A Frequency-Domain Analysis of Head-Motion Prediction

Ronald Azuma[§]
Hughes Research Laboratories

Gary Bishop[†]
University of North Carolina at Chapel Hill

Abstract

The use of prediction to eliminate or reduce the effects of system delays in Head-Mounted Display systems has been the subject of several recent papers. A variety of methods have been proposed but almost all the analysis has been empirical, making comparisons of results difficult and providing little direction to the designer of new systems. In this paper, we characterize the performance of two classes of head-motion predictors by analyzing them in the frequency domain. The first predictor is a polynomial extrapolation and the other is based on the Kalman filter. Our analysis shows that even with perfect, noise-free inputs, the error in predicted position grows rapidly with increasing prediction intervals and input signal frequencies. Given the spectra of the original head motion, this analysis estimates the spectra of the predicted motion, quantifying a predictor's performance on different systems and applications. Acceleration sensors are shown to be more useful to a predictor than velocity sensors. The methods described will enable designers to determine maximum acceptable system delay based on maximum tolerable error and the characteristics of user motions in the application.

CR Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism -- *virtual reality*
Additional Key Words and Phrases: Augmented Reality, delay compensation, spectral analysis, HMD

1 Motivation

A basic problem with systems that use Head-Mounted Displays (HMDs), for either Virtual Environment or Augmented Reality applications, is the end-to-end system delay. This delay exists because the head tracker, scene generator, and communication links require time to perform their tasks, causing a lag between the measurement of head location and the display of the corresponding images inside the HMD. Therefore, those images are displayed later than they should be, making the virtual objects appear to "lag behind" the user's head movements. This hurts the desired illusion of immersing a user inside a stable, compelling, 3-D virtual environment.

One way to compensate for the delay is to predict future head locations. If the system can somehow determine the future head position and orientation for the time when the images will be displayed, it can use that future location to generate the graphic images, instead of using the measured head location. Perfect predictions would eliminate the effects of system delay. Several predictors have been tried; examples include [2] [4] [5] [10] [11] [13] [17] [18] [19] [20].

Since prediction will not be perfect, evaluating how well predictors perform is important. Virtually all evaluation so far has been empirical, where the predictors were run in simulation or in real time to generate the error estimates. Therefore, no simple formulas exist to generate the values in the error tables or the curves in the error graphs. Without such formulas, it is difficult to tell how prediction errors are affected by changes in system parameters, such as the system delay or the input head motion. That makes it hard to compare one predictor against another or to evaluate how well a predictor will work in a different HMD system or with a different application.

2 Contribution

This paper begins to address this need by characterizing the theoretical behavior of two types of head-motion predictors. By analyzing them in the frequency domain, we derive formulas that express the characteristics of the predicted signal as a function of the system delay and the input motion. These can be used to compare predictors and explore their performance as system parameters change.

Frequency-domain analysis techniques are not new; Section 3 provides a quick introduction. The contribution here lies in the application of these techniques to this particular problem, the derivation of the formulas that characterize the behavior of the predictors, and the match between these frequency-domain results and equivalent time-domain results for collected motion data.

To our knowledge, only one previous work characterizes head-motion prediction in the frequency domain [18]. This paper builds upon that work by deriving formulas for two other types of predictors and exploring how their performance changes as system parameters are modified.

The two types of predictors were selected to cover most of the head-motion predictors that have been tried. Many predictors are based upon state variables: the current position x, velocity v, and sometimes acceleration a. Solving the differential equations under the assumption of constant velocity or acceleration during the entire prediction interval results in polynomial expressions familiar from introductory mechanics classes. Let the system delay (or prediction interval) be p. Then:

$$x_{predicted} = x + v\,p + \frac{1}{2} a\,p^2 \quad \text{or} \quad x_{predicted} = x + v\,p$$

The first type of predictor, covered in Section 4, uses the 2nd-order polynomial, under the assumption that position, velocity, and acceleration are perfectly measured.

In practice, real systems directly measure only a subset of position, velocity, and acceleration, so many predictors combine the polynomial expression with a Kalman filter to estimate the non-measured states. We know of no existing system that directly measures all three states for orientation, and linear rate sensors to measure translational velocity do not exist. The Kalman filter is an algorithm that estimates the non-measured states from the other

§ 3011 Malibu Canyon Road MS RL96; Malibu, CA 90265
(310) 317-5151 azuma@isl.hrl.hac.com
† CB 3175 Sitterson Hall; Chapel Hill, NC 27599-3175
(919) 962-1886 gb@cs.unc.edu

measurements and smoothes the measured inputs. Section 5 derives formulas for three different combinations of Kalman filters and polynomial predictors. The combinations depend on which states are measured and which are estimated. These form the second class of predictor explored.

Section 6 uses the formulas from Sections 4 and 5 to provide three main results:

1) Quantifying error distribution and growth: The error in the predicted signal grows both with increasing frequency and prediction interval. For the 2nd-order polynomial, the rate of growth is roughly the square of the prediction interval and the frequency. This quantifies the "jitter" commonly seen in predicted outputs, which comes from the magnification of relatively high-frequency signals or noise. For the Kalman-based predictors, we compare the three combinations and identify the frequencies where one is more accurate than the others. Theoretically, the most accurate combination uses measured positions and accelerations.

2) Estimating spectrum of predicted signal: Multiplying the spectrum of an input signal by the magnitude ratio determined by the frequency-domain analysis provides a surprisingly good estimate of the spectrum of the predicted signal. By collecting motion spectra exhibited in a desired application, one can use this result to determine how a predictor will perform.

3) Estimating peak time-domain error in predicted signal: Multiplying the input signal spectrum by the error ratio function generates an estimate of the error signal spectrum. Adding the absolute value of all the magnitudes in the error spectrum generates a rough estimate of the peak time-domain error. A comparison of estimated and actual peak errors is provided. With this, a system designer can specify the maximum allowable time-domain error and then determine the system delays that will satisfy that requirement for a particular application.

This paper is a short version of chapter 6 of [1]. That chapter is included with the CD-ROM version of this paper.

3 Approach

The frequency-domain analysis draws upon linear systems theory, spectral analysis, and the Fourier and Z-transforms. This section provides a brief overview; for details please see [3] [9] [12] [14] [15] [16].

Functions and signals are often defined in the time domain. A function $f(t)$ returns its value based upon the time t. However, it is possible to represent the same function in the frequency domain with a different set of basis functions. Converting representations is performed by a transform. For example, the Fourier transform changes the representation so the basis functions are sinusoids of various frequencies. When all the sinusoids are added together, they result in the original time-domain function. The Z-transform, which is valid for evenly-spaced discrete functions, uses basis functions of the form z^k, where k is an integer and z is a complex number. Specific examples of equivalent functions in the time, Fourier, and Z domains are listed in Table 1. Note that j is the square root of —1 and ω is the angular frequency. A function in the Fourier domain is indexed by ω, which means the coefficients representing the energy in the signal are distributed by frequency instead of by time, hence the name "frequency domain."

The analysis in this paper makes three specific assumptions. First, the predictor must be linear. A basic result of linear systems theory states that any sinusoidal input into a linear system results in an output of another sinusoid of the same frequency, but with different magnitude and phase. If the input is the sum of many different sinusoids (e.g., a Fourier-domain signal), then it is possible to compute the output by taking each sinusoid, changing its magnitude and phase, then summing the resulting output sinusoids, due to the property of superposition. This makes it possible to completely characterize linear systems by describing how the

	Time domain	Fourier domain
Linearity	$A g(t) + B h(t)$	$A G(\omega) + B H(\omega)$
Time shift	$g(t + a)$	$e^{j\omega a}\, G(\omega)$
Differentiation	$\dfrac{\partial g(t)}{\partial t}$	$j\omega\, G(\omega)$
	Time domain	Z domain
Linearity	$A\,x(k) + B\,y(k)$	$A\,X(z) + B\,Y(z)$
Time shift	$x(k - a)$	$z^{-a}\ X(z)$

Table 1: Time, Fourier and Z domain equivalents

magnitude and phase of input sinusoids transform to the output as a function of frequency. This characterization is called a transfer function, and these are what we will derive in Sections 4 and 5.

The second assumption is that the predictor separates 6-D head motion into six 1-D signals, each using a separate predictor. This makes the analysis simpler. The assumptions of linearity and separability are generally reasonable for the translation terms, but not necessarily for the orientation terms. For example, quaternions are neither separable nor linear [2]. To use this analysis, we must locally linearize orientation around the current orientation before each prediction, assuming the changes across the prediction interval are small. By using the small angle assumption, rotations can be characterized by linear yaw, pitch, and roll operations where the order of the operations is unimportant. Another approach for linearizing orientation is described in [8].

Finally, the third assumption is that the input signal is measured at evenly-spaced discrete intervals. This is not always true in practice, but this assumption does not really change the properties of the predictor as long as the sampling is done significantly faster than the Nyquist rate, and it makes the analysis easier.

What does the ideal predictor look like as a transfer function? Ideal prediction is nothing more than shifting the original signal in time. If the original signal is $g(t)$ and the prediction interval is p, then the ideal predicted signal $h(t) = g(t+p)$. By the timeshift formula in Table 1, the magnitude is unchanged, so the magnitude ratio is one for all frequencies, but the phase difference is $p\omega$.

What do input head motion signals look like in the frequency domain? The power spectrum shows the averaged squared magnitudes of the coefficients to the basis sinusoids at every frequency. The square root of those values is the average absolute values of the magnitudes. Figure 1 shows such a spectrum for one translation axis. This data came from recording a user who had never been inside an HMD before, while the user walked through a virtual museum of objects. Note that the vast majority of energy is below 2 Hz, which is typical of most other data we have and corroborates data taken by [19]. This is one way to quantify how quickly or slowly people move their heads. These spectra are application dependent, but note that the equations derived in Sections 4 and 5 are independent of the specific input spectrum. Faster head motions have spectra with more energy at higher frequencies.

Estimating the power spectrum of a time-domain signal is an inherently imperfect operation. Careful estimates require the use of frequency windows to reduce leakage [6]. Even with such steps, the errors can be significant. What this means is that the theoretical results in Section 6 that use estimated power spectra do not always perfectly match time-domain results from simulated or actual data. Please see [7] and [16] for details.

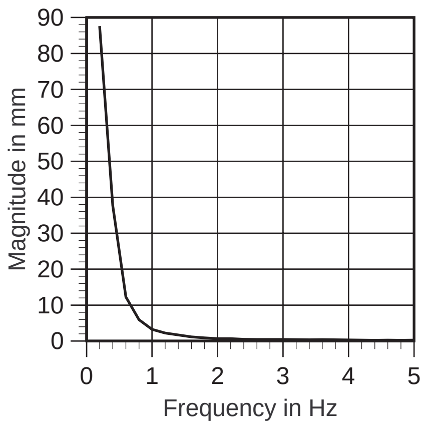


Figure 1: Head motion spectrum

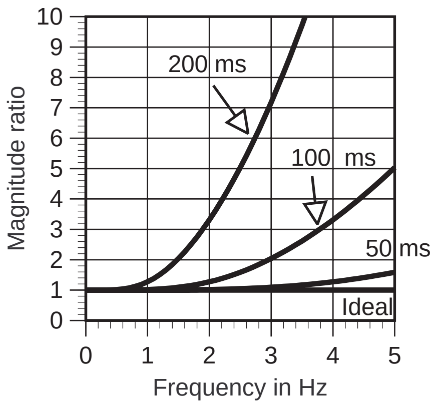


Figure 2: Polynomial predictor magnitude ratio

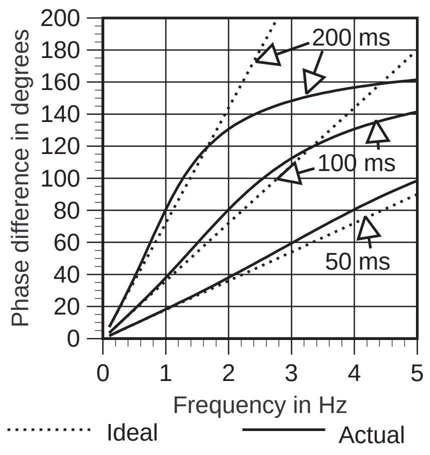


Figure 3: Polynomial predictor phase shift

4 Polynomial-based predictor

This section derives a transfer function that characterizes the frequency-domain behavior of a 2nd-order polynomial predictor. This analysis assumes that the current position, velocity, and acceleration are *perfectly* known, with no noise or other measurement errors. Even with perfect measurements, we will see that this predictor does not match the ideal predictor at long prediction intervals or high frequencies.

Let $g(t)$ be the original 1-D signal and $h(t)$ be the predicted signal, given prediction interval p. Then the 2nd-order polynomial predictor defines $h(t)$ as:

$$h(t) = g(t) + p\,g'(t) + \tfrac{1}{2}p^2 g''(t)$$

Convert this into the Fourier domain. $G(\omega)$ is the Fourier equivalent of $g(t)$. At any angular frequency ω, $G(\omega)$ is a single complex number, which we define as $G(\omega) = x + j\,y$. Then:

$$H(\omega) = \left(1 + j\,\omega\,p - \tfrac{1}{2}(\omega\,p)^2\right)(x + j\,y)$$

The transfer function specifies how the magnitude and phase change from the input signal, $G(\omega)$, to the output signal, $H(\omega)$. These changes are in the form of a magnitude ratio and a phase difference.

4.1 Magnitude ratio

We know the magnitude of the input signal. We need to derive the magnitude of the output signal. The squared magnitude of $H(\omega)$, after some simplification, is:

$$\|H(\omega)\|^2 = \left(x^2 + y^2\right)\left(1 + \tfrac{1}{4}(\omega\,p)^4\right)$$

Therefore, the magnitude ratio is:

$$\boxed{\frac{\|H(\omega)\|}{\|G(\omega)\|} = \sqrt{\left(1 + \tfrac{1}{4}(\omega\,p)^4\right)}} \qquad (1)$$

Figure 2 graphs equation (1) for three prediction intervals: 50 ms, 100 ms, and 200 ms. The ideal ratio is one at all frequencies, because the ideal predictor is simply a timeshift, but the actual predictor magnifies high frequency components, even with perfect measurements of position, velocity, and acceleration.

4.2 Phase difference

The phase α of the predicted signal $H(\omega)$ is:

$$\alpha = \tan^{-1}\left(\frac{\omega\,p\,x + y - \tfrac{1}{2}y\,p^2\,\omega^2}{x - y\,\omega\,p - \tfrac{1}{2}x\,p^2\,\omega^2}\right)$$

Let $\o$ be the phase of original signal $G(\omega) = x + j\,y$. Apply the following trigonometric identity:

$$\tan(\alpha - \o) = \frac{\tan(\alpha) - \tan(\o)}{1 + \tan(\alpha)\tan(\o)}$$

After simplification, the phase difference is:

$$\boxed{\alpha - \o = \tan^{-1}\left(\frac{p\,\omega}{1 - \tfrac{1}{2}(p\,\omega)^2}\right)} \qquad (2)$$

Figure 3 graphs equation (2) for three prediction intervals: 50 ms, 100 ms, and 200 ms, with the phase differences plotted in degrees. Note that the ideal difference is a straight line, and that the ideal difference changes with different prediction intervals. The actual phase differences follow the ideal only at low frequencies, with the error getting bigger at large prediction intervals or large frequencies. The phase differences asymptotically approach 180 degrees.

Note the intimate relationship between p and ω in the formulas in Sections 4.1 and 4.2; they always occur together as $\omega\,p$. This suggests a relationship between input signal bandwidth and the prediction interval. Halving the prediction interval means that the signal can double in frequency while maintaining the same prediction performance. That is, bandwidth times the prediction interval yields a constant performance level.

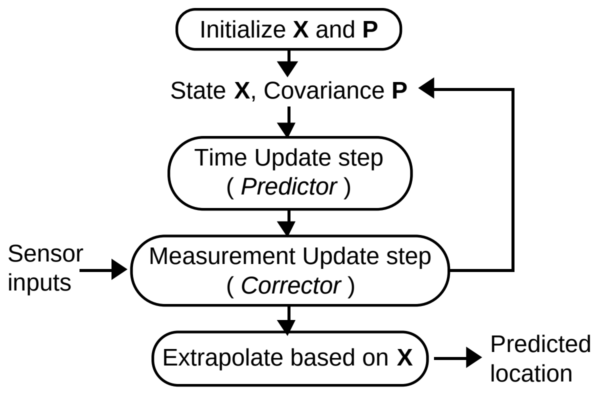
5 Kalman-based predictors

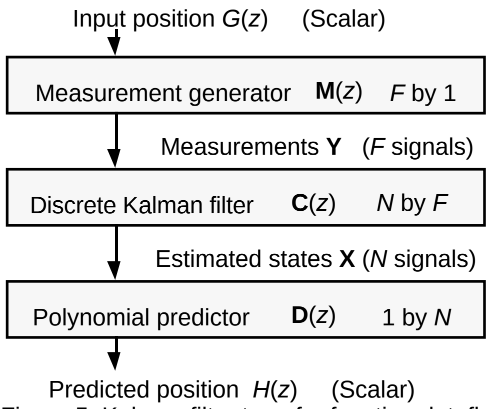
Real systems directly measure only a subset of p, v, and a, and those measurements are corrupted by noise. Therefore, many predictors use the Kalman filter to provide estimates of the states p, v, and a in the presence of noise. These estimated states are then given to the polynomial-based predictor to extrapolate future locations.

This section provides a high-level introduction on how the Kalman filter works, then it derives the Kalman predictor transfer matrix. This transfer matrix is the product of three other matrices, modeling the measurements, the predictor, and the Kalman filter itself. These matrices depend upon the type of filter and predictor being used. We derive the transfer matrix for three cases:

- Case 1: Measured position. Predictor based on x and v.
- Case 2: Measured position and velocity. Predictor based on x, v, and a.
- Case 3: Measured position and acceleration. Predictor based on x, v, and a.

Case 1 is typical of most predictors that have been tried, being solely based on the measurements from the head tracker. This


Initialize **X** and **P**
State **X**, Covariance **P**
Time Update step (*Predictor*)
Sensor inputs
Measurement Update step (*Corrector*)
Extrapolate based on **X**
Predicted location
Figure 4: Kalman filter high-level dataflow


Input position *G*(*z*) (Scalar)
Measurement generator **M**(*z*) *F* by 1
Measurements **Y** (*F* signals)
Discrete Kalman filter **C**(*z*) *N* by *F*
Estimated states **X** (*N* signals)
Polynomial predictor **D**(*z*) 1 by *N*
Predicted position *H*(*z*) (Scalar)
Figure 5: Kalman filter transfer function dataflow

predictor does not use acceleration because it is difficult to get a good estimate of acceleration from position in real time. Numerical differentiation accentuates noise, so performing two differentiation steps is generally impractical. A few predictors use inertial sensors to aid prediction, such as [2] [5] [11]. These sensors are used in Case 2 and Case 3. Section 6 will compare these three cases against each other, using the transfer functions derived in this section.

Throughout this section, x is position, v is velocity, a is acceleration, p is the prediction interval, T is the period separating the evenly-spaced inputs, and k is an integer representing the current discrete iteration index.

5.1 The Discrete Kalman filter

The Kalman filter is an optimal linear estimator that minimizes the expected mean-square error in the estimated state variables, provided certain conditions are met. It requires a model of how the state variables change with time in the absence of inputs, and the inaccuracies in both the measurements and the model must be characterizable by white noise processes. In practice, these conditions are seldom met, but the Kalman filter is commonly used anyway because it tends to perform well even with violated assumptions and because it has an efficient recursive formulation, suitable for computer implementation. Efficiency is important because the filter must operate in real time to be of any use to the head-motion prediction problem. This section outlines the basic operation of the filter; for details please see [3] [9]. Since the inputs are assumed to arrive at discrete, evenly-spaced intervals, the type of filter used is the Discrete Kalman filter.

Figure 4 shows the high-level operation of the Kalman filter. The Kalman filter maintains two matrices, **X** and **P**. **X** is an N by 1 matrix that holds the state variables, like x, v, and a, where N is the number of state variables. **P** is an N by N covariance matrix that indicates how accurate the filter believes the state variables are. After initialization, the filter runs in a loop, updating **X** and **P** for each new set of sensor measurements. This update proceeds in two steps, similar in flavor to the predictor-corrector methods commonly used in numerical integrators. First, the time update step must estimate, or predict, the values of **X** and **P** at the time associated with the incoming sensor measurements. Then the measurement update step blends (or corrects) **X** and **P** based on the sensor measurements. Whenever a prediction is required, the polynomial extrapolation bases it on the x, v, and a from the current state **X**.

5.2 Kalman-based predictor transfer matrix

The following discussion is terse due to space limitations; please read [1] for a more thorough explanation.

The goal is to derive a 1 by 1 transfer matrix $\mathbf{O}(z)$ relating input position $G(z)$ to predicted position $H(z)$. Figure 5 shows how this is done by combining the Discrete Kalman filter with the polynomial predictor. $\mathbf{O}(z)$ is the product of three other transfer matrices:

$$\boxed{H(z) = \mathbf{O}(z)G(z), \text{where } \mathbf{O}(z) = \mathbf{D}(z)\mathbf{C}(z)\mathbf{M}(z)}$$

$\mathbf{O}(z)$ is different for each of the three cases. The matrices needed to compute $\mathbf{O}(z)$ for each case are listed in Sections 5.3 to 5.5. Once computed, a basic result from control theory states that one can plot $\mathbf{O}(z)$'s frequency response by substituting for z as follows [14]:

$$z = e^{(j\omega T)} = \cos(\omega T) + j\sin(\omega T)$$

Note that z is a complex number, so the matrix routines must be able to multiply and invert matrices with complex components. Now we describe how to derive the three component transfer matrices $\mathbf{M}(z)$, $\mathbf{D}(z)$, and $\mathbf{C}(z)$.

1) Measurement generator transfer matrix $\mathbf{M}(z)$: The transfer function developed in Section 4 does not apply here because the Kalman filter treats the estimated states and measurements as separate and distinct signals. Therefore, the predictor transfer function is now a 1 by N matrix that specifies how to combine the state variables to perform the polynomial-based prediction. If the predicted position is a function of more than one measurement, rather than just a measured position, the analysis becomes complicated. To simplify things, we force the measurements to be perfectly matched with each other so that everything can be characterized solely in terms of the input position. This is enforced by the measurement generator $\mathbf{M}(z)$, which generates v and a from x by applying the appropriate magnitude ratios and phase shifts to the input position. If an input position x sinusoid is defined as:

$$x = M\sin(\omega t + \varnothing)$$

Then the corresponding velocity and acceleration sinusoids are:

$$v = \omega M\cos(\omega t + \varnothing)$$

$$a = -\omega^2 M\sin(\omega t + \varnothing)$$

For example, a is derived from x simply by multiplying by $-\omega^2$, as listed in the $\mathbf{M}(z)$ in Section 5.5.

2) Polynomial predictor transfer matrix $\mathbf{D}(z)$: This expresses the behavior of the polynomial-based predictor, as described in Section 4. For example, if the state is based on x, v, and a, then the predictor multiplies those by 1, p, and $0.5p^2$ respectively, as listed in the $\mathbf{D}(z)$ in Section 5.4.

3) Discrete Kalman filter transfer matrix $\mathbf{C}(z)$: Deriving a transfer matrix that characterizes the frequency-domain behavior of the Discrete Kalman filter requires that the filter operate in steady-state mode. This will occur if the noise and model characteristics do not change with time. This is usually the case in the Kalman-based predictors that have been used for head-motion prediction. In our implementation, the filter converges to the steady-state condition with just one or two seconds of input data, depending on the noise and model parameters and the initial **P**.

In the steady-state condition, **P** becomes a constant, so it is not necessary to keep updating it. This makes the equations for the time and measurement updates much simpler. The time update becomes:

$$\mathbf{X}^{-}(k+1)=\mathbf{A}\mathbf{X}(k)$$

and the measurement update becomes:

$$\mathbf{X}(k+1)=\mathbf{X}^{-}(k+1)+\mathbf{K}\left[\mathbf{Y}(k+1)-\mathbf{H}\mathbf{X}^{-}(k+1)\right]$$

where

- $\mathbf{Y}(k)$ is an F by 1 matrix holding F sensor measurements.
- $\mathbf{A}$ is an N by N matrix that specifies the model.
- $\mathbf{H}$ is an F by N matrix relating the measurements to the state variable $\mathbf{X}$.
- $\mathbf{K}$ is the N by F Kalman gain matrix that controls the blending in the measurement update.
- $\mathbf{X}^{-}(k+1)$ is the partially updated state variable.

Note that only the $\mathbf{X}(k)$ and $\mathbf{Y}(k)$ matrices change with time. Now combine the time and measurement update equations into one by solving for $\mathbf{X}(k+1)$:

$$\mathbf{X}(k+1)=\mathbf{A}\mathbf{X}(k)+\mathbf{K}\left[\mathbf{Y}(k+1)-\mathbf{H}\mathbf{A}\mathbf{X}(k)\right]$$

$$\mathbf{X}(k+1)=\left[\mathbf{A}-\mathbf{K}\mathbf{H}\mathbf{A}\right]\mathbf{X}(k)+\mathbf{K}\mathbf{Y}(k+1)$$

Convert this equation into the Z-domain and solve for $\mathbf{X}(z)$.

$$z\mathbf{X}(z)=\left[\mathbf{A}-\mathbf{K}\mathbf{H}\mathbf{A}\right]\mathbf{X}(z)+z\mathbf{K}\mathbf{Y}(z)$$

$$\mathbf{X}(z)=\left[z\mathbf{I}-\mathbf{A}+\mathbf{K}\mathbf{H}\mathbf{A}\right]^{-1}z\mathbf{K}\mathbf{Y}(z)$$

where $\mathbf{I}$ is the N by N identity matrix.

Define $\mathbf{C}(z)$ as the N by F transfer matrix for the Discrete Kalman filter. This specifies the relationship between the filter's inputs (measurement **Y**) and the outputs (state **X**):

$$\boxed{\mathbf{X}(z)=\mathbf{C}(z)\mathbf{Y}(z)\,\text{where}\,\mathbf{C}(z)=\left[z\mathbf{I}-\mathbf{A}+\mathbf{K}\mathbf{H}\mathbf{A}\right]^{-1}z\mathbf{K}}$$

This equation shows how to compute $\mathbf{C}(z)$ from the **A**, **K**, and **H** matrices listed in Sections 5.3 to 5.5. The steady-state **K** matrices in those three sections depend on the noise parameters used to tune the Kalman filter. We adjusted those parameters to provide a small amount of lowpass filtering on the state variable corresponding to the last sensor input in the **Y** matrix. Then we ran each filter in simulation to determine the steady-state **K** matrices.

5.3 Case 1: Measured position

$N = 2$, $F = 1$

$$\mathbf{X}=\begin{bmatrix}x\\v\end{bmatrix},\ \mathbf{H}=\begin{bmatrix}1&0\end{bmatrix},\ \mathbf{Y}=\begin{bmatrix}x_{measured}\end{bmatrix}$$

$$\mathbf{A}=\begin{bmatrix}1&T\\0&1\end{bmatrix},\ \mathbf{K}=\begin{bmatrix}0.568\\41.967\end{bmatrix},\mathbf{M}(z)=\begin{bmatrix}1\end{bmatrix},\mathbf{D}(z)=\begin{bmatrix}1&p\end{bmatrix}$$

5.4 Case 2: Measured position and velocity

$N = 3$, $F = 2$

$$\mathbf{X}=\begin{bmatrix}x\\v\\a\end{bmatrix},\mathbf{H}=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix},\mathbf{Y}=\begin{bmatrix}x_{measured}\\v_{measured}\end{bmatrix}$$

$$\mathbf{A}=\begin{bmatrix}1&T&\frac{1}{2}T^2\\0&1&T\\0&0&1\end{bmatrix},\mathbf{K}=\begin{bmatrix}0.0576&0.0032\\0.0034&0.568\\-0.0528&41.967\end{bmatrix}$$

$$\mathbf{M}(z)=\begin{bmatrix}1\\j\omega\end{bmatrix},\mathbf{D}(z)=\begin{bmatrix}1&p&\frac{1}{2}p^2\end{bmatrix}$$

5.5 Case 3: Measured position and acceleration

$N = 3$, $F = 2$

$$\mathbf{X}=\begin{bmatrix}x\\v\\a\end{bmatrix},\mathbf{H}=\begin{bmatrix}1&0&0\\0&0&1\end{bmatrix},\mathbf{Y}=\begin{bmatrix}x_{measured}\\a_{measured}\end{bmatrix}$$

$$\mathbf{A}=\begin{bmatrix}1&T&\frac{1}{2}T^2\\0&1&T\\0&0&1\end{bmatrix},\mathbf{K}=\begin{bmatrix}0.0807&0.000016\\0.342&0.00345\\0.0467&0.618\end{bmatrix}$$

$$\mathbf{M}(z)=\begin{bmatrix}1\\-\omega^2\end{bmatrix},\mathbf{D}(z)=\begin{bmatrix}1&p&\frac{1}{2}p^2\end{bmatrix}$$

6 Results

This section takes the transfer functions derived in Sections 4 and 5 and uses them to determine three characteristics of the predictor: 1) the distribution of error in the predicted signal, 2) the spectrum of the predicted signal, and 3) the peak time-domain error. The frequency-domain results are checked against time-domain results, where appropriate.

6.1 Error distribution and growth

We can now plot the prediction error for both the polynomial predictor and the Kalman-based predictors.

1) Polynomial predictor: Figure 6 graphs the overall error behavior of the polynomial-based predictor, using the transfer functions derived in Section 4. The plot shows the errors at three different prediction intervals. The errors grow rapidly with increasing frequency or increasing prediction interval.

The overall error is a Root-Mean-Square (RMS) metric. A problem with showing the error of the predicted signal in the frequency domain is the fact that the transfer functions return *two* values, magnitude ratio and phase shift, rather than just one value. Both contribute to the error in the predicted signal. If the magnitude ratio is large, then that term dominates the error, but it is not wise to ignore phase at low magnitude ratios. An RMS error metric captures the contribution from both terms. Pick an angular frequency ω. Let M_r be the magnitude ratio at that frequency, $ø$ be the difference between the transfer function's phase shift and the ideal predictor's phase shift, and T be the period of the frequency. Then define the RMS error at that frequency to be:

$$RMS_{error}(\omega,ø,M_r)=\sqrt{\frac{1}{T}\int_0^T\left[M_r\sin(\omega t+ø)-\sin(\omega t)\right]^2dt}$$

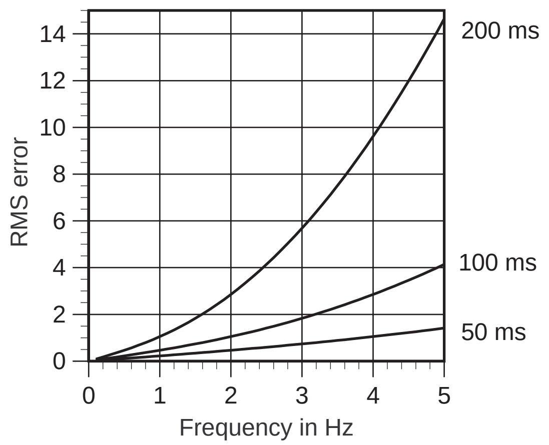


Figure 6: RMS error for polynomial predictor at three prediction intervals

Sinusoid = $M \sin(2\pi f t + \phi)$: Mag. M, Freq. f, Phase ϕ

#	Magnitude	Frequency (in Hz)	Phase (in radians)
1	5.0	1.0	0.5
2	3.0	2.0	-1.7
3	1.0	5.0	-0.3
4	0.01	60.0	0.2

Table 2: Four sinusoids

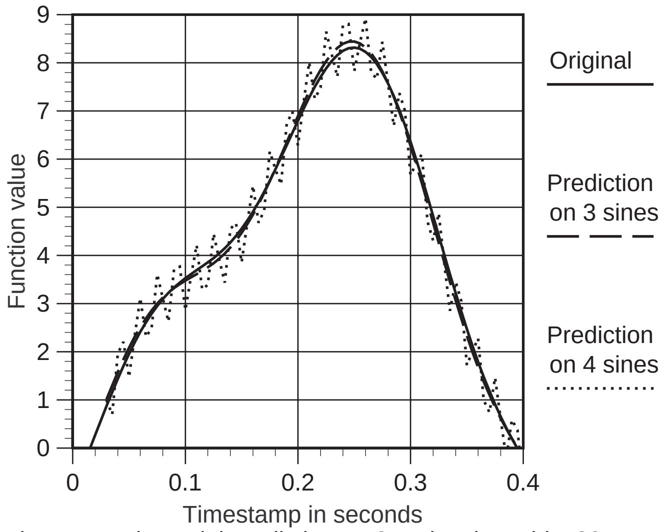


Figure 7: Polynomial prediction on 3 and 4 sinusoids, 30 ms prediction interval

The magnification of high-frequency components shown in Figure 6 appears as "jitter" to the user. Jitter makes the user's head location appear to "tremble" at a rapid rate. Because the magnification factor becomes large at high frequencies, even tiny amounts of noise at high frequencies can be a major problem.

We show this with a specific example of polynomial prediction on four sinusoids at a 30 ms prediction interval. Table 2 lists the four sinusoids. Figure 7 graphs these sinusoids and the predicted signals. The predicted signals are computed both by simulating the predictor in the time domain and by using the frequency-domain transfer functions to change the four sinusoids' magnitudes and phases: both approaches yield the same result. The "Original" curve is the sum of the sinusoids. The "Prediction on 3 sines" shows what the predictor generates when the input signal is the sum of the first three sines in the table. That predicted signal follows the original fairly closely. However, if we also include the 4th sinusoid, then the predicted signal becomes jittery, as shown by the "Prediction on 4 sines" curve. The last sine has a tiny magnitude, but it is a 60 Hz signal. One can think of it as a 60 Hz noise source. This example should make clear the need to avoid high-frequency input signals.

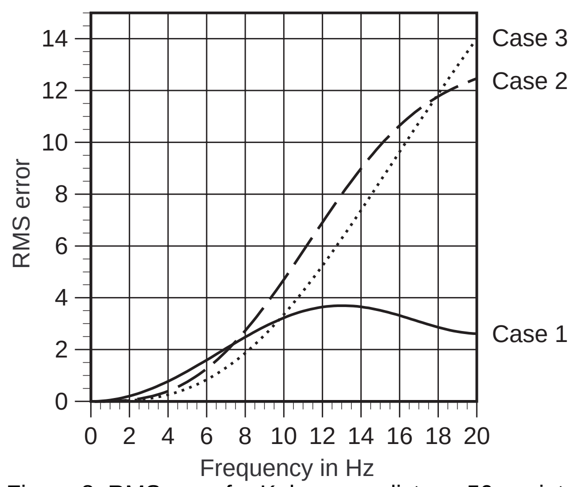


Figure 8: RMS error for Kalman predictors, 50 ms interval

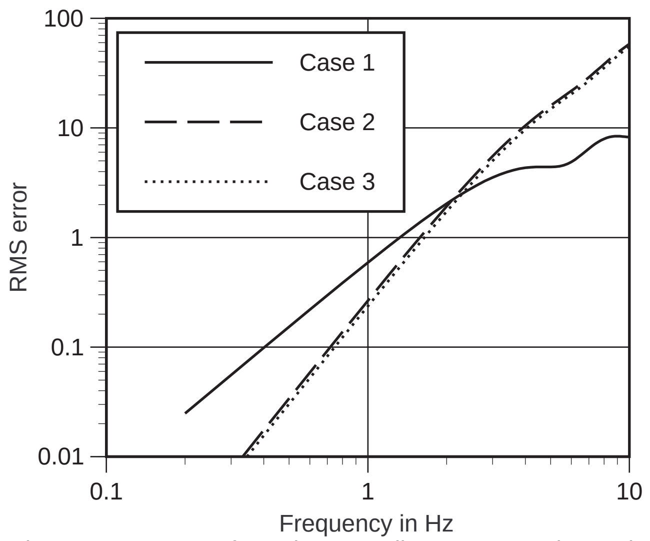


Figure 9: RMS error for Kalman predictors, 200 ms interval

2) Kalman-based predictors: We use the RMS error metric to compare the three Kalman cases and determine the frequency ranges where one is better than the others. These errors are computed using the transfer matrices described in Section 5.

Figure 8 graphs the RMS errors for the three cases at a 50 ms prediction interval. For frequencies under ~7 Hz, the inertial-based predictors Case 2 and Case 3 have lower errors than the non-inertial Case 1, and Case 3 is more accurate than Case 2 for frequencies under ~17 Hz. Figure 9 shows the errors for a 200 ms prediction interval. Both axes are plotted on a logarithmic scale. Now Case 2 and Case 3 have less error only at frequencies under ~2 Hz, instead of 7 Hz as in Figure 8.

These graphs provide a quantitative measurement of how much the inertial sensors help head-motion prediction. At high frequencies, Case 1 has less error than the other two because Case 1 does not make use of acceleration. Case 2 and Case 3 use acceleration to achieve smaller errors at low frequencies at the cost of larger errors at high frequencies. This tradeoff results in lower overall error for the inertial-based predictors because the vast majority of head-motion energy is under 2 Hz with today's HMD systems, as shown in Figure 1. The graphs also show that as the prediction interval increases, the range of frequencies where the prediction benefits from the use of inertial sensors decreases.

Case 3 is more accurate than Case 2 at low frequencies, because Case 3 has better estimates of acceleration. Case 2 directly measures velocity but must estimate acceleration through a numerical differentiation step. This results in estimated accelerations that are delayed in time or noisy. In contrast, Case 3 directly measures acceleration and estimates velocity given both measured position and acceleration, which is a much easier task. Case 3 is able to get nearly perfect estimates of velocity and acceleration. Since Case 2 represents using velocity sensors and Case 3 represents using accelerometers, this suggests that in theory, acceleration sensors are more valuable than velocity sensors for the prediction problem.

When the individual predictors are combined into a full 6-D predictor, the errors still increase dramatically with increasing

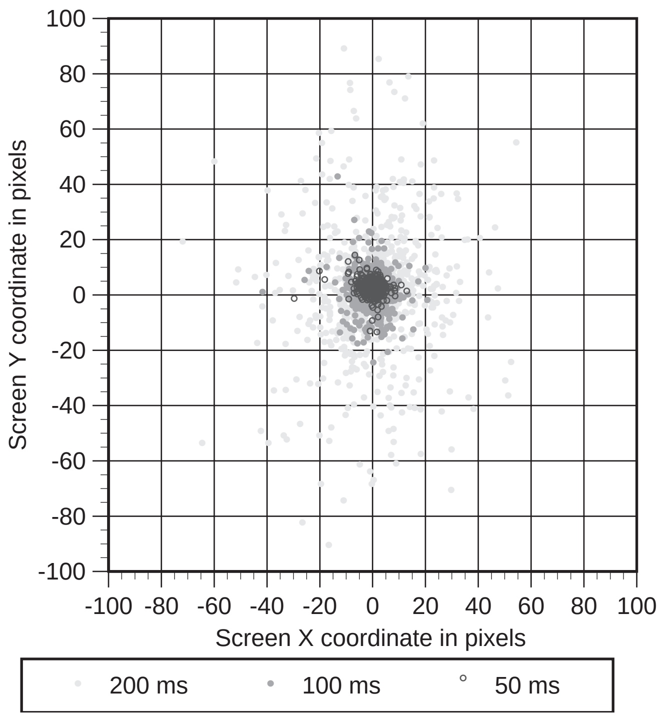


Figure 10: Scatterplots of screen-based error for simulated 6-D Kalman prediction, 30^o FOV HMD, 512x512 screen

prediction intervals. This is shown by the scatterplot diagram in Figure 10. A *scatterplot* is a way of representing screen-based error as seen inside an HMD. Imagine a point one meter in front of the user's right eye. This point is rigidly attached to the user's head, so that no matter how the user turns and moves his head, that point is always one meter in front of his right eye. Ideally, the projection of this imaginary point should always lie in the center of the field-of-view. However, system delays and prediction errors will cause the projection to stray from the center, resulting in the dots seen in Figure 10. The wider the spread of dots, the larger the error. The motion sequence used to generate these scatterplots came from recording the head motion of a first-time user in our HMD system. The predictor used Case 2 for the orientation terms and Case 3 for the translation. The average errors at the 100 ms prediction interval are 2.3 times as large as the errors at the 50 ms prediction interval, but the factor jumps to 9 when comparing 200 ms against 50 ms.

6.2 Spectrum of predicted signal

The prediction transfer functions can generate an estimate of the spectrum of the predicted signal, given the spectrum of the original signal. Multiply the input spectrum by the magnitude ratio of the polynomial predictor in Section 4, or by the magnitude ratio of $\mathbf{O}(z)$ for the Kalman-based predictors in Section 5. An example would be multiplying Figure 1 by one of the curves in Figure 2. Since different applications generate different input motion spectra, and a particular spectrum can represent an entire class of inputs, this technique can specify how a particular predictor will perform with a different application.

Figure 11 shows a specific example for the Case 1 Kalman predictor, with a 100 ms prediction interval. The spectrum of the original signal was derived from actual motion data of a first-time user in our HMD system. The spectrum of the predicted signal was estimated in two ways. First, we ran the Case 1 predictor in simulation, reading the time-domain signal, generating the predicted time-domain signal, then performing spectral analysis on the predicted signal. This is the "Predicted signal spectrum" graph in Figure 11. Second, we used the frequency-domain technique described in the previous paragraph. This is the "Frequency-estimated predicted signal spectrum" graph. The two estimates are virtually identical.

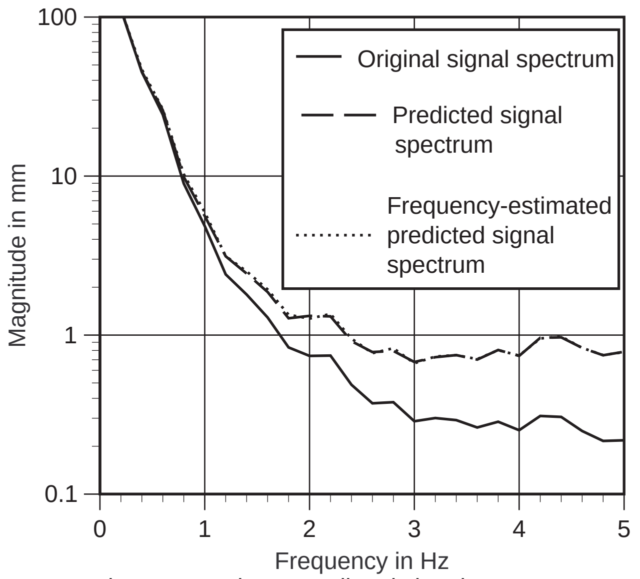


Figure 11: Kalman-predicted signal spectra

The spectrum of the predicted signal demonstrates what jitter looks like in the frequency domain. At low frequencies, the predicted and original spectra coincide, but as the frequency increases, the predicted spectrum becomes larger than the original. These "humps" in the predicted spectrum represent jitter.

6.3 Maximum time-domain error

Deriving a theoretical expression for the maximum error in the time domain would be useful. Determining this requires estimating the spectrum of the error signal, which is briefly sketched here for the Kalman predictors; see [1] for details. Let $e(t)$ be the error signal, which is the difference between the predicted position $h(t)$ and the original position $g(t)$:

$$e(t) = h(t) - g(t+p)$$

where p is the prediction interval. The goal is to derive the 1 by 1 error transfer matrix $\mathbf{U}(z)$, where $E(z) = \mathbf{U}(z)\, G(z)$. Define:

$$g_p(t) = g(t+p)$$

It turns out that for sinusoidal inputs:

$$G_p(z) = e^{j\omega p} G(z)$$

Now convert things into the Z-domain:

$$E(z) = H(z) - G_p(z)$$

Substitute for $H(z)$ using the expression from Section 5.2:

$$E(z) = \mathbf{O}(z) G(z) - e^{j\omega p} G(z)$$

$$\boxed{\mathbf{U}(z) = \mathbf{O}(z) - \left[e^{j\omega p}\right], \text{ where } E(z) = \mathbf{U}(z) G(z)}$$

Multiplying the spectrum of the original signal by the magnitude ratio of the error transfer matrix $\mathbf{U}(z)$ yields an estimate of the spectrum of the error signal. We can compute an upper bound for the largest time-domain value of this signal. The spectrum estimates the average magnitude of the coefficient for each component sinusoid. The absolute value of the magnitude is the maximum value that each component sinusoid will ever reach. Therefore, summing the absolute values of all the magnitudes provides a maximum upper bound. That also turns out to be an

Name	Estimated maximum	Actual highest	Actual average
Tx	137.2 mm	100.1 mm	3.6 mm
Ty	153.7 mm	155.6 mm	3.4 mm
Tz	69.2 mm	54.2 mm	1.8 mm
Yaw	6.9 deg	2.6 deg	0.3 deg
Pitch	8.9 deg	5.2 deg	0.4 deg
Roll	13.1 deg	11.7 deg	0.4 deg

Table 3: Estimated vs. actual time-domain errors in a recorded head-motion sequence, 100 ms prediction interval

achievable upper bound because we cannot put enough restrictions on the phase to prevent that from being a possibility.

By using this procedure, a system designer could specify the maximum tolerable time-domain error, then determine the maximum acceptable system delay that keeps errors below the specification. Unfortunately, the estimate is not a guaranteed upper bound because of uncertainties in the power spectrum (as mentioned at the end of Section 3) and because the measured spectrum is an *average* of the entire signal, which may not represent what happens at a *particular* subsection of the signal. Therefore, how closely do the estimated maximum bounds match the actual peak errors, in practice?

Table 3 lists the estimated maximums against the actual for all six degrees of freedom in one recorded HMD motion sequence. The maximums are usually within a factor of two of each other, although for the *Ty* sequence the estimated peak is lower than the actual peak. Overall, the estimated maxima are reasonable ballpark approximations that may be useful to a system designer.

7 Conclusions and limitations

This paper provides methods for comparing a class of head-motion predictors against each other, through analysis rather than a purely empirical basis. The results presented here quantify the need for short prediction intervals, demonstrate that accelerometers may be the most valuable inertial sensors to use, and provide a system designer with analysis tools.

The approach presented here is limited to *linear* predictors. While many existing head-motion predictors are linear or linearized versions of nonlinear formulations, in the future more sophisticated predictors will be nonlinear. They will be adaptive and will account for correlations in the motion signals. Analyzing nonlinear predictors is more difficult and is an area for future work.

Future HMDs will be lighter, allowing faster head motion. That will not invalidate this analysis, which is independent of the input motion spectra. However, motion spectra of rapidly changing head motion will have more energy at higher frequencies, making the prediction problem much harder. Future systems must have better predictors or shorter system delays.

Section 6.3 estimates the peak time-domain error, but a more useful measurement may be the *average* time-domain error. Note that the peaks in Table 3 are much larger than the average errors. An expression to estimate the average error could be useful.

Acknowledgements

We thank Vern Chi, Fred Brooks and Henry Fuchs for encouraging us to explore this topic. Funding was provided by ARPA contract DABT63-93-C-C048, the NSF/ARPA Science and Technology Center for Computer Graphics and Visualization (NSF prime contract 8920219), and Hughes Electro-Optical Systems. We thank the anonymous reviewers for their constructive comments and criticisms. Approved by ARPA for public release -- distribution unlimited.

References

[1] Azuma, Ronald. Predictive Tracking for Augmented Reality. Ph.D. dissertation. UNC Chapel Hill Department of Computer Science technical report TR95-007 (February 1995).

[2] Azuma, Ronald, and Gary Bishop. Improving Static and Dynamic Registration in an Optical See-Through HMD. *Proceedings of SIGGRAPH '94* (Orlando, FL, 24-29 August 1994), 197-204.

[3] Brown, Robert Grover, and Patrick Y.C. Hwang. Introduction to Random Signal and Applied Kalman Filtering, 2nd edition. John Wiley & Sons. (1992).

[4] Deering, Michael. High Resolution Virtual Reality. *Proceedings of SIGGRAPH '92* (Chicago, IL, 26-31 July 1992), 195-202.

[5] Emura, Satoru and Susumu Tachi. Compensation of Time Lag Between Actual and Virtual Spaces by Multi-Sensor Integration. *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* (Las Vegas, NV, 2-5 October 1994), 463-469.

[6] Harris, Frederic J. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. *Proceedings of the IEEE 66*, 1 (January 1978), 51-83.

[7] Jenkins, Gwilym M. and Donald G. Watts. Spectral Analysis and its Applications. Holden-Day. (1968).

[8] Lawton, W., T. Poston and L. Serra. Calibration and Coordination in a Medical Virtual Workbench. *Proceedings of Virtual Reality Applications* (Leeds, UK, 7-9 June 1994).

[9] Lewis, Frank L. Optimal Estimation. John Wiley & Sons, 1986.

[10] Liang, Jiandong, Chris Shaw, and Mark Green. On Temporal-Spatial Realism in the Virtual Reality Environment. *Proceedings of the 4th Annual ACM Symposium on User Interface Software & Technology* (Hilton Head, SC, 11-13 November 1991), 19-25.

[11] List, Uwe H. Nonlinear Prediction of Head Movements for Helmet-Mounted Displays. Technical report AFHRL-TP-83-45 [AD-A136590], Williams AFB, AZ: Operations Training Division (1984).

[12] Oppenheim, Alan V. and Alan S. Willsky. Signals and Systems. Prentice-Hall Inc. (1983).

[13] Paley, W. Bradford. Head-Tracking Stereo Display: Experiments and Applications. *SPIE Vol. 1669 Stereoscopic Displays and Applications III* (San Jose, CA, 12-13 February 1992), 84-89.

[14] Phillips, Charles L., and H. Troy Nagle. Digital Control System Analysis and Design, 2nd edition. Prentice-Hall, Inc. (1990).

[15] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. Numerical Recipes in C. Cambridge University Press (1988).

[16] Priestley, M.B. Spectral Analysis and Time Series, Vol. 1. Academic Press (1981).

[17] Rebo, Robert. A Helmet-Mounted Virtual Environment Display System. MS Thesis, Air Force Institute of Technology (December 1988).

[18] Riner, Bruce and Blair Browder. Design Guidelines for a Carrier-Based Training System. *Proceedings of IMAGE VI* (Scottsdale, AZ, 14-17 July 1992), 65-73.

[19] So, Richard H. Y. and Michael J. Griffin. Compensating Lags in Head-Coupled Displays Using Head Position Prediction and Image Deflection. *Journal of Aircraft 29*, 6 (November-December 1992), 1064-1068.

[20] Zikan, Karel, W. Dan Curtis, Henry A. Sowizral, and Adam L. Janin. A Note on Dynamics of Human Head Motions and on Predictive Filtering of Head-Set Orientations. *SPIE Proceedings volume 2351: Telemanipulator and Telepresence Technologies* (Boston, MA, 31 October - 4 November 1994).