Correcting for Dynamic Error

SIGGRAPH '97 Course Notes #30: Making Direct Manipulation Work in Virtual Reality August 1997

Ronald T. Azuma
Hughes Research Laboratories
3011 Malibu Canyon Road, MS RL96
Malibu, CA 90265
azuma@isl.hrl.hac.com
http://www.cs.unc.edu/~azuma
W: (310) 317-5151
Fax: (310) 317-5695

Abstract

Dynamic errors are the largest single source of registration errors, and system delays cause most of the dynamic errors. Prediction is a key technique in combatting the effects of system delay. This paper discusses several prediction methods and how well they may be applied to the head-motion prediction problem. Since Kalman filters are the basis of the most accurate previous work, a simple example of a Kalman filter is presented and explained. For certain types of predictors, examination in the frequency-domain provides a useful analysis tool for evaluating predictor performance. More accurate predictors and evaluation techniques need to be developed. This paper concludes by sketching some directions that should be explored.

1. Motivation

Registration errors can be divided into two types: static and dynamic. *Static* errors are the ones that cause registration errors even when the user's viewpoint and the objects in the environment remain completely still. *Dynamic* errors are the ones that have no effect until either the viewpoint or the objects begin moving. Correcting for dynamic errors is important because they are usually the largest contributors to registration error. This has been demonstrated both empirically and through an analysis of the sources and magnitudes of registration errors [Holloway95].

Dynamic errors are primarily caused by system delays. For example, a system with a typical 100 ms latency and a user executing a moderate head rotation rate of 50 degrees per second will have 5 degrees of registration error, equivalent to the length of ten full moons laid end-to-end. While dynamic tracker errors exist, few efforts have been

made to compensate for those and such approaches are usually specific to the particular tracking technology. In comparison, system delays are ubiquitous and have been the target of many research efforts.

Prediction is an important technique to compensate for system delays. The discussion of "Registration," elsewhere in this set of course notes, explained that methods of reducing dynamic registration error include reducing the real or apparent system lag, matching the temporal streams (when using video-based systems), and predicting future head locations [Azuma97]. Prediction is an important part of the first two approaches because it is not currently possible to eliminate lag as a significant source of registration error. The third approach, matching temporal streams, can produce impressive results but is not an ideal solution for direct manipulation interfaces. In video-based systems, a head-mounted video camera provides the user's view of the surrounding real environment. The video camera and digitization hardware impose inherent delays on the user's view of the real world. This is potentially a blessing when reducing dynamic errors, because it allows the temporal streams of the real and virtual images to be matched. Additional delay is added to the video from the real world to match the scene generator delay in rendering the virtual images. Detecting known landmarks in the real scene can be used to enforce almost perfect registration [State96]. However, such a system delays the user's view of *both* the real and the virtual. While the registration may visually appear internally consistent, the visual display is delayed with respect to the user's physical actions, thus maintaining registration errors in direct manipulation interfaces. This is similar to the problems encountered by telepresence systems with significant delays and is a very difficult problem to overcome. Therefore, the fourth approach, prediction, may be the best hope for direct manipulation interfaces. The rest of these notes focuses on the use of prediction to reduce dynamic registration errors.

2. Approaches

Prediction is like driving a car when the only available view is the rear view mirror. The driver can see what has happened in the past, but he has no direct view of the future. This is called a *causal* situation, where all past values, but none of the future, are known. To keep the car on the road, the driver must anticipate, or predict, where the road will go, based solely on the past observations and his knowledge of roads in general. How difficult this task is depends on the shape of the road and how fast the car is going. If the road is straight and remains so, then the task is easy. If the road twists and turns unpredictably, the task may be nigh impossible. Or it may lie somewhere in between. Thus, the first question to ask about head-motion prediction is what type of "road" it is -- trivial, intractable, or tractable?

Predicting head motions for HMD systems is an interesting and potentially tractable problem because of the motion characteristics and the required prediction intervals. Some motions are very well characterized and easy to accurately predict. For example, the dates and times of solar eclipses can be predicted centuries in advance. Other motions are so random that they are basically unpredictable except in a broad statistical sense, like predicting the location of one particle undergoing Brownian motion. Head motion lies somewhere in between these two extremes. It exhibits strong temporal and spatial coherence, but the motion is not so simple that one can easily find a model

that completely characterizes it. For the range of user head motions and system delays in existing systems, it is possible for prediction to reduce registration errors.

Predicting head motion is a specific example of a much larger class of prediction and estimation problems that have received a great deal of attention. The characteristics of this problem determine the most appropriate approach to take. What are the characteristics of head motion?

First, head motion is a collection of multidimensional signals that are difficult to perfectly model. Translation may be represented by a linear model. If orientation is represented by quaternions, then the orientation model is nonlinear because the four quaternion terms are nonlinearly dependent on each other. However, orientation is not intrinsically nonlinear, at least in a local neighborhood around the current orientation. It may be possible take a small angle approximation around the current orientation and use a linear model for prediction that way. However, determining a model that fits head motion closely is nontrivial, linear or not.

Second, head motion is *nonstationary*, which means that the statistical properties of the curves may change with time. For example, a user may keep her head still for a long period of time, then suddenly start moving around at rapid velocities.

Finally, the measurements of head motion will be corrupted by noise, generally of a complicated nature. A *bandlimited white noise process* is one that has equal magnitudes for all frequencies below a certain limit and zero energy for all frequencies above the limit. White noise is well behaved, but it not generally the case that the tracker and inertial sensor measurement inaccuracies are accurately modeled solely by adding white noise; the errors are usually more complicated than that. The existence of noise requires estimators to extract the best guess of the true value of the signal. The combination of difficult modeling, nonstationary signals and noise makes it difficult to apply standard prediction and estimation techniques without simplifying the problem or violating assumptions.

For existing virtual environment systems, the energy of user head motion is concentrated below 2 Hz when displayed in the frequency domain, and system delays typically fall between 50 - 250 ms. Section 4 will show that the system delay and the spectrum of head-motion energy have a great deal of impact upon the performance of predictors.

Many different prediction techniques exist. The ones that seem the closest match to this particular problem are:

- Curve fitting
- Information theory
- Control theory
- Time-series analysis
- Wiener and Kalman filters

Curve fitting: Fitting curves or splines to data is a reasonable method for smoothing but generally gives unsatisfactory results when used for prediction. Because the data points are corrupted with noise, techniques that approximate the points should be

used, instead of ones that go through the points. Since high order curves can introduce "wiggles" that are not representative of the original motion, locally-applied low-order curves like quadratic or cubic curves are usually the technique of choice. In general, extrapolating future data points with such curves yields unsatisfactory results.

Control theory: Several previous works use control theory to aid predictor design; unfortunately that does not apply to this particular problem. Control theory applies to problems where a system attempts to follow a known input signal as accurately as possible, within the physical limitations of the system. For example, imagine a robot arm. The input signal tells the arm to move to a new position. However, since the arm has inertia and the motor strength cannot generate an infinite amount of force, the arm cannot instantaneously move to the new position. It requires some time to execute the move. The inertia in the system is modeled by a transfer function. Control theory is the art of designing controllers that replace the input signals to the motor with a new set of signals that make the arm move in a way that is optimal with respect to some criteria, such as minimizing the time required or the energy consumed. Control theory is useful in flight simulators because airplanes have inertial characteristics and input signals that are easily measured. A plane cannot instantaneously change direction or speed, and these limitations are modeled by a transfer function. The input signals come from the pilot's controls in the cockpit. Figure 1 compares the flight simulator application with the problem of head-motion prediction in a virtual environment system. There is no transfer function within the virtual environment system itself, because there is no "inertia" between the input and output head locations. The only thing the system adds is delay. The inertia exists in the human head. Using control theory in this problem would require somehow measuring the neural signals that control the muscles that move the head.

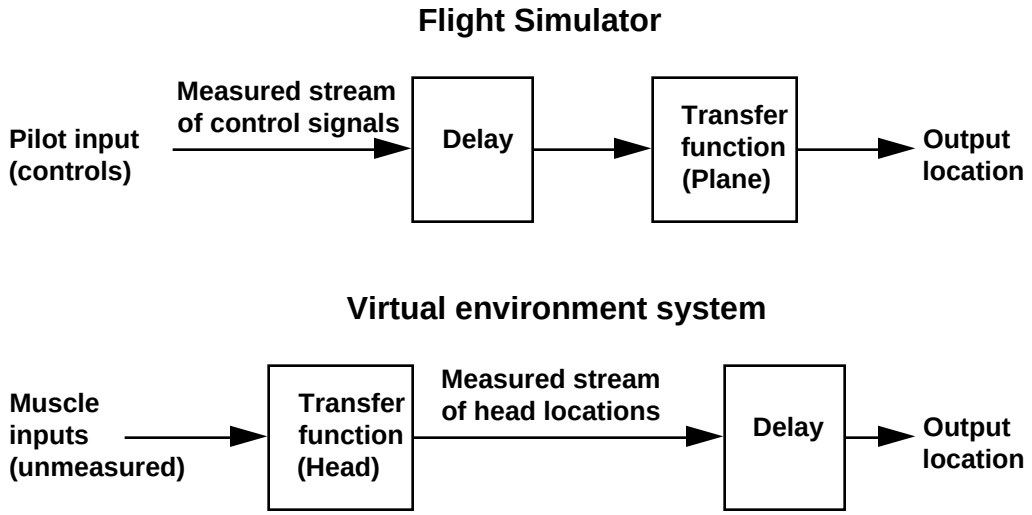

Figure 1: Flight simulator vs. virtual environment system

Information theory provides some interesting results on the prediction problem under specific conditions. Assume a 1-D signal is bandlimited, stationary, and free of noise. *Bandlimited* means that the signal has no energy in the frequency domain above a specified frequency. *Stationary* means that the statistical characteristics of the signal, such as mean and covariances, remain constant with time. With these three assumptions, it is theoretically possible to predict *arbitrarily far* into the future with complete accuracy. This requires knowledge of all past values of the signal and that the signal be

sampled faster than the Nyquist rate. This result makes use of the bandwidth restriction, not of any statistical properties of the signal. Conceptually, this result is true because a stationary bandlimited signal will eventually start repeating itself if observed for sufficiently long. By looking into the past far enough, a predictor can extract enough information to essentially reconstruct the entire function, allowing prediction of arbitrary distances into the future. If the number of past values is limited, formulas exist that minimize the prediction error based on that number of values. For details, see [Splettstösser82].

Unfortunately, these theoretical results are not useful in practice because measured head-motion signals are noisy and nonstationary. With nonstationary signals, looking deeply into the past may not be useful for predicting future values. The existence of noise makes the prediction formulas impractical. The formulas are extremely sensitive to any noise in the data. Even the limited precision of floating point numbers causes problems in practice, restricting effective prediction distances to fractions of the sampling interval, which is not a useful range for the head-motion prediction problem.

Time-series approaches work on noisy data by approximately fitting specific models to the data. Almost all the literature in this area assumes stationary, linear signals. Given that values in the incoming signals are somehow correlated from current time t to a future time $t + dt$, one wants to find a model that fits the data so well that the difference between the model and the data is white noise. In general, this is impossible, so the task becomes one of finding or picking a model that matches reasonably well. Standard models include linear ramps, impulse functions, constant values, and other curves. Time series analysis is often used in economic applications, such as forecasting the demand for heating oil four months from now, so models often include cyclical components like sinusoidal functions. Regression methods can be used to find parameters to best fit a specific model to provided data. Unfortunately, such simple models do not accurately fit head motion data for more than short time intervals. More complicated models include autoregressive, moving average (MA), autoregressive moving-average (ARMA) and Box-Jenkins approaches [Montgomery90].

Basic time-series approaches do not use any information about the derivatives of the signals, since in most time-series applications the only information available is the signal itself. Clearly, having sensors that directly measure velocity or acceleration information of the user's head should make the prediction problem easier. While it is possible to estimate derivative information from position signals, numerical differentiation is a noisy operation and the derivative estimates will be delayed in time from their true values. That is, a derivative estimator requires some history of position values in order to make an accurate estimate, which introduces lag. In contrast, sensors that directly measure velocity information can report that as it happens, instead of after it happens.

Two general classes of optimal linear estimators exist: *Wiener and Kalman filters.* They generally supersede the ARMA and Box-Jenkins models used in time-series forecasting, at the cost of additional complexity. Both Wiener and Kalman filters are optimal in the sense that they minimize the expected mean-square error, given certain assumptions. Wiener filtering assumes that the signal to be detected is described by a white noise process, and this signal is corrupted by a different white noise source. Both the noise and the signal processes must be characterizable by knowing their

autocovariance and cross-covariance functions. Kalman filtering has a different set of assumptions. It assumes that the signals can be modeled by a set of variables that capture the state of the system at any time, along with a process that determines how these state variables change in time in the absence of any input. The output of this model, when subtracted from the signal, results in white noise, and the covariances of that noise are known. The signals are assumed to be corrupted with white noise of known covariances. The initial values of the state variables and their covariances are known. The signals can be nonstationary. Then the Kalman filter will take the measurements and return the optimal estimate for the state variables at any desired instant of time.

Kalman filtering is a better choice than Wiener filtering for the head-motion prediction problem. Wiener filtering assumes a noiselike signal, but head-motion signals have far too much structure to be accurately modeled as a noiselike signal. Wiener filtering is also more computationally intensive than Kalman filtering is, especially when extracting multiple outputs simultaneously out of a set of signals. Kalman filtering has an efficient recursive formulation that is suitable for computer implementation, an important factor when the predictor has to run in real time. Finally, the Kalman filter's use of state variables allows the formulation of a simple motion model that uses the derivative information provided by any head-mounted inertial sensors.

For virtual environment systems, researchers have used time-series analysis [Albrecht89] [Deering92] [Paley92] [Wu95], Kalman filters [Azuma94] [Freidmann92] [Liang91] [Rebo88] [Zikan94], and other ad hoc approaches [Smith84] [Welch86]. Since Kalman filters form the basis of the most accurate predictors demonstrated so far, the next section provides an example of a simple Kalman filter predictor.

3. Example

This section provides an introduction to the Kalman filter -- what it does, how to operate it, and what one example looks like. This section does not derive the Kalman filter, step through the example, or give full details of the filter. For that, please see [Brown92] [Lewis86]. A good historical background is in [Sorenson70]. The example I use is based on [Lewis86] and [Maybeck79].

The example filter operates on the following problem: a car travels down a long, straight road. This makes it a 1-D problem, for simplicity. A remote observer (such as an aircraft) occasionally measures the position of the car. The car's velocity and acceleration are not directly measured. The goal is to estimate the car's present and future positions and velocities. This problem is a 1-D version of the virtual environment tracker problem, since most head trackers provide occasional position measurements but no velocity or acceleration readings.

The Kalman filter maintains the status of the car in two matrices: **X** and **P**. **X** is a vector that holds the state variables. In this case, **X** is a 2 by 1 vector, with position *d* and velocity *v*:

$$\mathbf{X} = \begin{bmatrix} d \\ v \end{bmatrix}$$

The values in this state vector are only approximations of the unknown true values. This is due to limited sensor accuracies and the fact that measurements are not always available. The inaccuracies in the measurements are assumed to be accurately described by adding white noise. The Kalman filter models this uncertainty by a Gaussian probability distribution, as shown in Figure 2. This probability distribution comes from the assumption that all noise processes are white.

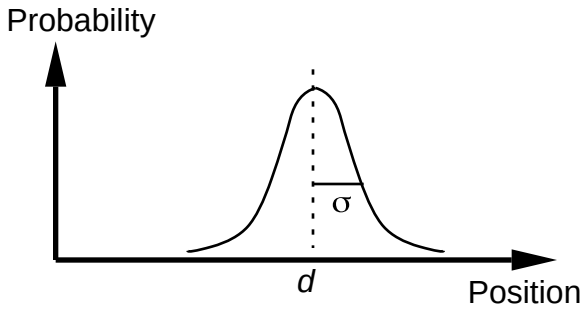

Figure 2: Gaussian probability distribution for position *d*

The Gaussian represents the distribution of values where the true position might be. The most likely position is *d*, the position in the state vector. The width of the Gaussian is specified by standard deviation σ. Large standard deviations yield "fat" Gaussians, showing that the estimate of *d* is not very accurate, while small standard deviations represent more accurate estimates.

The covariance matrix **P** stores these uncertainties. **P** is an *N* by *N* matrix, where *N* is the number of entries in state vector **X**. In this example, **P** is 2 by 2. Let σ_d be the standard deviation of variable *d* and σ_v be the standard deviation of variable *v*. Then:

$$\mathbf{P} = \begin{bmatrix} \sigma_d^2 & \sigma_d\sigma_v \\ \sigma_d\sigma_v & \sigma_v^2 \end{bmatrix}$$

The Kalman filter requires two models: a system model and a measurement model. The system model describes how the state vector changes with time in the absence of any new measurements. It is described by:

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{G}w$$

where *w* is a white noise process with covariance **Q** (a 1 by 1 matrix). For this example:

$$\begin{bmatrix} \dot{d} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w$$

This system model is very simple. The derivative of position is the velocity, while the derivative of velocity is declared to be zero, except for the addition of some white noise.

The second model is the measurement model, which describes how the state vector **X** is related to measurements **Z**:

$$\mathbf{Z} = \mathbf{H}\mathbf{X} + e$$

In this example, **Z** is a 1 by 1 matrix holding the remotely measured position *y*, and *e* is a white noise process with covariance **R** (a 1 by 1 matrix):

$$[y] = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} d \\ v \end{bmatrix} + e$$

Figure 3 provides a flowchart of how the Kalman filter operates. The filter starts by initializing **X** and **P** and sets current time *t*. Every time a new position is measured, the filter updates **X** and **P** to reflect this new information. It does this in two steps that are similar in form to predictor-corrector methods used in numerical integrators. Say the new measurement is taken at time t_1. The filter runs a *time update* (or predictor) step that uses the system model to generate new estimates of **X** and **P** at time t_1. Then the filter runs a *measurement update* (or corrector) step that uses the measurement model to blend in the remotely measured position into the new estimate. The result is a final estimate of **X** and **P**. The current time is then set to t_1, and the filter repeats the predictor-corrector steps with each new measurement. Whenever a predicted output is required, the Kalman filter runs a time update step from the current filter time to the desired predicted output time.

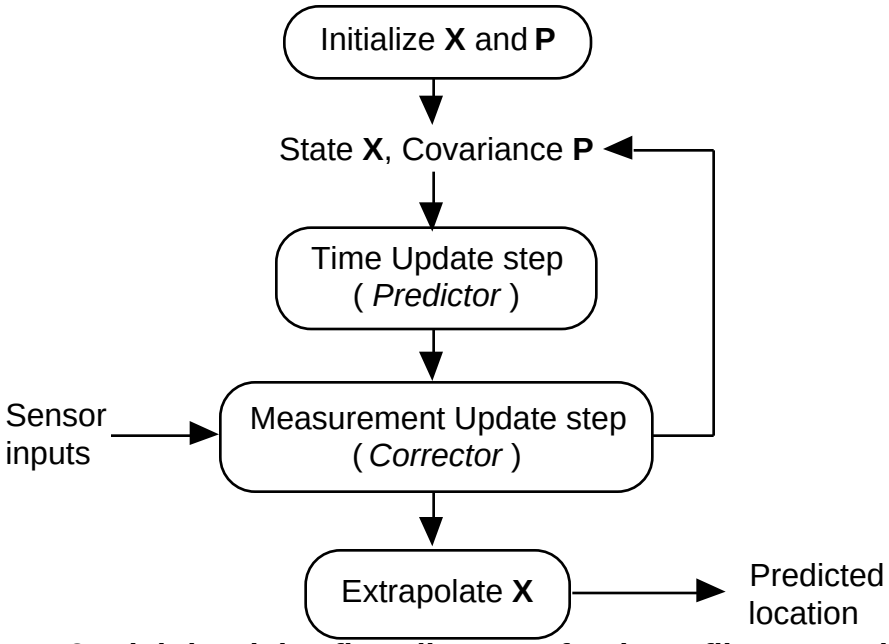

Figure 3: High-level dataflow diagram of Kalman filter operation

The time update step extrapolates **X** and **P** from time *t* to time t_1 in the absence of any measurements. The time update step integrates the following derivatives:

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X}$$

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^{\mathbf{T}} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathbf{T}}$$

The solution to this is usually performed by an Ordinary Differential Equation solver, such as a 4th order Runge-Kutta method. Sometimes, though, a closed-form solution can be found. In this example, the closed-form solution is very simple:

$$d_{t1} = d_t + v\left(t_1 - t\right)$$

This is the familiar equation for distance traveled by an object of constant velocity.

The measurement update step computes a new **X** and **P** at time t_1 by blending in the measured data. The equations are:

$$\mathbf{K} = \mathbf{P}\mathbf{H}^{\mathbf{T}}\left[\mathbf{H}\mathbf{P}\mathbf{H}^{\mathbf{T}} + \mathbf{R}\right]^{-1}$$

$$\mathbf{P}_{t1} = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{P}$$

$$\mathbf{X}_{t1} = \mathbf{X} + \mathbf{K}\left(\mathbf{Z} - \mathbf{H}\mathbf{X}\right)$$

where **K** is the Kalman gain matrix and **I** is the identity matrix. For this example, **K** is a 2 by 1 matrix and **I** is the 2 by 2 identity matrix.

How does this combination occur? Rather than give a numerical example, I will graphically show what happens to the position and standard deviation as a result of the time and measurement update steps (Figure 4). The initial position d at current time t, with associated standard deviation σ_d, is on the left side of the diagram. A new measurement y is taken at time t_1. That measurement has an associated standard deviation σ_y. The car moves from left to right in this diagram. The time update step takes the initial position d and predicts a new position d_t at time t_1. The new standard deviation σ_{dt} is larger than the initial standard deviation, because the covariance always increases when running the time update step (the noise makes the uncertainty grow with time in the absence of any new measurements). Then the measurement update step combines the time update estimate with the measurement, to form the final estimate d_f with standard deviation σ_{df}. Note that the final estimate is a blend of the time-update estimate and the measurement, and that its associated standard deviation σ_{df} is smaller than either σ_{dt} or σ_y. By combining the time-update estimate and the measurement, the Kalman filter provides a more accurate overall estimate.

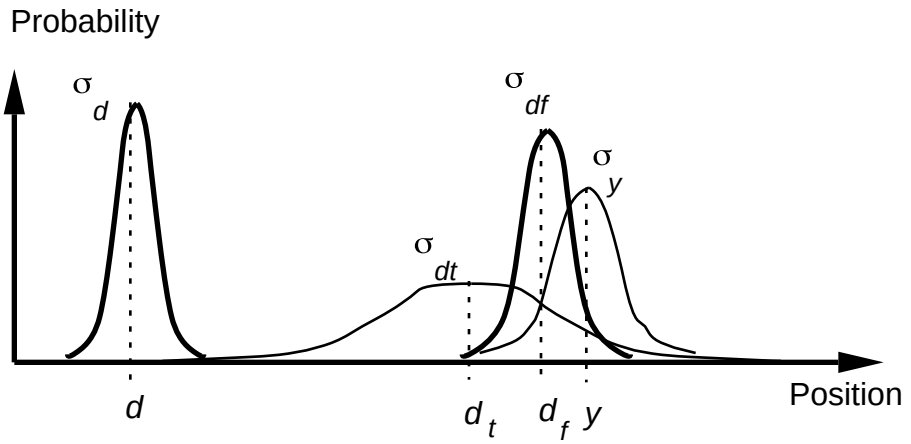

Figure 4: An example of how position and standard deviation change during the time and measurement update steps

Note that the state vector **X** includes an estimate of velocity even though velocity is not directly measured. One way to think about this predictor is that the Kalman filter continually tries to provide the best estimate of position *d* and velocity *v* to use with the simple predictor:

$$d_{t1} = d_t + v(t_1 - t)$$

This example shows that implementing a Kalman filter is fairly easy. The filter, as shown in Figure 3, looks like a computer algorithm. Most of the steps are multiplying and adding various matrices, which are operations familiar to all computer graphics programmers. So what is hard about this? The difficult part is in modeling the problem: coming up with the variables to use in the state vector **X**, determining the relationships in matrix **A**, setting the noise values **Q** and **R**, etc. Ideally the model should be so perfect that the residual (**Z - H X**) behaves as white noise; in practice this doesn't happen for head motion, so the goal is to get as close as possible. The filter must also be implemented in a real-time system where accurate timestamps are taken with each tracker measurement and the system delays are accurately known. Building such a system is not trivial.

Kalman filters come in two main flavors: discrete and continuous. *Discrete* filters are used when all measurements and computations occur across evenly-spaced time intervals. *Continuous* filters support time intervals that are not evenly spaced. This example is a hybrid, blending a continuous time update with a discrete measurement update.

The Kalman filter is linear, but a variant can be used to handle nonlinear problems. Optimal nonlinear estimation in general is either intractable or too computationally expensive to be practical. Any nonlinear estimator that runs in real time must therefore be suboptimal. [Chang84] surveys such approaches, including the finite memory filter, the fading memory filter, and the constant gain filter, but it recommends the Extended Kalman Filter (EKF) for most purposes. A variation of the standard Kalman filter, the EKF linearizes the model about the operating point specified by the current set of state variables, using that approximation to make the problem tractable.

4. Evaluation

The example described in the previous section was very simple. One can build far more sophisticated models. This raises the question: how much improvement can one expect from a predictor, given the ability to choose any possible prediction method? What is the most improvement any predictor can hope to accomplish? Unfortunately, expressing a bound when one can choose any arbitrary predictor is an intractable problem.

The problem comes from the combination of the following two properties: 1) Head-motion signals are nonstationary, and 2) No algorithm exists that can select the optimal model for a given situation. The combination is important, because for bandlimited stationary signals, perfect prediction is theoretically possible for arbitrary intervals into the future, as mentioned in Section 2. The problem of finding models and their associated parameters to match collected data is sometimes called the *system identification* problem, and no systematic technique exists for finding the optimal model for arbitrary data. What is an optimal model? Ideally, a model would be able to predict future values so well that the error between the predicted signal and the true signal has the characteristics of white noise. If this occurs, then no model can do better because no systematic information is left to be extracted. In essence, any potentially superior model must predict the remaining difference, which is white noise, and that cannot be done. Note that the definition of the Kalman filter assumes that you have such a model, but it does not tell you how to generate that model in the first place! In general, such an optimal model is not feasible. The question then becomes, how close can one come to this optimal model? No algorithm exists that can determine this model. Regression techniques specify how to find parameters that best fit a *specific model*, such as a line, a curve, or a sinusoid, to observed data. They do not specify which model to pick, nor can they find the best model out of the arbitrary, infinite space of potential models. A theoretical limit of how well the best predictor could possibly do on head-motion signals cannot be specified. In practice, system identification usually requires experimentation to find an acceptable model [Fleming89].

If an overall theoretical bound is not available, then how well do predictors work empirically? [Azuma94] demonstrated a system that combined a simple motion model with inputs from head-worn inertial sensors. For system delays under 80 ms, prediction reduced average registration errors by a factor of 2-3 without the use of inertial sensors and a factor of 5-10 with the use of inertial sensors. Therefore, prediction can be effective in removing significant amounts of error, and even better performance is probably possible with a more sophisticated motion model.

It is also possible to analyze the characteristics of *specific* types of predictors. The rest of this section briefly summarizes the results of one such analysis; please see [Azuma95a] [Azuma95b] for details. For linear, separable and temporally discrete predictors, it is possible to use a frequency-domain approach to examine predictor performance. These assumptions allow the use of a basic result of linear systems theory: Any sinusoidal input into a linear system results in *another sinusoid of the same frequency* but with different magnitude and phase. This makes it possible to completely

characterize linear systems by describing how the magnitude and phase of incoming sinusoids change as a function of frequency. This characterization is called a *transfer function*. If the input is the sum of many different sinusoids (e.g., a Fourier-domain signal), then to compute the output, take each individual sinusoid, change its magnitude and phase as specified by the transfer function, and sum the resulting output sinusoids. This property, which is true of linear systems, is called *superposition*. Analyzing predictors in the frequency-domain allows one to examine and compare predictors without using specific motion datasets.

What are the ideal magnitude ratios and phase differences for a predictor? In the time domain, an ideal predicted signal is simply the original signal shifted forward in time by some prediction interval p. In the frequency domain, this corresponds to a magnitude ratio of 1 for all frequencies (since the magnitude of each predicted sinusoid should be exactly the same as the magnitude of the original sinusoid) and a phase difference of $p\,\omega$ (since each sinusoid is shifted forward in time).

Actual predictors do not match this ideal. For the simple predictor:

$$x_{predicted} = x + v\,p + 0.5\,a\,p^{2}$$

where x = position
v = velocity
a = acceleration
p = prediction interval

The magnitude ratio for this predictor is:

$$\sqrt{1 + \frac{1}{4}(\omega\,p)^{4}}$$

where ω = angular frequency of the sinusoid

The magnitude ratio is the ideal of 1 only when $p = 0$ (no prediction) or $\omega = 0$ (a constant signal). The ratio grows roughly as the square of either p or ω. Note the intimate relationship between p and ω. This suggests a relationship between allowable bandwidth and the prediction interval. Halving the prediction interval means that signal can double in frequency while maintaining the same prediction performance. That is, bandwidth times the prediction interval yields a constant performance level.

This magnitude ratio is the cause of the "jitter" seen in predicted signals. Jitter is annoying, low-magnitude high-frequency motions in the predicted signal that do not seem representative of the original signal. The predicted signal appears to shake rapidly, as if the user drank too much caffeine before wearing the HMD, because prediction magnifies the high-frequency components of the original signal. Figure 5 shows how the predicted signal spectrum has more energy than the original signal at high frequencies. Thus, the analysis can quantify the amount and distribution of jitter.

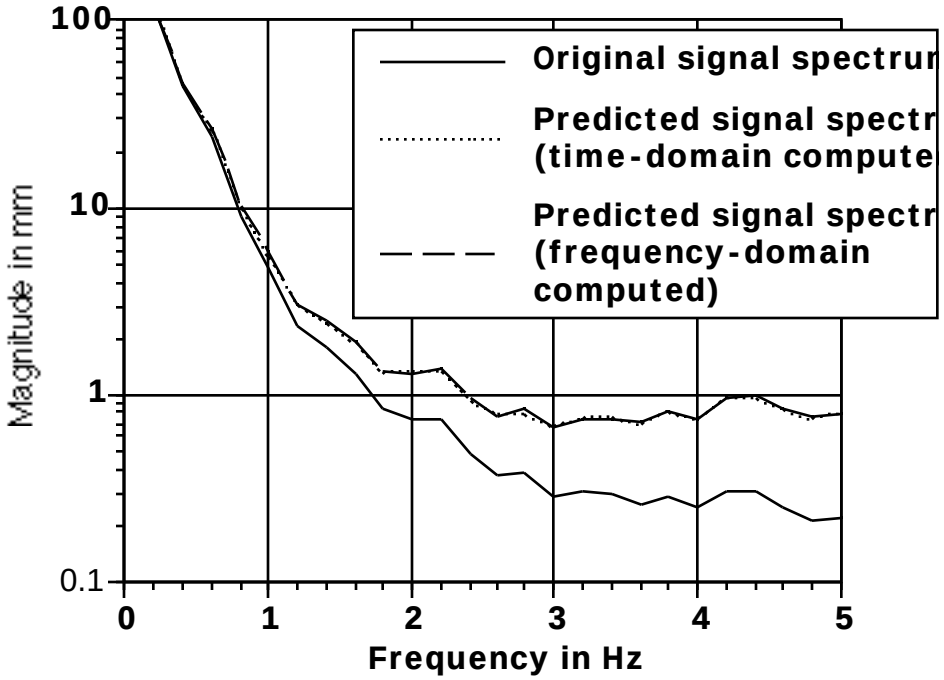

Figure 5: Spectra of original and predicted signals

A frequency-domain analysis can also compute the root-mean-square (RMS) error of a predictor, plotted against frequency. Figure 6 shows the errors for three types of predictors. The case 1 predictor is a Kalman predictor without inertial sensors, using only position and velocity in the state, much like the example in Section 3. The case 2 and 3 predictors are Kalman predictors that use inertial sensors to measure velocity and acceleration, respectively, and use position, velocity and acceleration in the state vector. The error for the inertial predictors is lower than the non-inertial predictor for low frequencies but is much larger for high frequencies. This results in lower overall error if the original signal is concentrated only at low frequencies, which is the case with head motion (< 2 Hz). However, if any high frequency components exist (e.g., a 60 Hz noise source from the power supply), that will be greatly magnified by the predictor and appear as jitter.

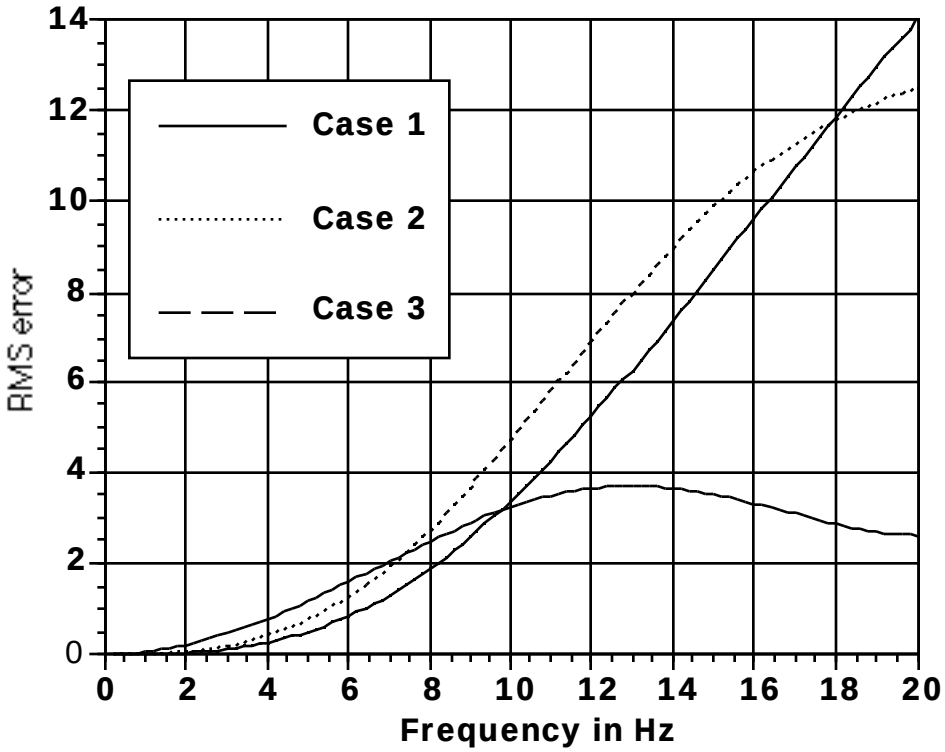

Figure 6: RMS errors for Case 1, 2, and 3 predictors at 50 ms prediction interval

While prediction can be effective, it is not a panacea. Because errors grow rapidly with both increasing prediction interval and motion frequency, one must be careful about the range of motions and system delays that prediction is applied to. Simply attaching a predictor to a virtual environment system with 250 ms of delay is not going to yield satisfactory results. To be most effective, prediction must be combined with efforts to control and reduce overall latency. For a class of linear predictors, some analysis tools are available to help system designers incorporate predictors, but more sophisticated analysis tools and predictors must be developed.

5. Future work

More work needs to be done in evaluating and comparing prediction methods, preferably in ways that are not dependent upon specific motion datasets. Analysis techniques for nonlinear predictors will be needed. While previous work has shown how to estimate the *maximum* time-domain error produced by a predictor, a more useful measurement would be an estimate of the *average* time-domain error, which is typically much smaller than the peak. Another useful service might be to provide a set of "typical" motion datasets that everyone can use as benchmarks for comparing predictors, much as certain volumetric and image datasets have been used to compare rendering and image processing methods.

More sophisticated prediction methods might further reduce dynamic registration errors. Adaptive methods that adjust to varying head motion deserve more exploration. Using a nonadaptive predictor is like trying to race a car at constant speed; slowing down

on the curves and speeding up on the straight-aways will improve performance. This adaptation may take the form of using non-white noise models, varying the model and measurement covariance matrices, or running several different models in parallel and selecting amongst them. Analyzing head motion for recognizable patterns, correlations amongst several signals, or high-level characteristics may aid prediction. If one can assume a specific task, then more accurate prediction may be possible because a more accurate model can be built. Other researchers have begun looking for such patterns [Shaw92]. For example, Fitts' Law has been shown to apply to head motion [Andres89] [Jagacinski85]. A book edited by Peterson describes the physiology of head movement and may be of use to anyone interested in developing physically-based models of head motion [Peterson88].

Since head motion is nonstationary, it may be possible to generate several models and switch between them adaptively. The framework for adaptive filters is straightforward [Magill65]. A common method is to run several Kalman filters in parallel, each with a different model, then choose one or blend the various outputs. These are sometimes called Multiple Model or Generalized Likelihood techniques. The difficult part of adaptive filters is not in the adaptive framework; it is in building the models in the first place.

Tuning models too tightly to specific motions can be dangerous. Prediction will be highly accurate if the actual motion matches expectations. But if the actual motion is different from what the model expects, errors can become very large. In target tracking applications this is sometimes referred to as "losing lock" on the target. Therefore, a tradeoff is going to exist between generalized models that have larger average errors versus more specialized models that have smaller average error but large peak errors. Specializing prediction for a particular application has the potential to improve accuracy, at the cost of generality.

Be careful about developing complicated prediction methods, because the time consumed running the predictor makes the prediction task much harder. The time it takes to run the prediction routine is added directly to the end-to-end system delay. The difficulty of the prediction problem grows rapidly with increasing end-to-end delay. Increasing the prediction interval from 50 ms to 500 ms does not make the problem ten times harder; it makes the problem virtually intractable. Therefore, execution time is a significant factor in predictor design. A complicated predictor that takes 50 ms longer to execute than a simple predictor must be much more accurate than the simple predictor, just to break even in terms of performance!

Will the prediction problem simply vanish in the future? After all, computers get faster every year, and eventually the latency may become so small that prediction is trivial. However, while latencies may become smaller, it is also likely that user motions will become faster, making the prediction problem much harder. Virtual environment equipment will become smaller and lighter, reducing inertia and allowing faster user motions. Maximizing performance out of direct manipulation interfaces may mean that fast user motions must be supported and even encouraged. Certain popular applications, such as entertainment, often encourage fast user motions. So far, research efforts have focused on head motion, but motions of other body parts (such as the user's hands) may be an order of magnitude faster. It is likely that the prediction problem will not go away, but will migrate to a different realm of faster motions across shorter prediction intervals.

References

Albrecht89 — Albrecht, R. E. An Adaptive Digital Filter to Predict Pilot Head Look Direction for Helmet-Mounted Displays. M.S. Thesis, Electrical Engineering, University of Dayton, Ohio (July 1989).

Andres89 — Andres, Robert O. and Kenny J. Hartung. Prediction of Head Movement Time Using Fitts' Law. *Human Factors 31*, 6 (1989), 703-713.

Azuma94 — Azuma, Ronald and Gary Bishop. Improving Static and Dynamic Registration in a See-Through HMD. *Proceedings of SIGGRAPH '94* (Orlando, FL, 24-29 July 1994). In *Computer Graphics*, Annual Conference Series, 1994, 197-204.

Azuma95a — Azuma, Ronald T. Predictive Tracking for Augmented Reality. Ph.D. dissertation. UNC Chapel Hill Department of Computer Science technical report TR95-007 (February 1995).

Azuma95b — Azuma, Ronald and Gary Bishop. A Frequency-Domain Analysis of Head-Motion Prediction. *Proceedings of SIGGRAPH '95* (Los Angeles, CA, 6-11 August 1995). In *Computer Graphics*, Annual Conference Series, 1995, 401-408.

Azuma97 — Azuma, Ronald T. A Survey of Augmented Reality. To appear in *Presence: Teleoperators and Virtual Environments 6*, 4 (Fall 1997). Earlier version appeared in Course Notes #9: Developing Advanced Virtual Reality Applications, ACM SIGGRAPH (Los Angeles, CA, 6-11 August 1995), 20-1 to 20-38.

Brown92 — Brown, Robert Grover. Introduction to random signal analysis and applied Kalman filtering, 2nd edition. John Wiley and Sons (1992). ISBN 0471-52573-1.

Chang84 — Chang, Chaw-Bing, and John Tabaczynski. Application of State Estimation to Target Tracking. *IEEE Transactions of Automatic Control AC-29*, 2 (February 1984), 98-109.

Deering92 — Deering, Michael. High Resolution Virtual Reality. *Proceedings of SIGGRAPH '92* (Chicago, IL, 26-31 July 1992). In *Computer Graphics 26*, 2 (July 1992), 195-202.

Emura94 — 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.

Fleming89 — Fleming, Wendell H., chairman. Report of the Panel on Future Directions in Control Theory: a Mathematical Perspective. Society for Industrial and Applied Mathematics (1989).

Friedmann92 — Friedmann, Martin, Thad Starner, and Alex Pentland. Device Synchronization Using an Optimal Linear Filter. *Proceedings of 1992 Symposium on Interactive 3D Graphics* (Cambridge, MA, 29 March - 1 April 1992). A special issue of *Computer Graphics*, 57-62.

Holloway95 — Holloway, Richard. Registration Errors in Augmented Reality. Ph.D. dissertation. UNC Chapel Hill Department of Computer Science technical report TR95-016 (August 1995).

Jagacinski85 — Jagacinski, Richard J. and Donald L. Monk. Fitts' Law in Two Dimensions with Hand and Head Movements. *Journal of Motor Behavior 17*, 1 (1985), 77-95.

Lewis86 — Lewis, Frank L. Optimal Estimation with an Introduction to Stochastic Control Theory. John Wiley and Sons (1986). ISBN 0-471-83741-5.

Liang91 — 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.

Magill65 — Magill, D. T. Optimal Adaptive Estimation of Sampled Stochastic Processes. *IEEE Transactions on Automatic Control AC-10*, 4 (October 1965), 434-439.

Maybeck79 — Maybeck, Peter S. Stochastic Models, Estimation and Control, Volume 1. Academic Press (1979).

Montgomery90 — Montgomery, Douglas C., Lynwood A. Johnson, and John S. Gardiner. Forecasting and Time Series Analysis (2nd edition). McGraw-Hill (1990). ISBN 0-07-042858-1.

Paley92 — Paley, W. Bradford. Head-Tracking Stereo Display: Experiments and Applications. *SPIE Vol. 1669 Stereoscopic Displays and Applications ///* (San Jose, CA, 12-13 February 1992), 84-89.

Peterson88 — Peterson, Barry W. and Frances J. Richmond. Control of Head Movement. Oxford University Press (1988). ISBN 0-19-504499-1.

Rebo88 — Rebo, Robert. A Helmet-Mounted Virtual Environment Display System. M.S. Thesis, Air Force Institute of Technology (December 1988).

Shaw92 — Shaw, Chris, and Jiandong Liang. An Experiment to Characterize Head Motion in VR and RR using MR. Proceedings of 1992 Western Computer Graphics Symposium (Banff, Alberta, Canada, 6-8 April 1992), 99-101.

Smith84 — Smith Jr., Bernard R. Digital Head Tracking and Position Prediction for Helmet Mounted Visual Display Systems. *Proceedings of AIAA 22nd Aerospace Sciences Meeting* (Reno, NV, 9-12 January 1984) [AIAA 84-0557].

Sorenson70 — Sorenson, Harold W. Least-squares estimation: from Gauss to Kalman. *IEEE Spectrum* (July 1970), 63-68.

Splettstösser82 — Splettstösser, W. On the Prediction of Band-Limited Signals from Past Samples. *Information Sciences 28* (1982), 115-130.

State96 — State, Andrei, Gentaro Hirota, David T. Chen, Bill Garrett, and Mark Livingston. Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. *Proceedings of SIGGRAPH '96* (New Orleans, LA, 4-9 August 1996). In *Computer Graphics*, Annual Conference Series, 1996, 429-438.

Welch86 — Welch, Brian K., Ron V. Kruk, Jean J. Baribeau, Charles L. Schlef, Martin Shenker, and Paul E. Weissman. Flight Simulator: Wide-Field-Of-View Helmet-Mounted Infinity Display System, Air Force Human Resources Laboratory technical report AFHRL-TR-85-59 (May 1986), 48-60.

Wu95 — Wu, Jiann-Rong and Ming Ouhyoung. A 3D Tracking Experiment on Latency and its Compensation Methods in Virtual Environments. *Proceedings of UIST '95* (Pittsburgh, PA, 14-17 November 1995), 41-49.

Zikan94 — 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), 328-336.