Proceedings of the IEEE ITSC 2006
2006 IEEE Intelligent Transportation Systems Conference
Toronto, Canada, September 17-20, 2006

R-Adaptive Kalman Filtering Approach to Estimate Head Orientation for Driving Simulator

Henry Himberg, Yuichi Motai, Member, IEEE, and Cesar Barrios, Student Member, IEEE

Abstract - **Kalman filters are used to overcome system latency by predicting head orientation using AC magnetic tracker. To achieve optimum performance from the Kalman filter, the process and measurement covariance must be accurately set to tune the filter in the intelligent environment, *i.e.* driver assistive service. This paper discusses two adaptive Kalman filters, one using the well known fading memory algorithm and a new approach that adaptively tunes the filter to improve output signal quality. The two filters are tested in a head orientation prediction, and their performance is compared to a non-adaptive Kalman filter. The test results indicate that the adaptive tuning algorithm significantly reduces output noise in real-time head orientation for feasible applications of a driving simulator.**

I. Introduction

Head tracking devices such as AC magnetic tracker are widely used for innovative man-machine interfaces in the intelligent vehicle systems and virtually-simulated driving environments. For such an intelligent driver assistance service, the sensory outcomes of head tracking synchronously provide head localization, driving profile, and behavioral modeling in a vehicle. The perceived latency between head motion and computer/machine response, however, causes loss of immersion for the user, which can result in dizziness in extreme cases [10], [11], [14], [15], [16]. Additionally, noise in the head orientation data can result in a "swimming effect" that destroys the illusion of virtual reality [9].

The Extended Kalman Filter (EKF) is typically used to compensate for system latency [7]. The EKF can provide optimal estimation if the filter is correctly tuned to the incoming data. In head tracking applications, the large dynamic range of head motion and the unknown noise content of the incoming data require a compromise of the tuning parameters that results in less than optimum performance. Additionally, prediction amplifies the output error of the filter when the state vector is projected out in time. Adaptive filter techniques can be applied to the EKF to dynamically mitigate the undesirable behavior of the filter due to less than optimum tuning.

The Kalman filter can be made adaptive using a technique referred to as Fading Memory. Hu et al [8] used a fading memory algorithm based on residual averaging to improve dynamic performance in GPS vehicle navigation.

Zhang, Hu and Zhou [13] presented an optimal fading memory algorithm for human movement tracking that improved noise performance, frame prediction and noise performance but performance in multi-frame prediction applications was not addressed. Malkawi [12] uses a two-stage filter that improves noise performance but is not adaptable to changing input data.

In this paper, we evaluate the performance of two adaptive Kalman filters for head orientation prediction, one using a fading memory algorithm and one that is based on measurement model adaptive tuning. Unlike other adaptive techniques that optimize filter behavior by measuring the residual, the proposed **R-Adaptive Algorithm** uses a measurement of the predicted orientation to modify filter behavior. Extensive experimentation is performed with the two adaptive filters and a conventional EKF for different types of head motion and prediction times.

II. Background

The Extended Kalman Filter (EKF) is a prediction-correction filter used in systems with a stochastic difference equation (1) and measurement equation (2).

$$x_k = f\left(x_{k-1}, u_k, w_{k-1}\right) \tag{1}$$

$$z_k = h\left(x_k, v_k\right) \tag{2}$$

Equation 1 relates the state at time k to the state at time $k-1$ while (2) relates the measurement at time k to the state at time k. The process noise (w) and measurement noise (v) are assumed to be independent Gaussian random variable with zero mean [1], [2], [4], [5]. The EKF equations are developed for the non-linear case using a Taylor expansion to linearize the system about the current state.

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

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

The A, W, H and V Jacobian matrices are recomputed each time the filter iterates. The prediction equations estimate the next state (5) and the prediction error covariance (6).

$$\tilde{x}_k = f\left(x_{k-1}, u_k, 0\right) \tag{5}$$

$$\tilde{P}_k = A_k \cdot \hat{P}_{k-1} \cdot A_k^T + W_k \cdot Q \cdot W_k^T \tag{6}$$

1-4244-0094-5/06/$20.00 ©2006 IEEE

Note that the *a priori* state (5) assumes that the process noise is zero. The *a priori* error covariance (6) propagates the uncertainty forward in time.

The Correction equations compute the Kalman gain (7) to correct the state prediction (8) and error covariance (9).

$$K_k = \tilde{P}_k \cdot H_k^T \cdot \left(H_k \cdot \tilde{P}_k \cdot H_k^T + V_k \cdot R \cdot V_k^T\right)^{-1} \quad (7)$$

$$\hat{x}_k = \tilde{x}_k + K_k \cdot \left[z_k - h(\tilde{x}_k, 0)\right] \quad (8)$$

$$\hat{P}_k = (I - K_k \cdot H_k) \cdot \tilde{P}_k \quad (9)$$

III. Quaternion Filter Implementation

The Quaternion filter assumes that the sensor is moving with a constant angular velocity which represents the average value during the time between measurements [3]. Changes in sensor velocity are assumed to be random and are therefore modeled as process noise. The system is modeled as a normally distributed random process *X* with system state *x*. The system state consists of the current orientation (*q*) and angular velocity (ω) of the sensor.

$$x \equiv \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & \omega_0 & \omega_1 & \omega_2 \end{bmatrix}^T \equiv \begin{bmatrix} q & \omega \end{bmatrix}^T$$

The EKF estimates the system state as *x* with covariance matrix (*P*). The system output is modeled as a normally distributed random process *Z* with measurement *z*.

A. Process Model

The relationship between quaternion motion and angular velocity (*ω*) using quaternion multiplication is

$$\dot{q} = (1/2) \cdot \Omega \circ q \quad (10)$$

$$\Omega = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \quad (11)$$

Rearranging (10) in standard matrix form:

$$\dot{q} = \Psi \cdot q \quad (12)$$

$$\Psi = \frac{1}{2} \cdot \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & -\omega_z & \omega_y \\ \omega_y & \omega_z & 0 & -\omega_x \\ \omega_y & -\omega_y & \omega_x & 0 \end{bmatrix} \quad (13)$$

As shown in Goddard [4], the solution to this differential equation is

$$q(t_k) = e^{\Psi \cdot \tau} \cdot q(t_{k-1}) \quad (14)$$

Solving for the closed form we get:

$$q(t_k) = \begin{bmatrix} \cos((1/2) \cdot \|\Omega\| \cdot \tau) \cdot I \\ + (2/\|\Omega\|) \cdot \sin((1/2) \cdot \|\Omega\| \cdot \tau) \cdot \Psi(t_k) \end{bmatrix} \cdot q(t_{k-1}) \quad (15)$$

$$\tau = t_k - t_{k-1}$$

Switching to the discrete case with quaternion algebra:

$$q_k = \Delta q_k(\omega_{k-1}, \tau) \circ q_{k-1} \quad (16)$$

$$\Delta q_k(\omega, \tau) = \begin{bmatrix} \cos((1/2) \cdot \|\Omega\| \cdot \tau) \cdot I \\ + (2/\|\Omega\|) \cdot \sin((1/2) \cdot \|\Omega\| \cdot \tau) \cdot \Psi(t_k) \end{bmatrix} \quad (17)$$

Since the model assumes a constant angular velocity, the angular velocity state variables are propagated forward without change by the process model.

$$\omega_k = \omega_{k-1} \quad (18)$$

Combining the equations for x and ω, we get the process model as a function of the present state (x) and the process noise (w). The process noise consists of random angular accelerations denoted by the Ω matrix.

$$f(x, w) = \begin{bmatrix} q \\ \omega \end{bmatrix} = \begin{bmatrix} \Delta q(\omega + \Omega, \tau) \circ q \\ \omega \end{bmatrix} \quad (19)$$

The process model (19) predicts the next state from the previous state, and assumes that there is no process noise (angular acceleration).

The A matrix is the Jacobian of partial derivative of the state with respect to the previous state.

$$A_k = \frac{\partial}{\partial x_k} f(\hat{x}_{k-1}, 0, 0)) = \begin{bmatrix} \frac{\partial}{\partial q_{k-1}} q_k & \frac{\partial}{\partial \omega_{k-1}} q_k \\ \frac{\partial}{\partial q_{k-1}} \omega_k & \frac{\partial}{\partial \omega_{k-1}} \omega_k \end{bmatrix} \quad (20)$$

The partial derivative of quaternion state with respect to the previous quaternion state is the delta quaternion (17).

$$\frac{\partial}{\partial q_{k-1}} q_k = \frac{\partial}{\partial q_{k-1}} \left[\Delta q_k \circ q_{k-1}\right] = \Delta q_k \quad (21)$$

The partial derivative with respect to angular velocity is a three column matrix with each column being the partial with respect to one axis of angular acceleration.

$$\frac{\partial}{\partial \omega} q = \begin{bmatrix} \frac{\partial}{\partial \omega_x} q & \frac{\partial}{\partial \omega_y} q & \frac{\partial}{\partial \omega_z} q \end{bmatrix} \quad (22)$$

Starting with the definition of the quaternion (16), the partial derivative of the quaternion with respect to angular velocity is the partial derivative of the delta quaternion (17).

$$\frac{\partial}{\partial \omega} q_k = \frac{\partial}{\partial \omega} \left[\Delta q_k \circ q_{k-1}\right] = \left(\frac{\partial}{\partial \omega} \Delta q_k\right) \circ q_{k-1} \quad (23)$$

$$\begin{aligned} \frac{\partial}{\partial \omega_i} \Delta q_k &= -\left(\frac{\omega_i \cdot \tau}{2 \cdot \|\Omega\|} \cdot \sin\left(\frac{\|\Omega\| \cdot \tau}{2}\right)\right) \cdot I \\ &+ \frac{1}{\|\Omega\|^2} \left(\omega_i \cdot \tau \cdot \cos\left(\frac{\|\Omega\| \cdot \tau}{2}\right) - \frac{2 \cdot \omega_i}{\|\Omega\|} \cdot \sin\left(\frac{\|\Omega\| \cdot \tau}{2}\right)\right) \cdot \Psi \\ &+ \left(\frac{2}{\|\Omega\|} \cdot \sin\left(\frac{\|\Omega\| \cdot \tau}{2}\right) \cdot \frac{\partial}{\partial \omega_i} \Psi\right) \end{aligned} \quad (24)$$

Equation 24 can be simplified by replacing its elements with the elements of Δq (17).

$$\frac{\partial}{\partial \omega_i}\Delta q = -(\tau/2)\cdot \Delta q_{i+1}\cdot I + \left(1/\|\Omega\|^2\right)\cdot\left[\omega_i\cdot\tau\cdot\Delta q_0 - 2\cdot\Delta q_{i+1}\right]\cdot\Psi + (2/\omega_i)\cdot\Delta q_{i+1}\cdot\frac{\partial}{\partial\omega_i}\Psi \quad (25)$$

The partial derivative of the current state with respect to angular velocity (26) is found by substituting (25) in to (16).

$$\frac{\partial}{\partial \omega_i} q_k = (-(\tau/2)\cdot(\Delta q_k)_{i+1}\cdot I + \left(1/\|\Omega\|\right)\cdot\left[\omega_i\cdot\tau\cdot(\Delta q_k)_0 - 2\cdot(\Delta q_k)_{i+1}\right]\cdot\Psi + (2/\omega_i)\cdot(\Delta q_k)_{i+1}\cdot\frac{\partial}{\partial\omega_i}\Psi)\cdot q_{k-1} \quad (26)$$

The partial derivative of the angular velocity with respect to previous quaternion state is 0.

$$\frac{\partial}{\partial q_{k-1}}\omega_k = \frac{\partial}{\partial q_{k-1}}\omega_{k-1} = 0 \quad (27)$$

The partial derivative of the angular velocity with respect to the previous state is the identity matrix.

$$\frac{\partial}{\partial \omega_{k-1}}\omega_k = \frac{\partial}{\partial \omega_{k-1}}\omega_{k-1} = I \quad (28)$$

The Jacobian matrix *W* contains the partial derivatives of the state with respect to the process noise. The process noise (*w*) consists of changes in the angular velocity due to acceleration.

$$W_k = \frac{\partial}{\partial w} f(\hat{x}_{k-1},0,0) = \frac{\partial}{\partial\Omega}\begin{bmatrix}\Delta q_k(\omega_{k-1}+\Omega,\tau)\circ q_{k-1}\\ \omega_{k-1}+\Omega\end{bmatrix}_{\Omega=0} \quad (29)$$

Closer inspection of (29) indicates it is equivalent to the partial with respect to velocity (26).

B. Measurement Model

The H matrix is the Jacobian of partial derivatives of the measurement model with respect to state. The quaternion orientation is both a state variable and the measurement therefore the measurement equation is linear.

$$H_k = \frac{\partial}{\partial x_k}h(\tilde{x}_k,0)) = \begin{bmatrix}\frac{\partial}{\partial q_k}h(\tilde{x}_k,0) & \frac{\partial}{\partial\omega_k}h(\tilde{x}_k,0)\end{bmatrix} \quad (30)$$

$$H_k = \begin{bmatrix}\frac{\partial}{\partial q_k}[q_k+v]_{v=0} & \frac{\partial}{\partial\omega_k}[q_k+v]_{v=0}\end{bmatrix} = [I \quad 0] \quad (31)$$

The *V* matrix is the Jacobian of partial derivative of the measurement model with respect to measurement noise. Since the measurement model is linear, *V* is a 4x4 identity matrix.

$$V_k = \frac{\partial}{\partial v_k}h(\tilde{x}_k,0) = \frac{\partial}{\partial v_k}[q_k + v_k]_{v_k=0} = I \quad (32)$$

C. Orientation Prediction

The filter maintains a current estimate of the sensor orientation by predicting the next state with the system model and correcting it with the measurement data. The future orientation is estimated by computing the delta quaternion (Δq) that results if the current angular velocity (ω_k) is maintained over the prediction interval (τ_p) and applying it to the quaternion state estimate (8).

$$q_k^P = \Delta q_k(\hat{x}_k,\tau_P)\circ H\cdot\hat{x}_k \quad (33)$$

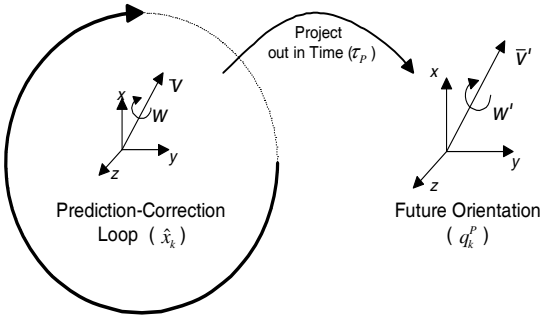

Figure 1. Orientation Prediction

D. Adaptive Filtering

Adaptive filtering techniques are typically applied to the Kalman filter by modifying the Kalman gain (K) in response to a measurement parameter. In the conventional Kalman filter, K is computed as the ratio of the predicted measurement error covariance to the residual covariance.

$$K = \frac{\tilde{P}\cdot H^T}{\left(H\cdot\tilde{P}\cdot H^T + R\right)^{-1}} \equiv \frac{E[\tilde{e},\tilde{e}^T]\cdot H^T}{\left(E[r,r^T]\right)^{-1}}$$

The Kalman gain determines the weighting used to combine the system model with the measurement data in the computation of the corrected state estimate. The residual (the difference between the predicted measurement and the actual measurement) is multiplied by K to correct the predicted state estimate. Increasing K increases the effect of measurement data on the corrected state estimate, forcing the filter to follow the measurement data more closely. Decreasing K decreases the influence of the measurement data in favor of the system model.

E. Fading Memory Algorithm

The fading memory algorithm adapts the EKF to the incoming data set by applying a scalar gain (S_k) to the predicted error covariance (Fig. 2).

$$\tilde{P}_k = S_k\cdot\left(A_k\cdot\hat{P}_{k-1}\cdot A_k^T + W_k\cdot Q\cdot W_k^T\right) \quad (34)$$

Increasing $\tilde{P}$ increases the weight applied to the measurement data by increasing the Kalman gain, allowing it to respond more quickly to changes in the measurement data. Decreasing $\tilde{P}$ decreases the Kalman gain, forcing the system to follow the model more closely. Changes in $\tilde{P}$ effect the Kalman gain and the

corrected state estimate (8) and are propagated forward in time through the corrected error covariance (9).

The Fading memory algorithm uses a sliding window to detect changes in the residual component. The average residual is computed as the RMS average of the residual over the previous N filter iterations (35). The algorithm then computes the scalar gain (S) as the ratio of the vector norm of current residual to the vector norm of the average residual (36). The scalar gain is limited to a minimum value of 1 to allow non-adaptive filtering (near optimal) when the residual is decreasing from the previous average.

$$\Psi_k = \sqrt{(1/N)\cdot\sum_{i=1}^{N}\left(z_{k-i} - H\cdot\tilde{x}_{k-i}\right)^2} \quad (35)$$

$$S_k = \left(\left(z_k - H\cdot\tilde{x}_k\right)^T\cdot\left(z_k - H\cdot\tilde{x}_k\right)\right)\Big/\left(\Psi_k^T\cdot\Psi_k\right) \quad (36)$$

The residual value increases as the filter diverges from the system model causing an increase in the scalar gain and consequently, an increase in the predicted error covariance. The increased $\tilde{P}$ decreases the weight of the system model and allows the filter to more closely track the measurement data, reducing the divergence.

F. R-Adaptive Algorithm

The R-Adaptive algorithm tries to maintain a maximum filter output noise level by applying a scalar gain (α) to the measurement noise covariance (*R*).

$$K_k = \tilde{P}_k\cdot H^T\cdot\left(H\cdot\tilde{P}_k\cdot H^T + \cdot\alpha_k\cdot R\right)^{-1} \quad (37)$$

When the EKF is used for prediction, the filter output is a projection in time of the current state estimate. Corrections to the estimated state vector are magnified by the time projection, resulting in noisy filter output when process noise is high. The R-Adaptive algorithm controls the Kalman gain response to the output ripple level of the filter, closing the loop around the projection of the quaternion orientation (Fig. 3).

The value of *K* is computed as the ratio of the predicted measurement error to the expected residual. The prediction error is relatively constant whenever the system model is an accurate predictor of system behavior. In this situation, small changes in the Kalman gain do not cause large residuals that would significantly impact the prediction error, allowing the Kalman gain to be controlled through the measurement covariance. The R-Adaptive uses a scalar gain (α) to increases the measurement covariance when the output ripple is above a preset threshold, thereby reducing the Kalman gain and the magnitude of the state estimate correction. If the ripple is below the threshold, the scalar gain is set to 1, reverting to the conventional EKF.

The quaternion projection (33) is linear least-square curve fit in a sliding window of size *N* (38) where φ is the intercept and δ is the slope. The RMS average output noise is estimated as the difference between the fitted curve and the quaternion projection (39). The scaling coefficient is computed as the ratio of vector norm of the estimated average RMS noise in the quaternion output to a preset filter parameter (40). The quaternion output is linear least-square curve fit in a sliding window of size *N* and the RMS value of the difference between the fitted curve and the quaternion data is computed.

$$\lambda(i) = \varphi - \delta\cdot i \quad (38)$$

$$\xi_k = \sqrt{(1/N)\cdot\sum_{i=0}^{N-1}\left[\lambda(i) - q^P_{(k-N+1)+i}\right]^2} \quad (39)$$

$$\alpha_k = \sqrt{\xi_k^T\cdot\xi_k}\Big/\beta \quad (40)$$

The scaling coefficient is limited to a minimum of unity to allow normal operation of the EKF when the estimated noise is less than the preset value (β).

IV. Experimental Results

The three filter types, Non-adaptive, Fading, and R-Adaptive algorithm were constructed in the MathCAD environment for various amounts of prediction. The filters were evaluated for noise suppression and their ability to accurately track (or predict) the input data.

A. Benchmark Input Data

Quaternion head orientation data was collected for slow (D1), moderate (D2) and rapid (D3) head movement using a Polhemus Liberty AC magnetic tracker operating at a 240 Hz frame rate. The collected data was curve fit using spline interpolation to create a benchmark data set for the simulation. The benchmark data set was then multiplied by a normal distribution function to create data sets with AC noise level at -20 dB, -40dB, -60dB, -80 dB and -100 dB. The benchmark data closely follows the profile of the measured data while providing an input data set that can be precisely represented by a known function.

B. Performance Criteria

Filter performance was rated by measuring the SNR and RMS accuracy of the output data while the filter operated on input data of a known SNR. The SNR of the filter output was computed as the vector norm of the RMS average of the difference between the output data and a C-spline function fitted to the data.

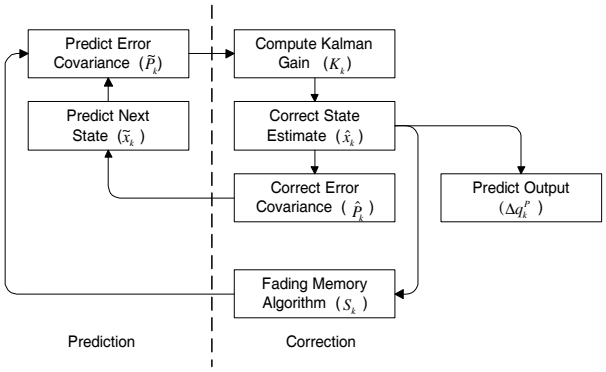

Figure 2 Fading Memory Flow Diagram

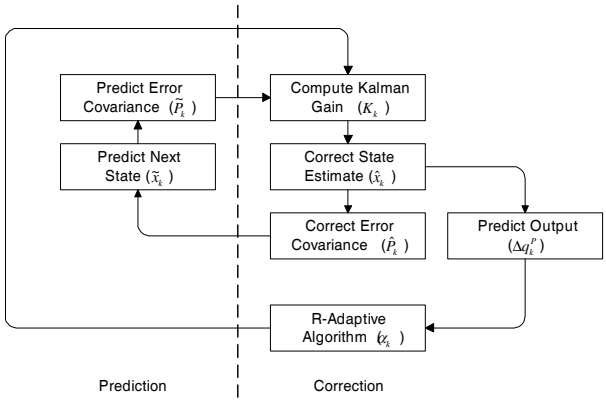

Figure 3 R-Adaptive Flow Diagram

The RMS accuracy of the filter output was calculated as the vector norm of the RMS average of the difference between a C-spline function fitted to the output data and the benchmark data used to create the input data set. The RMS accuracy is expressed in degrees of error to facilitate a more intuitive measure of the filters ability to track the input data.

C. R-Adaptive Filter Tuning

The beta parameter (β) was determined through experimentation by running the filter with a sliding window size of 12 at three prediction intervals (0ms, 50ms and 100ms). The SNR of the input data was varied from 100dB down to 20dB over a spread of β values as the output SNR and RMS accuracy were measured. The output SNR, RMS error and max error were averaged over 100 simulations of the filter at each combination of input noise, β value and prediction time.

Fig. 4 shows the effect of β on the output signal is dependent on the quality of the input signal. Reducing the value of β reduces the Kalman gain, causing the model to weight the predicted state more heavily than the measured data. Poor quality input data results in improved performance as compared to the non-adaptive case since the system model has less error than the input data. However, if the input data is of high quality, the reduced weight of the input data results in lower SNR than that of the non-adaptive EKF. The experimental data shows that for a given value of β, the output SNR of the filter levels off despite increasing SNR of the input data. When the input signal is better then 50dB SNR, increasing β decreases the output SNR. If the input signal is worse than 50dB SNR, increasing β increases the output SNR.

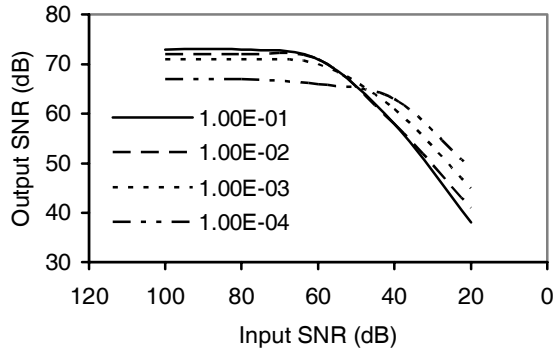

Figure 4 R-Adaptive Output SNR vs. Beta with 0 ms prediction

In addition to the effect on the output SNR, β also impacts the RMS output error of the filter. Fig. 5 illustrates how the output error increases as the value of β is decreased. The algorithm reduces output noise by reducing the influence of the measurement data, forcing the filter to diverge somewhat. If the β value is too high, the filter will diverge from the measurement data resulting in large errors in the filter output. This behavior is clearly seen in the data when the RMS output error increases by approximately 100% as β is increased from 1^{-03} to 1^{-04}.

The effect of β on output SNR and RMS error must be balanced to give a filter that achieves the desired noise reduction while not diverging from the true input signal too much. The collected data indicates that a β of 1e-03 results in improved SNR with a 40dB input SNR without losing much when the input SNR is 60dB or above. The RMS error is seen to increase as β is reduced below 1^{-02}, roughly doubling with each order of magnitude decrease.

TABLE 1 OUTPUT SNR VS. BETA

Prediction Time	Beta Value	Input Data SNR (dB)				
		100	80	60	40	20
0	1.00E-01	73	73	71	58	38
	1.00E-02	72	72	71	58	41
	1.00E-03	71	71	70	61	45
	1.00E-04	67	67	66	63	49
50ms	1.00E-01	68	68	66	52	33
	1.00E-02	67	67	66	53	39
	1.00E-03	64	64	64	58	43
	1.00E-04	63	63	63	60	48
100ms	1.00E-01	65	65	63	49	31
	1.00E-02	63	63	62	51	37
	1.00E-03	61	61	61	56	42
	1.00E-04	60	60	60	58	47

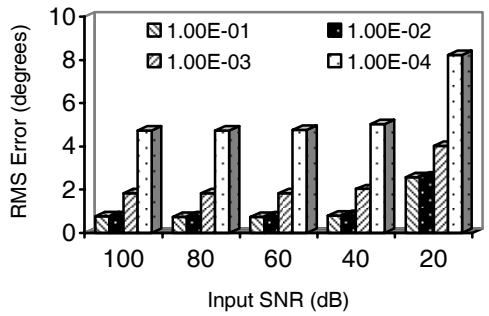

Figure 5 R-Adaptive RMS error vs. Beta with 0 ms of prediction

When β is set to 1^{-02} or below, the RMS error is essentially unchanged, indicating that the adaptive algorithm is not significantly effecting the filter operation. Based on the collected data, the optimal value for β is 1^{-03} for this application. The sliding window size was optimized for selected β value using a similar procedure and found to give best results at 12 samples. This result suggests that the β value is dependent on the window size as would be expected. Given the real-time aspect of the application, the window was fixed at 12 samples to reduce computation requirements.

D. Fading Memory Filter Window Size

The performance of the fading memory algorithm is dependent on the size of the sliding window used in the algorithm. To determine the optimum window size for head tracking, the filter performance was examined across a range of input noise levels, prediction intervals and window sizes. Each combination of window size, input noise and prediction interval was simulated 100 times and the average output SNR, RMS and max error computed.

The tuning experiment indicated that window size is inversely proportional to the output SNR but directly proportional to the RMS error. As the window size was increased, the SNR dropped significantly, as much as 20dB when using 100ms of prediction. The window size had minimal effect on RMS error except when exceedingly poor input data was used (20dB). Large window size (24 frames; 100ms) reduced RMS error slightly while also reducing the output SNR significantly. Small window size (3 frames; 12ms) gave the best output SNR along with a small increase in RMS error. Output SNR and RMS error showed little or no improvement with input SNR better than 60dB. The lack of improvement with better input data is most likely due to the fixed measurement covariance (R) that was chosen when the filter was initially tuned. Based on the experimental data, the window size was fixed at 3 frames for the comparison study.

E. Performance Comparison

A conventional Kalman filter (Non-adaptive), and two adaptive filters (Fading and R-Adaptive) were tested on each of the input data sets with increasing amounts of noise (20dB, 40dB, 50dB, 60dB) and prediction (τ_P; 0ms, 50ms, 100ms). The test was performed with an EKF

TABLE 2 RMS ERROR VS. BETA

Prediction Time	Beta Value	Input Data SNR (dB)				
		100	80	60	40	20
0	1.00E-01	0.75	0.75	0.75	0.79	2.56
	1.00E-02	0.77	0.77	0.77	0.84	2.61
	1.00E-03	1.82	1.82	1.83	2.02	4.04
	1.00E-04	4.75	4.74	4.78	5.05	8.24
50ms	1.00E-01	2.59	2.59	2.59	2.63	4.76
	1.00E-02	2.68	2.68	2.68	2.93	5.13
	1.00E-03	4.37	4.37	4.40	4.79	7.51
	1.00E-04	8.40	8.40	8.46	8.86	12.78
100ms	1.00E-01	5.35	5.35	5.35	5.37	7.66
	1.00E-02	5.54	5.54	5.55	6.03	8.66
	1.00E-03	7.85	7.86	7.90	8.48	11.80
	1.00E-04	12.93	12.93	12.97	13.52	18.02

tuned specifically for head tracking with $Q = 10^{-6}*I$ and $R=10^{-4}*I$. The Non-Adaptive filter performed very well with high quality input data (60dB) but output SNR decreased rapidly with increasing input noise. The RMS output error was essentially flat across all values of prediction and input data SNR until below 40dB. Below 40dB the filter diverged, resulting in large RMS errors and poor output SNR gain. Except for the 60dB input case, filter performance was essentially unchanged as head motion increased from slow to fast. With 60dB input SNR, output SNR dropped slightly with increasing head motion indicating that process noise due to acceleration was the dominant error.

The Fading Memory filter showed increased RMS error as compared to the Non-Adaptive filter when measurement noise was the dominant source of error. With D1 or D2 as the input data, there was very little process noise in the data therefore increasing the variance did not improve filter tune but forced the filter to weight the system model more heavily. The D1 and D2 data sets are a good match for the constant velocity system model so the reduced weighting of the input data allowed the filter to diverge slightly, improving output SNR but also increasing the RMS error. When fast head motion was used (D3), the Fading Memory algorithm suffered a loss in output SNR and increased RMS error. The filter was unable to adapt to the large residuals which occur when the filter model is a poor fit to the input data.

The improved SNR with the R-Adaptive algorithm illustrated that it is effective for situations when the system model is accurate. The small accelerations in the D1 and D2 data sets closely approximate a system with constant velocity, resulting in relatively constant values for the predicted error covariance and the residual covariance. The algorithm adjusts the R matrix to more closely approximate the actual measurement noise covariance of the input data, resulting in improved performance (Fig 6). When fast head motion was used (D3), the filter maintained similar output SNR to that of the D1 and D2 simulations but had a large increase in the RMS error. The reduction in gain during accelerations caused the filter to diverge from the input data slightly and overshoot the input data waveform (Fig. 7). In most instances the overshoot was slightly greater than the Non-Adaptive filter but high accelerations caused substantial overshoot, especially with prediction.

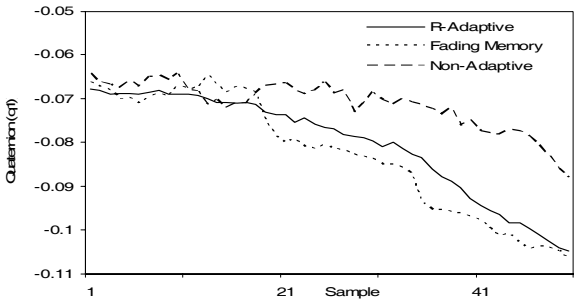

Figure 6 Slow Head Motion Comparison (D1; 40dB; 100ms)

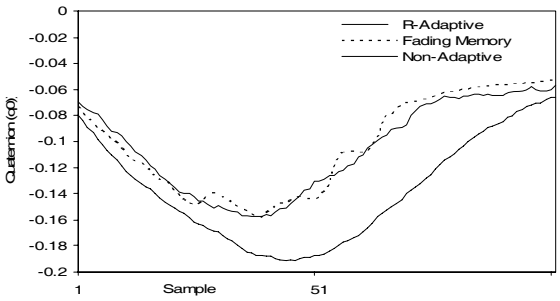

Figure 7 Fast Head Motion Comparison (D3; 40dB; 100ms)

TABLE 3 OUTPUT SNR (dB)

Data	τ_p (ms)	Input SNR (dB)											
		Non-Adaptive				Fading Memory				R Adaptive			
		20	40	50	60	20	40	50	60	20	40	50	60
D1	0	38	58	67	78	45	65	71	77	45	61	69	78
	50	33	52	62	78	37	56	64	77	43	59	65	78
	100	29	49	58	69	33	52	61	66	42	57	63	71
D2	0	38	58	67	76	45	64	68	72	46	61	69	77
	50	32	52	62	76	37	55	61	72	43	58	65	77
	100	29	49	58	68	33	51	57	62	42	57	63	70
D3	0	38	57	66	71	44	56	60	66	45	61	67	70
	50	33	52	61	66	36	49	54	60	43	58	63	64
	100	29	49	58	63	32	46	50	56	42	56	60	60

TABLE 4 RMS OUTPUT ERROR (DEGREES)

Data	τ_p (ms)	Input SNR (dB)											
		Non-Adaptive				Fading Memory				R Adaptive			
		20	40	50	60	20	40	50	60	20	40	50	60
D1	0	2	0	0	0	3	0	0	0	2	0	0	0
	50	4	0	0	0	6	1	0	0	2	0	0	0
	100	6	1	0	0	8	1	1	0	2	1	0	0
D2	0	3	0	0	0	4	1	0	0	2	0	0	0
	50	4	1	1	0	6	1	1	0	2	1	1	0
	100	6	1	1	1	8	1	1	1	3	1	1	1
D3	0	3	1	1	1	4	1	1	1	4	2	2	2
	50	5	3	3	3	7	3	2	2	8	5	5	4
	100	8	5	5	5	10	5	5	5	12	9	8	8

V. CONCLUSIONS

In this paper, we have presented two adaptive Kalman filtering methods for head orientation prediction in a driving simulator, both of which modify the Kalman gain to control filter behavior. The first approach, Fading Memory, adjusts the filter variance in response to changes in the residual component. The second approach, R-Adaptive, modifies the measurement covariance as a function of output noise in a sliding window. A Non-Adaptive Kalman filter was implemented along with the two adaptive filters and extensive simulation was performed on each filter for prediction time, input noise and motion model. The experimental results indicated that both adaptive algorithms improved output SNR for slow to moderate head motion. The Fading algorithm improved output SNR but had increased RMS error when using noisy input data. The filter showed reduced SNR when working on fast head motion data but had nearly the same RMS error as the conventional filter in most cases. The R-Adaptive filter provided significantly improved SNR and RMS error when working with slow head motion. Fast head motion causes larger RMS error in the R-Adaptive filter but had little effect on output SNR improvements.

Additional work needs to be performed to reduce the overshoot that occurs when large prediction times are used. Potential changes include increasing the sample rate of the tracker (from 240 Hz to 480 Hz or higher) and/or extension of the Kalman filter to include acceleration.

A. Acknowledgment

The authors acknowledge Polhemus Inc, Colchester, Vermont, U.S.A. for the support and time for the study.

B. References

[1] G. Welch, G. Bishop, "An introduction to the kalman filter", SIGGRAPH 2001, Course notes,. http://www.cs.unc.edu/

[2] P. Maybeck, "Stochastic models, estimation and control", Academic Press, 1979.

[3] J. Bohg, "Real-time structure from motion using Kalman filtering", PhD Dissertation, 2005. http://www.phenomene.de/bohg/

[4] J. S. Goddard Jr. "Pose and motion estimation from vision using dual quaternion-based extended kalman filtering", Proc. of SPIE, Vol. 3313, 1998.

[5] E V Stansfield, "Introduction to kalman filters",tutorial, 2001

[6] R. F. Stengel, "Stochastic optimal control", Wiley and Sons, 1986.

[7] A. van Rhijn, R. van Liere, J. Mulder, "An analysis of orientation prediction and filtering methods for VR/AR", Proc. of the IEEE Virtual Reality Conference, pp. 67-74, 2005.

[8] C. Hu, W. Chen, Y. Chen, D. Liu, "Adaptive kalman filtering for vehicle navigation", Journal of Global Positioning Systems, Vol. 2, No, 1, pp.42-47, 2003.

[9] J. Wu, M. Ouhyoung, "On latency compensation and its effects on head-motion trajectories in virtual environments", The Visual Computer, vol. 16, pp. 79-90, 2000.

[10] R. H.-Y. So, "Lag compensation by image deflection and prediction: a review on the potential benefits to virtual training applications for manufacturing industry", Proc. of the 7th Int. Conference on Human-Computer Interaction, 1997.

[11] R. H. Y. So, W. T. Lo, A. T.K. Ho, "Effects of navigational speed on motion sickness caused by an immersive virtual environment", Human Factors, vol. 43, no. 3, pp. 452-461, 2001.

[12] A. Malkawi, R. Srinivasan., "Building performance visualization using augmented reality", Proc. of Int'l Conf. on Computer Graphics, GRAPHICON, Russia, 2004

[13] Y. Zhang, H. Hu, H. Zhou, "Study on adaptive Kalman filtering algorithms in human movement tracking", Proc. of IEEE Int. Conference on Information Acquisition, 2005.

[14] J. Liang, C. Shaw, M. Green, "On temporal-spatial realism in the virtual reality environment", Proc. of Annual Symposium on User Interface Software and Technology, pp 19-25, 1991.

[15] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, J. E. Zacher, "Tolerance of temporal delay in virtual environments", IEEE Virtual Reality Conference, pp. 247, 2001.

[16] J.Y.Jung, B. D. Adelstein, S. R. Ellis, "Discriminability of prediction artifacts in a time-delayed virtual environment", Proc. of IEA/HFES, pp. 499-502, 2000.