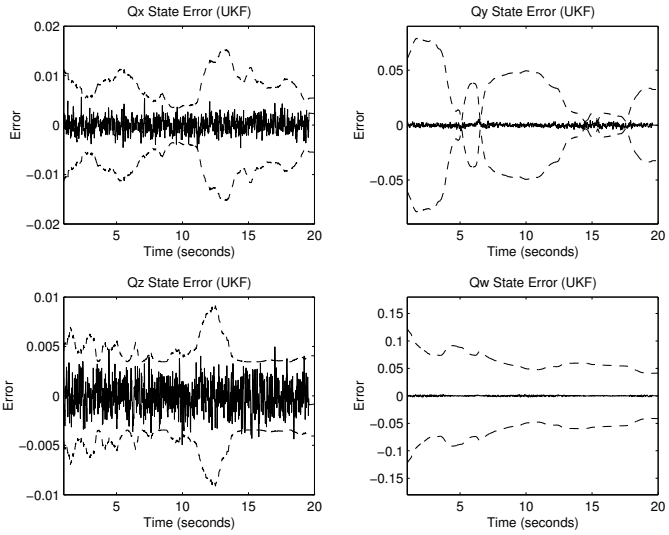

Fig. 4. State errors from the UKF for the four quaternion components in the HEAD dataset sampled at 80Hz. The solid lines represent the errors while the dashed lines show the 3 standard deviation bounds. The component estimates are unitless.

UKF has to perform 15 Runge-Kutta integrations to propagate the sigma points through time while the EKF only has to perform one integration. Even if we used Julier and Uhlmann's method for reducing the number of sigma points[19], we would still need to do 8 Runge-Kutta integrations for the UKF to only one for the EKF. If the estimation accuracy of the UKF was better than the EKF, this additional computational overhead would be warranted. However, since the UKF does not give us any additional accuracy, from a running time standpoint, the EKF seems the more appropriate estimator in this case.

In addition to the issue of time complexity between the

EKF and UKF, we also need to examine their theoretical performance. From [6], we know that UKF can predict the state estimate and error covariance to 4th order accuracy while the EKF only predicts up to 2nd order for the state estimate and 4th order for the error covariance. However, the UKF will make more accurate estimates only if the kurtosis and higher order moments in the state error distributions are significant. In our application, the magnitudes of the quaternion component covariances are significantly less than unity (on the order of 10^{-4} to 10^{-6} in most cases) which means the kurtosis and higher order moments are very small. This fact is one indication of why the UKF does not perform better than the EKF. Additionally, this indicates why there is little, if any, effect in UKF performance with different values for the UKF parameters α, β, and κ. Sampling rate is another indication why the UKF does not provide better performance when estimating quaternion motion. In general, if the sampling rate is sufficiently high, the quaternion dynamics behave in a quasi-linear fashion since, with small timesteps, the integration steps propagate the quaternions only small deviations away from the unit sphere, making the error in linearization minimal.

Finally, one of the main advantages of the UKF is that it does not require the calculation of Jacobian matrices. In many applications, Jacobian matrix evaluation can be non-trivial and lead to implementation difficulties[6]. However, in our application, the calculation of the Jacobian matrices is quite simple based on the structure of the process and measurement functions (see equations 1 and 8) and quaternion mathematics[20]. Therefore, the UKF does not provide us with any additional benefit in this case. In fact, the simplicity of the Jacobian calculations for the process model allowed us to use the same method for calculating $\mathbf{Q}_k$ in both the EKF and UKF formulations.

Although our work has focused on head and hand orientation tracking in VR applications, we hypothesize that these results may extend to other domains, such as robotics and underwater vehicle navigation, requiring quaternion motion estimation with motion dynamics that behave in a quasi-linear fashion. Such motion dynamics would have to have the important characteristic of small angle deviations and sampled at relatively high rates. Future work can validate this hypothesis.

VI. Conclusion

In this paper, we have presented an experiment which compares extended and unscented Kalman filtering of head and hand orientation data represented with quaternions. Our results indicate that, although the EKF and UKF have roughly the same accuracy, the computational overhead of the UKF, the simplicity of the Jacobian matrix calculations, and the quasi-linear nature of the quaternion dynamics makes the EKF a better choice for the task of improving tracking of noisy quaternion signals in virtual reality applications.

Acknowledgments

Special thanks to Simon Julier, Gary Bishop, Greg Welch, John Hughes, and Andy van Dam for valuable guidance and discussion. This work is supported in part by the NSF Graphics and Visualization Center, IBM, the Department of Energy, Alias/Wavefront, Microsoft, Sun Microsystems, and TACO.

References

[1] Stanney, Kay M. *Handbook of Virtual Environments: Design, Implementation, Applications*, Lawrence Erlbaum Associates, 2002.

[2] Azuma, Ronald and Gary Bishop. Improving Static and Dynamic Registration in a See-Through HMD. In *Proceedings of SIGGRAPH'94*, 197-204, 1994.

[3] Foxlin, Eric. Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. In *Proceedings of the Virtual Reality Annual International Symposium '96*, 185-194, 1996.

[4] Welch, Greg, and Gary Bishop. SCAAT: Incremental Tracking with Incomplete Information, In *Proceedings of SIGGRAPH'97*, ACM Press, 333-344, 1997.

[5] Sorenson, H. W. *Kalman Filtering: Theory and Application*, IEEE Press, 1985.

[6] Julier, Simon J., Jeffery K. Uhlmann, and Hugh F. Durrant-Whyte. A New Approach for Filtering Nonlinear Systems.In *Proceedings of the 1995 American Control Conference*, 1628-1632, 1995.

[7] Julier, Simon J. and Jeffery K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing,Simulation and Controls*, Multi Sensor Fusion, Tracking and Resource Management II, SPIE, 1997.

[8] Julier, Simon J. and H. F. Durrant-Whyte. Navigation and Parameter Estimation of High Speed Road Vehicles. In *Robotics and Automation Conference*, 101-105, 1995.

[9] Wan, E. A., and R. van der Merwe. The Unscented Kalman Filter for Nonlinear Estimation. In *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control(AS-SPCC)*, IEEE Press, 2000.

[10] van der Merwe, R. and E. A. Wan, Efficient Derivative-Free Kalman Filters for Online Learning, In *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2001.

[11] Peihua, Li and Tianwen Zhang. Unscented Kalman Filter for Visual Curve Tracking. In *Proceedings of Statistical Methods in Video Processing*, June, 2002.

[12] Stenger, B., P. R. S. Mendonça, and R. Cipolla. Model-Based Hand Tracking Using an Unscented Kalman Filter. In *Proceedings of the British Machine Vision Conference*, 63-72, September 2001.

[13] Grassia, F. Sebastian. Practical Parameterization of Rotations Using the Exponential Map. In *Journal of Graphics Tools*, 3(3):29-48, 1998.

[14] Welch, Greg and Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[15] Maybeck, Peter S. *Stochastic models, estimation, and control*. Volume 1, Academic Press, 1979.

[16] Zarachan, Paul and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. Progress in Astronautics and Aeronautics, Volume 190, American Institute of Aeronautics and Astronautics, Inc., 2000.

[17] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, 1993.

[18] Wan, E. A., and R. van der Merwe. The Unscented Kalman Filter, In *Kalman Filtering and Neural Networks*, S. Haykin (ed.), Wiley Publishing, 2001.

[19] Julier, Simon J., and Jeffrey K. Uhlmann. Reduced Sigma Point Filters for the Propagation of Means and Covariances Through Nonlinear Transformations. In *Proceedings of the 2002 American Control Conference*, 887-892, 2002.

[20] Shoemake, Ken. Animating Rotations with Quaternion Curves. In *Proceedings of SIGGRAPH 85*, ACM Press, 245-254, 1985.