SIGGRAPH 2001

Course 8

An Introduction to the Kalman Filter

Greg Welch
Gary Bishop

University of North Carolina at Chapel Hill
Department of Computer Science
Chapel Hill, NC 27599-3175

http://www.cs.unc.edu/~{welch, gb}
{welch, gb}@cs.unc.edu

TABLE OF CONTENTS

Preface

In putting together this course pack we decided not to simply include copies of the slides for the course presentation, but to attempt to put together a small booklet of information that could stand by itself. The course slides and other useful information, including a new Java-based *Kalman Filter Learning Tool* are available at

`http://www.cs.unc.edu/~tracker/ref/s2001/kalman/`

In addition, we maintain a popular web site dedicated to the Kalman filter. This site contains links to related work, papers, books, and even some software.

`http://www.cs.unc.edu/~welch/kalman/`

We expect that you (the reader) have a basic mathematical background, sufficient to understand explanations involving basic linear algebra, statistics, and random signals.

Course Syllabus

Time	Speaker	Topic	**Time**
10:00 AM	Bishop	Welcome, Introduction, Intuition	0:30
10:30 AM	Welch	Concrete examples	0:30
11:00 AM	Bishop	Non-linear estimation	0:15
11:15 AM	Welch	System identification and multi-modal filters	0:30
11:45 AM	Welch	Conclusions (summary, resources, etc.)	0:15
12:00 PM			
		Total time	2:00

1. Introduction

The Kalman filter is a mathematical power tool that is playing an increasingly important role in computer graphics as we include sensing of the real world in our systems. The good news is you don't have to be a mathematical genius to understand and effectively use Kalman filters. This tutorial is designed to provide developers of graphical systems with a basic understanding of this important mathematical tool.

1.1 Course Description

While the Kalman filter has been around for about 30 years, it (and related optimal estimators) have recently started popping up in a wide variety of computer graphics applications. These applications span from simulating musical instruments in VR, to head tracking, to extracting lip motion from video sequences of speakers, to fitting spline surfaces over collections of points.

The Kalman filter is the best possible (optimal) estimator for a large class of problems and a very effective and useful estimator for an even larger class. With a few conceptual tools, the Kalman filter is actually very easy to use. We will present an intuitive approach to this topic that will enable developers to approach the extensive literature with confidence.

1.2 Speaker/Author Biographies

Greg Welch is a Research Assistant Professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. His research interests include hardware and software for man-machine interaction, 3D interactive computer graphics, virtual environments, tracking technologies, tele-immersion, and projector-based graphics. Welch graduated with *highest distinction* from Purdue University with a degree in Electrical Engineering Technology in 1986 and received a Ph.D. in computer science from UNC-Chapel Hill in 1996. Before coming to UNC he worked at NASA's Jet Propulsion Laboratory and Northrop-Grumman's Defense Systems Division. He is a member of the IEEE Computer Society and the Association of Computing Machinery.

Gary Bishop is an Associate Professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. His research interests include hardware and software for man-machine interaction, 3D interactive computer graphics, virtual environments, tracking technologies, and image-based rendering. Bishop graduated with highest honors from the Southern Technical Institute in Marietta, Georgia, with a degree in Electrical Engineering Technology in 1976. He completed his Ph.D. in computer science at UNC-Chapel Hill in 1984. Afterwards he worked for Bell Laboratories and Sun Microsystems before returning to UNC in 1991.

2. Probability and Random Variables

What follows is a very basic introduction to probability and random variables. For more extensive coverage see for example (Maybeck 1979; Brown and Hwang 1996; Kailath, Sayed et al. 2000).

2.1 Probability

Most of us have some notion of what is meant by a "random" occurrence, or the probability that some event in a *sample space* will occur. Formally, the probability that the outcome of a discrete event (e.g., a coin flip) will favor a particular event is defined as

$$p(A) = \frac{\text{Possible outcomes favoring event A}}{\text{Total number of possible outcomes}}.$$

The probability of an outcome favoring either A *or* B is given by

$$p(A \cup B) = p(A) + p(B). \tag{2.1}$$

If the probability of two outcomes is *independent* (one does not affect the other) then the probability of *both* occurring is the product of their individual probabilities:

$$p(A \cap B) = p(A)p(B). \tag{2.2}$$

For example, if the probability of seeing a "heads" on a coin flip is 1/2, then the probability of seeing "heads" on both of two coins flipped at the same time is 1/4. (Clearly the outcome of one coin flip does not affect the other.)

Finally, the probability of outcome A given an occurrence of outcome B is called the *conditional probability* of A given B, and is defined as

$$p(A|B) = \frac{p(A \cap B)}{p(B)}. \tag{2.3}$$

2.2 Random Variables

As opposed to discrete events, in the case of tracking and motion capture, we are more typically interested with the randomness associated with a *continuous* electrical voltage or perhaps a user's motion. In each case we can think of the item of interest as a *continuous*

random variable. A random variable is essentially a function that maps all points in the sample space to real numbers. For example, the continuous random variable $X(t)$ might map time to position. At any point in time, $X(t)$ would tell us the expected position.

In the case of continuos random variables, the probability of any *single* discrete event A is in fact 0. That is, $p(A) = 0$. Instead we can only evaluate the probability of events within some interval. A common function representing the probability of random variables is defined as the *cumulative distribution function*:

$$F_X(x) = p(-\infty, x]. \tag{2.4}$$

This function represents the cumulative probability of the continuous random variable X for all (uncountable) events up to and including x. Important properties of the cumulative density function are

1. $F_X(x) \to 0$ as $x \to -\infty$
2. $F_X(x) \to 1$ as $x \to +\infty$
3. $F_X(x)$ is a non-decreasing function of x.

Even more commonly used than equation (2.4) is its derivative, known as the *probability density function*:

$$f_X(x) = \frac{d}{dx} F_X(x). \tag{2.5}$$

Following on the above given properties of the cumulative probability function, the density function also has the following properties:

1. $f_X(x)$ is a non-negative function
2. $\int_{-\infty}^{\infty} f_X(x)dx = 1$.

Finally note that the probability over any interval $[a, b]$ is defined as

$$p_X[a, b] = \int_a^b f_X(x)dx.$$

So rather than summing the probabilities of discrete events as in equation (2.1), for continuous random variables one integrates the probability density function over the interval of interest.

2.3 Mean and Variance

Most of us are familiar with the notion of the *average* of a sequence of numbers. For some N samples of a discrete random variable X, the average or *sample mean* is given by

$$\bar{X} = \frac{X_1 + X_2 + \ldots + X_N}{N}.$$

Because in tracking we are dealing with continuous signals (with an uncountable sample space) it is useful to think in terms of an infinite number of trials, and correspondingly the outcome we would *expect* to see if we sampled the random variable infinitely, each time seeing one of n possible outcomes $x_1 \ldots x_n$. In this case, the *expected value* of the discrete random variable could be approximated by averaging probability-weighted events:

$$\bar{X} \approx \frac{(p_1 N)x_1 + (p_2 N)x_2 + \ldots + (p_n N)x_N}{N}.$$

In effect, out of N trials, we would expect to see $(p_1 N)$ occurrences of event x_1, etc. This notion of infinite trials (samples) leads to the conventional definition of *expected value* for *discrete* random variables

$$\text{Expected value of } X = E(X) = \sum_{i=1}^{n} p_i x_i \tag{2.6}$$

for n possible outcomes $x_1 \ldots x_n$ and corresponding probabilities $p_1 \ldots p_n$. Similarly for the continuous random variable the expected value is defined as

$$\text{Expected value of } X = E(X) = \int_{-\infty}^{\infty} x f_X(x) dx. \tag{2.7}$$

Finally, we note that equation (2.6) and equation (2.7) can be applied to functions of the random variable X as follows:

$$E(g(X)) = \sum_{i=1}^{n} p_i g(x_i) \tag{2.8}$$

and

$$E(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x) dx. \tag{2.9}$$

The expected value of a random variable is also known as the *first statistical moment*. We can apply the notion of equation (2.8) or (2.9), letting $g(X) = X^k$, to obtain the k^{th} statistical moment. The k^{th} statistical moment of a continuous random variable X is given by

$$E(X^k) = \int_{-\infty}^{\infty} x^k f_X(x)dx . \tag{2.10}$$

Of particular interest in general, and to us in particular, is the *second moment* of the random variable. The second moment is given by

$$E(X^2) = \int_{-\infty}^{\infty} x^2 f_X(x)dx . \tag{2.11}$$

When we let $g(X) = X - E(X)$ and apply equation (2.11), we get the *variance* of the signal about the mean. In other words,

$$\begin{aligned} \text{Variance } X &= E[(X - E(X))^2] \\ &= E(X^2) - E(X)^2 . \end{aligned}$$

Variance is a very useful statistical property for random signals, because if we knew the variance of a signal that was otherwise supposed to be "constant" around some value—the mean, the magnitude of the variance would give us a sense how much jitter or "noise" is in the signal.

The square root of the variance, known as the *standard deviation*, is also a useful statistical unit of measure because while being always positive, it has (as opposed to the variance) the same units as the original signal. The standard deviation is given by

$$\text{Standard deviation of } X = \sigma_X = \sqrt{\text{Variance of } X} .$$

2.4 Normal or Gaussian Distribution

A special probability distribution known as the *Normal* or *Gaussian* distribution has historically been popular in modeling random systems for a variety of reasons. As it turns out, many random processes occurring in nature actually appear to be normally distributed, or very close. In fact, under some moderate conditions, it can be proved that a sum of random variables with *any* distribution tends toward a normal distribution. The theorem that formally states this property is called the *central limit theorem* (Maybeck 1979; Brown and Hwang 1996). Finally, the normal distribution has some nice properties that make it mathematically tractable and even attractive.

Given a random process $X \sim N(\mu, \sigma^2)$, i.e. a continuous random process X that is normally distributed with mean μ and variance σ^2 (standard deviation σ), the probability density function for X is given by

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

for $-\infty < x < \infty$. Any linear function of a normally distributed random process (variable) is also a normally distributed random process. In particular if $X \sim N(\mu, \sigma^2)$ and $Y = aX + b$, then

$$Y \sim N(a\mu + b, a^2\sigma^2). \tag{2.12}$$

The probability density function for Y is then given by

$$f_Y(y) = \frac{1}{\sqrt{2\pi a^2\sigma^2}} e^{-\frac{1}{2}\frac{(y-(a\mu+b))^2}{a^2\sigma^2}}. \tag{2.13}$$

Finally, if X_1 and X_2 are independent (see Section 2.5 below), $X_1 \sim N(\mu_1, \sigma_1^2)$, and $X_2 \sim N(\mu_2, \sigma_2^2)$, then

$$X_1 + X_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2), \tag{2.14}$$

and the density function becomes

$$f_X(x_1 + x_2) = \frac{1}{\sqrt{2\pi(\sigma_1^2 + \sigma_2^2)}} e^{-\frac{1}{2}\frac{(x-(\mu_1+\mu_2))^2}{(\sigma_1^2+\sigma_2^2)}}. \tag{2.15}$$

See (Kelly 1994) pp. 351-358 for further explanation and proofs of the above. Graphically, the normal distribution is what is likely to be familiar as the “bell-shaped” curve shown below in Figure 2.1.

2.5 Continuous Independence and Cond. Probability

Finally we note that as with the discrete case and equations (2.2) and (2.3), independence and conditional probability are defined for continuous random variables. Two continuous random variables X and Y are said to be *statistically independent* if their *joint* probability $f_{XY}(x, y)$ is equal to the product of their individual probabilities. In other words, they are considered independent if

$$f_{XY}(x, y) = f_X(x) f_Y(y).$$

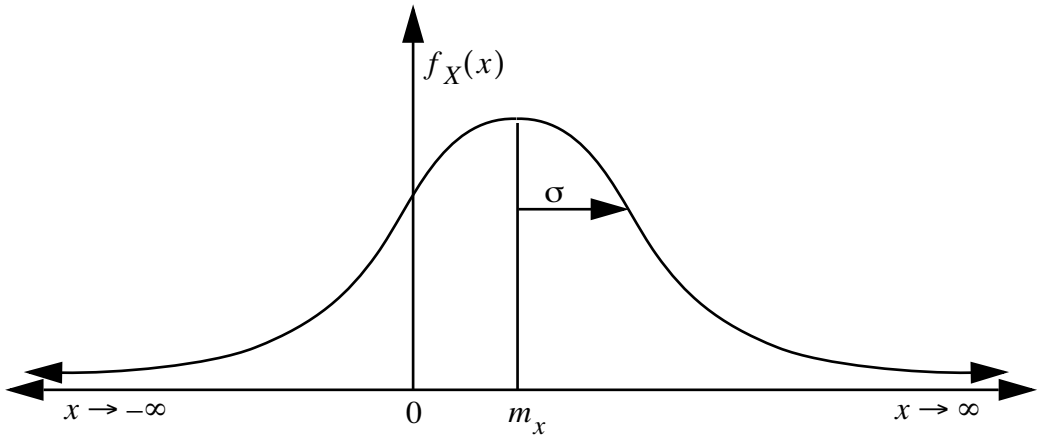

Figure 2.1: The Normal or Gaussian probability distribution function.

Bayes' Rule

In addition, Bayes' rule follows from (2.3), offering a way to specify the probability density of the random variable X given (in the presence of) random variable Y. Bayes' rule is given as

$$f_{X|Y}(x) = \frac{f_{Y|X}(y) f_X(x)}{f_Y(y)}.$$

Continuous-Discrete

Given a *discrete* process X and a *continuous* process Y, the discrete probability mass function for X conditioned on $Y = y$ is given by

$$p_X(x \mid Y = y) = \frac{f_Y(y \mid X = x) p_X(x)}{\sum_z f_Y(y \mid X = z) p_X(z)}. \tag{2.16}$$

Note that this formula provides a discrete probability based on the conditioning density, *without any integration*. See (Kelly 1994) p. 546 for further explanation and proofs of the above.

2.6 Spatial vs. Spectral Signal Characteristics

In the previous sections we looked only at the *spatial* characteristics of random signals. As stated earlier, the magnitude of the variance of a signal can give us a sense of how much jitter or "noise" is in the signal. However a signal's variance says nothing about the

spacing or the rate of the jitter over time. Here we briefly discuss the temporal and hence *spectral* characteristics of a random signal. Such discussion can be focused in the time or the frequency domain. We will look briefly at both.

A useful time-related characteristic of a random signal is its *autocorrelation*—its correlation with itself over time. Formally the autocorrelation of a random signal $X(t)$ is defined as

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)] \tag{2.17}$$

for sample times t_1 and t_2. If the process is *stationary* (the density is invariant with time) then equation (2.17) depends only on the difference $\tau = t_1 - t_2$. In this common case the autocorrelation can be re-written as

$$R_X(\tau) = E[X(t)X(t+\tau)]. \tag{2.18}$$

Two hypothetical autocorrelation functions are shown below in Figure 2.1. Notice how compared to random signal X_2, random signal X_1 is relatively short and wide. As $|\tau|$ increases (as you move away from $\tau = 0$ at the center of the curve) the autocorrelation signal for X_2 drops off relatively quickly. This indicates that X_2 is less correlated with itself than X_1.

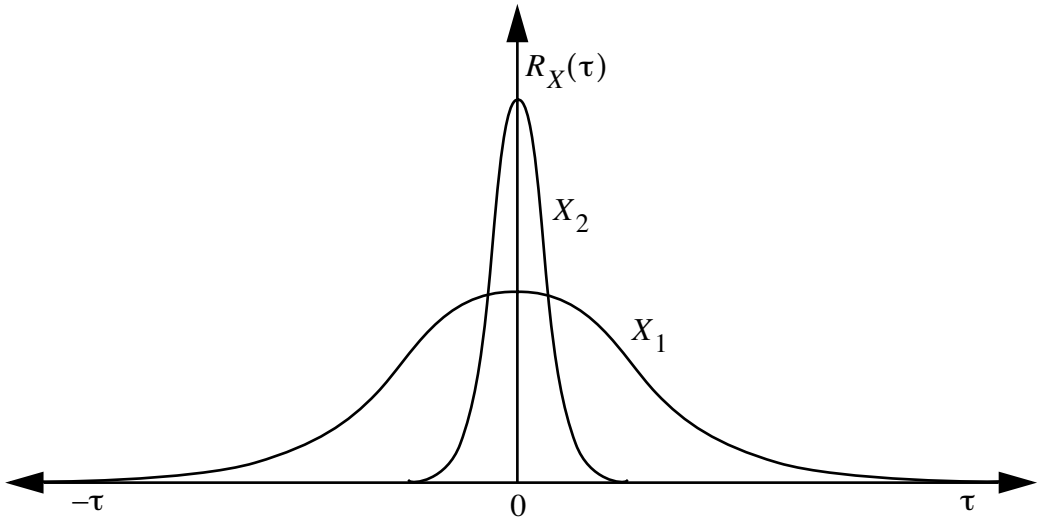

Figure 2.2: Two example (hypothetical) autocorrelation functions X_1 and X_2.

Clearly the autocorrelation is a function of time, which means that it has a spectral interpretation in the frequency domain also. Again for a stationary process, there is an important temporal-spectral relationship known as the *Wiener-Khinchine relation*:

$$S_X(j\omega) = \Im[R_X(\tau)] = \int_{-\infty}^{\infty} R_X(\tau)e^{-j\omega\tau}d\tau$$

where $\Im[\bullet]$ indicates the Fourier transform, and ω indicates the number of (2π) cycles per second. The function $S_X(j\omega)$ is called the *power spectral density* of the random signal. As you can see, this important relationship ties together the time and frequency spectrum representations of the same signal.

White Noise

An important case of a random signal is the case where the autocorrelation function is a *dirac delta* function $\delta(\tau)$ which has zero value everywhere except when $\tau = 0$. In other words, the case where

$$R_X(\tau) = \begin{cases} \text{if } \tau = 0 \text{ then } A \\ \text{else } 0 \end{cases}$$

for some constant magnitude A. In this special case where the autocorrelation is a "spike" the Fourier transform results in a *constant* frequency spectrum. as shown in Figure 2.3. This is in fact a description of *white noise*, which be thought of both as having power at all

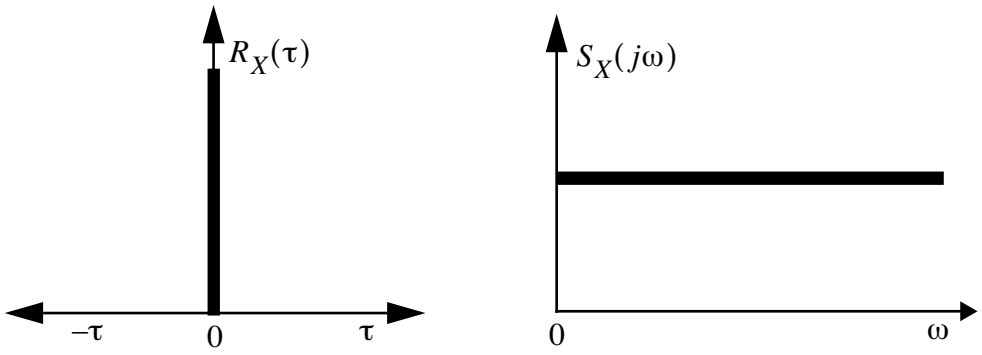

Figure 2.3: White noise shown in both the time (left) and frequency domain (right).

frequencies in the spectrum, and being completely uncorrelated with itself at any time except the present $(\tau = 0)$. This latter interpretation is what leads white noise signals to be called *independent*. Any sample of the signal at one time is completely independent (uncorrelated) from a sample at any other time.

While impossible to achieve or see in practice (no system can exhibit infinite energy throughout an infinite spectrum), white noise is an important building block for design and analysis. Often random signals can be modeled as filtered or *shaped* white noise. Literally this means that one could filter the output of a (hypothetical) white noise source to achieve a non-white or *colored* noise source that is both band-limited in the frequency domain, and more correlated in the time domain.

3. Stochastic Estimation

While there are many application-specific approaches to "computing" (estimating) an unknown state from a set of process measurements, many of these methods do not inherently take into consideration the typically *noisy* nature of the measurements. For example, consider our work in tracking for interactive computer graphics. While the requirements for the tracking information varies with application, the fundamental source of information is the same: pose estimates are derived from noisy electrical measurements of mechanical, inertial, optical, acoustic, or magnetic sensors. This noise is typically statistical in nature (or can be effectively modeled as such), which leads us to *stochastic* methods for addressing the problems. Here we provide a very basic introduction to the subject, primarily aimed at preparing the reader for Chapter 4. For a more extensive discussion of stochastic estimation see for example (Lewis 1986; Kailath, Sayed et al. 2000).

3.1 State-Space Models

State-space models are essentially a notational convenience for estimation and control problems, developed to make tractable what would otherwise be a notationally-intractable analysis. Consider a dynamic process described by an n-th order difference equation (similarly a differential equation) of the form

$$y_{i+1} = a_{0,i} y_i + \ldots + a_{n-1,i} y_{i-n+1} + u_i, \; i \geq 0,$$

where $\{u_i\}$ is a *zero-mean* (statistically) *white* (spectrally) random "noise" process with autocorrelation

$$E(u_i, u_j) = R_u = Q_i \delta_{ij},$$

and initial values $\{y_0, y_{-1}, \ldots, y_{-n+1}\}$ are zero-mean random variables with a known $n \times n$ *covariance matrix*

$$P_0 = E(y_{-j}, y_{-k}), \; j, k \in \{0, n-1\}.$$

Also assume that

$$E(u_i, y_i) = 0 \text{ for } -n+1 \leq j \leq 0 \text{ and } i \geq 0,$$

which ensures (Kailath, Sayed et al. 2000) that

$$E(u_i, y_i) = 0\,,\; i \geq j \geq 0\,.$$

In other words, that the noise is statistically independent from the process to be estimated. Under some other basic conditions (Kailath, Sayed et al. 2000) this difference equation can be re-written as

$$\vec{x}_{i+1} \equiv \begin{bmatrix} y_{i+1} \\ y_i \\ y_{i-1} \\ \vdots \\ y_{i-n+2} \end{bmatrix} = \underbrace{\begin{bmatrix} a_0 & a_1 & \dots & a_{n-2} & a_{n-1} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} y_i \\ y_{i-1} \\ y_{i-2} \\ \vdots \\ y_{i-n+1} \end{bmatrix}}_{\vec{x}_i} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{G} u_i$$

which leads to the *state-space model*

$$\vec{x}_{i+1} = A\vec{x}_i + Gu_i$$

$$\vec{y}_i = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \vec{x}_i$$

or the more general form

$$\vec{x}_{i+1} = A\vec{x}_i + Gu_i \tag{3.1}$$

$$\vec{y}_i = H_i\vec{x}_i\,. \tag{3.2}$$

Equation (3.1) represents the way a new state $\vec{x}_{i+1}$ is modeled as a linear combination of both the previous state $\vec{x}_i$ and some *process noise* u_i. Equation (3.2) describes the way the process measurements or *observations* $\vec{y}_i$ are derived from the internal state $\vec{x}_i$. These two equations are often referred to respectively as the *process model* and the *measurement model*, and they serve as the basis for virtually all linear estimation methods, such as the *Kalman filter* described below.

3.2 The Observer Design Problem

There is a related general problem in the area of linear systems theory generally called the *observer design problem*. The basic problem is to determine (estimate) the internal *states* of a linear system, given access only to the system's *outputs*. (Access to the system's control inputs is also presumed, but we omit that aspect here. See for example (Kailath, Sayed et al. 2000) for more information.) This is akin to what people often think of as the "black box" problem where you have access to some signals coming from the box (the outputs) but you cannot directly observe what's inside.

The many approaches to this basic problem are typically based on the state-space model presented in the previous section. There is typically a *process model* that models the transformation of the process state. This can usually be represented as a linear stochastic difference equation similar to equation (3.1):

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}. \tag{3.3}$$

In addition there is some form of *measurement model* that describes the relationship between the process state and the measurements. This can usually be represented with a linear expression similar to equation (3.2):

$$z_k = Hx_k + v_k. \tag{3.4}$$

The terms w_k and v_k are random variables representing the process and measurement noise respectively. Note that in equation (3.4) we changed the dependent variable to z_k instead of y_k as in equation (3.2). The rationale is to reinforce the notion that the measurements to not have to be of elements of the state specifically, but can be any linear combination of the state elements.

Measurement and Process Noise

We consider here the common case of noisy sensor measurements. There are many sources of noise in such measurements. For example, each type of sensor has fundamental limitations related to the associated physical medium, and when pushing the envelope of these limitations the signals are typically degraded. In addition, some amount of random electrical noise is added to the signal via the sensor and the electrical circuits. The time-varying ratio of "pure" signal to the electrical noise continuously affects the *quantity* and *quality* of the information. The result is that information obtained from any one sensor must be qualified as it is interpreted as part of an overall sequence of estimates, and analytical measurement models typically incorporate some notion of random measurement noise or uncertainty as shown above.

There is the additional problem that the actual state transform model is completely unknown. While we can make predictions over relatively short intervals using models based on recent state transforms, such predictions assume that the transforms are predictable, which is not always the case. The result is that like sensor information, ongoing estimates of the state must be qualified as they are combined with measurements in an overall sequence of estimates. In addition, process models typically incorporate some notion of random motion or uncertainty as shown above.

4. The Kalman Filter

Within the significant toolbox of mathematical tools that can be used for stochastic estimation from noisy sensor measurements, one of the most well-known and often-used tools is what is known as the *Kalman filter*. The Kalman filter is named after Rudolph E. Kalman, who in 1960 published his famous paper describing a recursive solution to the discrete-data linear filtering problem (Kalman 1960). A very "friendly" introduction to the general idea of the Kalman filter is offered in Chapter 1 of (Maybeck 1979)—which is available from the above Kalman filter web site, *and* we have included it (with permission) in this course pack. A more complete introductory discussion can be found in (Sorenson 1970), which also contains some interesting historical narrative. More extensive references include (Gelb 1974; Maybeck 1979; Lewis 1986; Jacobs 1993; Brown and Hwang 1996; Grewal and Andrews 2001). In addition, for many years we have maintained a web site dedicated to the Kalman filter. This site contains links to related work, papers, books, and even some software including a new Java-based *Kalman Filter Learning Tool*.

`http://www.cs.unc.edu/~welch/kalman/`

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated *error* covariance—when some presumed conditions are met. Since the time of its introduction, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. This is likely due in large part to advances in digital computing that made the use of the filter practical, but also to the relative simplicity and robust nature of the filter itself. Rarely do the conditions necessary for optimality actually exist, and yet the filter apparently works well for many applications in spite of this situation.

Of particular note here, the Kalman filter has been used extensively for tracking in interactive computer graphics. We use a *single-constraint-at-a-time* Kalman filter (see Section 5.4 on page 41) in our HiBall Tracking System (Welch, Bishop et al. 1999; Welch, Bishop et al. 2001) which is commercially available from 3rdTech (3rdTech 2000). It has also been used for motion prediction (Azuma and Bishop 1994; Azuma 1995), and it is used for multi-sensor (inertial-acoustic) fusion in the commercial Constellation™ wide-area tracking system by Intersense (Foxlin, Harrington et al. 1998; Intersense 2000). See also (Fuchs (Foxlin) 1993; Van Pabst and Krekel 1993; Azarbayejani and Pentland 1994; Emura and Tachi 1994; Emura and Tachi 1994; Mazuryk and Gervautz 1995).

4.1 The Discrete Kalman Filter

This section describes the filter in its original formulation (Kalman 1960) where the measurements occur and the state is estimated at discrete points in time.

4.1.1 The Process to be Estimated

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}, \tag{4.1}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k. \tag{4.2}$$

The random variables w_k and v_k represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0, Q), \tag{4.3}$$

$$p(v) \sim N(0, R). \tag{4.4}$$

In practice, the *process noise covariance* Q and *measurement noise covariance* R matrices might change with each time step or measurement, however here we assume they are constant.

The $n \times n$ matrix A in the difference equation equation (4.1) relates the state at the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. Note that in practice A might change with each time step, but here we assume it is constant. The $n \times l$ matrix B relates the optional control input $u \in \Re^l$ to the state x. The $m \times n$ matrix H in the measurement equation equation (4.2) relates the state to the measurement z_k. In practice H might change with each time step or measurement, but here we assume it is constant.

4.1.2 The Computational Origins of the Filter

We define $\hat{x}_k^- \in \Re^n$ (note the "super minus") to be our *a priori* state estimate at step k given knowledge of the process prior to step k, and $\hat{x}_k \in \Re^n$ to be our *a posteriori* state estimate at step k given measurement z_k. We can then define *a priori* and *a posteriori* estimate errors as

$$e_k^- \equiv x_k - \hat{x}_k^-, \text{ and}$$

$$e_k \equiv x_k - \hat{x}_k.$$

The *a priori* estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}], \tag{4.5}$$

and the *a posteriori* estimate error covariance is

$$P_k = E[e_k e_k^T]. \tag{4.6}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$ as shown below in equation (4.7). Some justification for equation (4.7) is given in "The Probabilistic Origins of the Filter" found below.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{4.7}$$

The difference $(z_k - H\hat{x}_k^-)$ in equation (4.7) is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K in equation (4.7) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance equation (4.6). This minimization can be accomplished by first substituting equation (4.7) into the above definition for e_k, substituting that into equation (4.6), performing the indicated expectations, taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for K. For more details see (Maybeck 1979; Jacobs 1993; Brown and Hwang 1996). One form of the resulting K that minimizes equation (4.6) is given by[1]

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \frac{P_k^- H^T}{H P_k^- H^T + R} \end{aligned} . \tag{4.8}$$

Looking at equation (4.8) we see that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}.$$

1. All of the Kalman filter equations can be algebraically manipulated into to several forms. Equation equation (4.8) represents the Kalman gain in one popular form.

On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K_k = 0 .$$

Another way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance P_k^- approaches zero the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

4.1.3 The Probabilistic Origins of the Filter

The justification for equation (4.7) is rooted in the probability of the *a priori* estimate $\hat{x}_k^-$ conditioned on all prior measurements z_k (Bayes' rule). For now let it suffice to point out that the Kalman filter maintains the first two moments of the state distribution,

$$E[x_k] = \hat{x}_k$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k .$$

The *a posteriori* state estimate equation (4.7) reflects the mean (the first moment) of the state distribution— it is normally distributed if the conditions of equation (4.3) and equation (4.4) are met. The *a posteriori* estimate error covariance equation (4.6) reflects the variance of the state distribution (the second non-central moment). In other words,

$$\begin{aligned} p(x_k | z_k) &\sim N(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]) \\ &= N(\hat{x}_k, P_k) . \end{aligned}$$

For more details on the probabilistic origins of the Kalman filter, see (Brown and Hwang 1996).

4.1.4 The Discrete Kalman Filter Algorithm

We will begin this section with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter (see the previous footnote). After presenting this high-level view, we will narrow the focus to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance

estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown below in Figure 4.1.

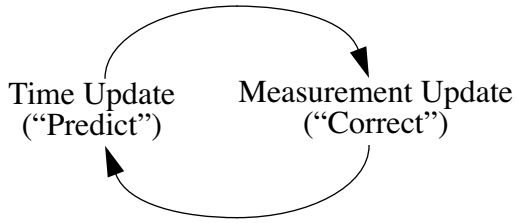

Figure 4.1: The ongoing discrete Kalman filter cycle. The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time and measurement updates are presented below in table 4.1 and table 4.2.

Table 4.1: Discrete Kalman filter time update equations.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{4.9}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4.10}$$

Again notice how the time update equations in table 4.1 project the state and covariance estimates forward from time step $k-1$ to step k. A and B are from equation (4.1), while Q is from equation (4.3). Initial conditions for the filter are discussed in the earlier references.

Table 4.2: Discrete Kalman filter measurement update equations.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.11}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.12}$$

$$P_k = (I - K_k H)P_k^- \tag{4.13}$$

The first task during the measurement update is to compute the Kalman gain, K_k. Notice that the equation given here as equation (4.11) is the same as equation (4.8). The next step is to actually measure the process to obtain z_k, and then to generate an *a posteriori* state estimate by incorporating the measurement as in equation (4.12). Again equation (4.12) is simply equation (4.7) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via equation (4.13).

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible than (for example) an implementation of a Wiener filter (Brown and Hwang 1996) which is designed to operate on *all* of the data *directly* for each estimate. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure 4.2 below offers a complete picture of the operation of the filter, combining the high-level diagram of Figure 4.1 with the equations from table 4.1 and table 4.2.

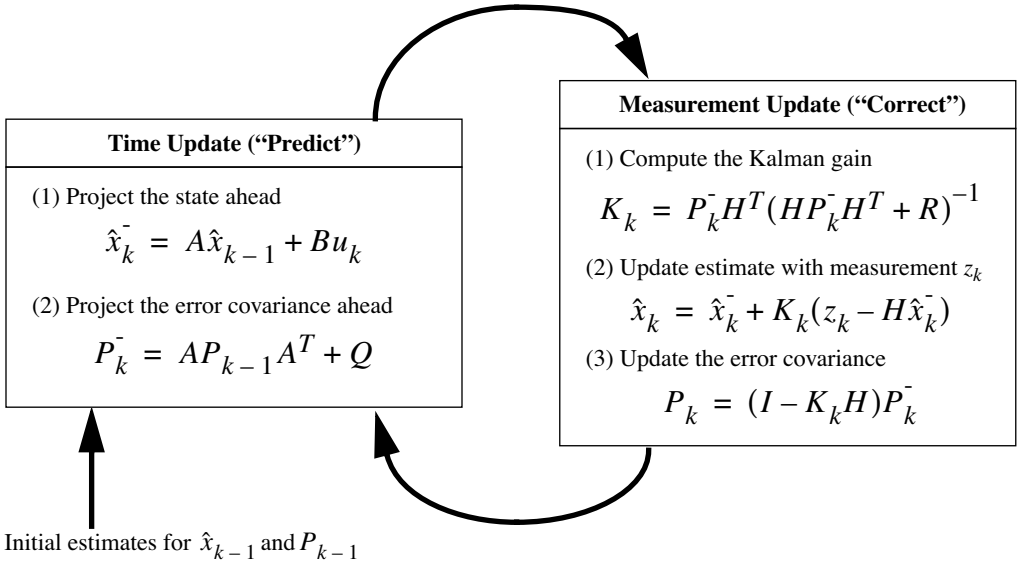

Figure 4.2: A complete picture of the operation of the Kalman filter, combining the high-level diagram of Figure 4.1 with the equations from table 4.1 and table 4.2.

4.2 The Extended Kalman Filter (EKF)

4.2.1 The Process to be Estimated

As described above in Section 4.1.1, the Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a *linear* stochastic difference equation. But what happens if the process to be estimated and (or) the measurement relationship to the process is non-linear? Some of the most

interesting and successful applications of Kalman filtering have been such situations. A Kalman filter that linearizes about the current mean and covariance is referred to as an *extended Kalman filter* or EKF.

In something akin to a Taylor series, we can linearize the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships. To do so, we must begin by modifying some of the material presented in Section 4.1. Let us assume that our process again has a state vector $x \in \Re^n$, but that the process is now governed by the *non-linear* stochastic difference equation

$$x_k = f(x_{k-1}, u_k, w_{k-1}), \tag{4.14}$$

with a measurement $z \in \Re^m$ that is

$$z_k = h(x_k, v_k), \tag{4.15}$$

where the random variables w_k and v_k again represent the process and measurement noise as in equation (4.3) and equation (4.4). In this case the *non-linear* function f in the difference equation equation (4.14) relates the state at the previous time step $k-1$ to the state at the current time step k. It includes as parameters any driving function u_k and the zero-mean process noise w_k. The *non-linear* function h in the measurement equation equation (4.15) relates the state x_k to the measurement z_k.

In practice of course one does not know the individual values of the noise w_k and v_k at each time step. However, one can approximate the state and measurement vector without them as

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0) \tag{4.16}$$

and

$$\tilde{z}_k = h(\tilde{x}_k, 0), \tag{4.17}$$

where $\hat{x}_k$ is some *a posteriori* estimate of the state (from a previous time step k).

It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply an *ad hoc* state estimator that only approximates the optimality of Bayes' rule by linearization. Some interesting work has been done by Julier et al. in developing a variation to the EKF, using methods that preserve the normal distributions throughout the non-linear transformations (Julier and Uhlmann 1996).

4.2.2 The Computational Origins of the Filter

To estimate a process with non-linear difference and measurement relationships, we begin by writing new governing equations that linearize an estimate about equation (4.16) and equation (4.17),

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}\,, \tag{4.18}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k\,. \tag{4.19}$$

where

- x_k and z_k are the actual state and measurement vectors,
- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors from equation (4.16) and equation (4.17),
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step k,
- the random variables w_k and v_k represent the process and measurement noise as in equation (4.3) and equation (4.4).
- A is the Jacobian matrix of partial derivatives of f with respect to x, that is

$$A_{[i,\,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_k, u_k, 0)\,,$$

- W is the Jacobian matrix of partial derivatives of f with respect to w,

$$W_{[i,\,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_k, u_k, 0)\,,$$

- H is the Jacobian matrix of partial derivatives of h with respect to x,

$$H_{[i,\,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0)\,,$$

- V is the Jacobian matrix of partial derivatives of h with respect to v,

$$V_{[i,\,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}_k, 0)\,.$$

Note that for simplicity in the notation we do not use the time step subscript k with the Jacobians A, W, H, and V, even though they are in fact different at each time step.

Now we define a new notation for the prediction error,

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k \,, \tag{4.20}$$

and the measurement residual,

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k \,. \tag{4.21}$$

Remember that in practice one does not have access to x_k in equation (4.20), it is the *actual* state vector, i.e. the quantity one is trying to estimate. On the other hand, one *does* have access to z_k in equation (4.21), it is the actual measurement that one is using to estimate x_k. Using equation (4.20) and equation (4.21) we can write governing equations for an *error process* as

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k \,, \tag{4.22}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k \,, \tag{4.23}$$

where ε_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VRV^T, with Q and R as in (4.3) and (4.4) respectively.

Notice that the equations equation (4.22) and equation (4.23) are linear, and that they closely resemble the difference and measurement equations equation (4.1) and equation (4.2) from the discrete Kalman filter. This motivates us to use the actual measurement residual $\tilde{e}_{z_k}$ in equation (4.21) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{e}_{x_k}$ given by equation (4.22). This estimate, call it $\hat{e}_k$, could then be used along with equation (4.20) to obtain the *a posteriori* state estimates for the original non-linear process as

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \,. \tag{4.24}$$

The random variables of equation (4.22) and equation (4.23) have approximately the following probability distributions (see the previous footnote):

$$p(\tilde{e}_{x_k}) \sim N(0, E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T])$$

$$p(\varepsilon_k) \sim N(0, WQ_kW^T)$$

$$p(\eta_k) \sim N(0, VR_kV^T)$$

Given these approximations and letting the predicted value of $\hat{e}_k$ be zero, the Kalman filter equation used to estimate $\hat{e}_k$ is

$$\hat{e}_k = K_k\tilde{e}_{z_k} \,. \tag{4.25}$$

By substituting equation (4.25) back into equation (4.24) and making use of equation (4.21) we see that we do not actually need the second (hypothetical) Kalman filter:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k \tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k (z_k - \tilde{z}_k) \end{aligned} \tag{4.26}$$

Equation equation (4.26) can now be used for the measurement update in the extended Kalman filter, with $\tilde{x}_k$ and $\tilde{z}_k$ coming from equation (4.16) and equation (4.17), and the Kalman gain K_k coming from equation (4.11) with the appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below in table 4.3 and table 4.4. Note that we have substituted $\hat{x}_k^-$ for $\tilde{x}_k$ to remain consistent with the earlier "super minus" a priori notation, and that we now attach the subscript k to the Jacobians A, W, H, and V, to reinforce the notion that they are different at (and therefore must be recomputed at) each time step.

Table 4.3: EKF time update equations.

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{4.27}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{4.28}$$

As with the basic discrete Kalman filter, the time update equations in table 4.3 project the state and covariance estimates from the previous time step $k-1$ to the current time step k. Again f in equation (4.27) comes from equation (4.16), A_k and W_k are the process Jacobians at step k, and Q_k is the process noise covariance equation (4.3) at step k.

Table 4.4: EKF measurement update equations.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{4.29}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \tag{4.30}$$

$$P_k = (I - K_k H_k) P_k^- \tag{4.31}$$

As with the basic discrete Kalman filter, the measurement update equations in table 4.4 correct the state and covariance estimates with the measurement z_k. Again h in equation (4.30) comes from equation (4.17), H_k and V are the measurement Jacobians at step k, and R_k is the measurement noise covariance equation (4.4) at step k. (Note we now subscript R allowing it to change with each measurement.)

The basic operation of the EKF is the same as the linear discrete Kalman filter as shown in Figure 4.1. Figure 4.3 below offers a complete picture of the operation of the EKF, combining the high-level diagram of Figure 4.1 with the equations from table 4.3 and table 4.4.

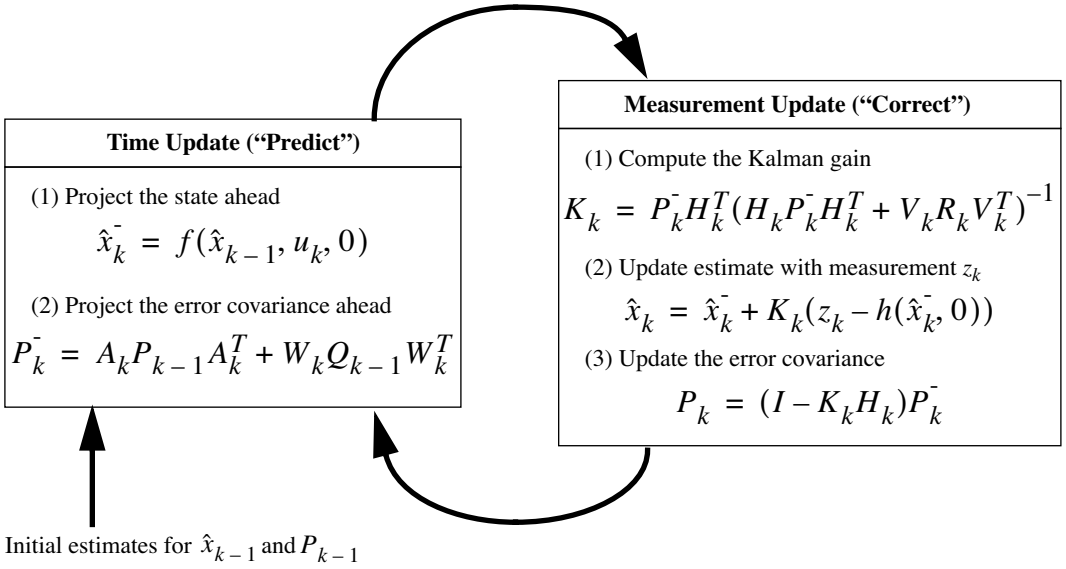

Figure 4.3: A complete picture of the operation of the *extended* Kalman filter, combining the high-level diagram of Figure 4.1 with the equations from table 4.3 and table 4.4.

An important feature of the EKF is that the Jacobian H_k in the equation for the Kalman gain K_k serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_k and the state via h, the Jacobian H_k affects the Kalman gain so that it only magnifies the portion of the residual $z_k - h(\hat{x}_k, 0)$ that does affect the state. Of course if over *all* measurements there is *not* a one-to-one mapping between the measurement z_k and the state via h, then as you might expect the filter will quickly diverge. In this case the process is *unobservable*.

4.3 An Example: Estimating a Random Constant

In the previous two sections we presented the basic form for the discrete Kalman filter, and the extended Kalman filter. To help in developing a better feel for the operation and capability of the filter, we present a very simple example here.

4.3.1 The Process Model

In this simple example let us attempt to estimate a scalar random constant, a voltage for example. Let's assume that we have the ability to take measurements of the constant, but that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise (e.g. our analog to digital converter is not very accurate). In this example, our process is governed by the linear difference equation

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_k + w_k \\ &= x_{k-1} + w_k \end{aligned},$$

with a measurement $z \in \Re^1$ that is

$$\begin{aligned} z_k &= Hx_k + v_k \\ &= x_k + v_k \end{aligned}.$$

The state does not change from step to step so $A = 1$. There is no control input so $u = 0$. Our noisy measurement is of the state directly so $H = 1$. (Notice that we dropped the subscript k in several places because the respective parameters remain constant in our simple model.)

4.3.2 The Filter Equations and Parameters

Our time update equations are

$$\hat{x}_k^- = \hat{x}_{k-1},$$

$$P_k^- = P_{k-1} + Q,$$

and our measurement update equations are

$$\begin{aligned} K_k &= P_k^-(P_k^- + R)^{-1} \\ &= \frac{P_k^-}{P_k^- + R} \end{aligned}, \tag{4.32}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-),$$

$$P_k = (1 - K_k)P_k^-.$$

Presuming a very small process variance, we let $Q = 1e-5$. (We could certainly let $Q = 0$ but assuming a small but non-zero value gives us more flexibility in "tuning" the filter as we will demonstrate below.) Let's assume that from experience we know that the

true value of the random constant has a standard normal probability distribution, so we will "seed" our filter with the guess that the constant is 0. In other words, before starting we let $\hat{x}_{k-1} = 0$.

Similarly we need to choose an initial value for P_{k-1}, call it P_0. If we were absolutely certain that our initial state estimate $\hat{x}_0 = 0$ was correct, we would let $P_0 = 0$. However given the uncertainty in our initial estimate $\hat{x}_0$, choosing $P_0 = 0$ would cause the filter to initially and always believe $\hat{x}_k = 0$. As it turns out, the alternative choice is not critical. We could choose almost any $P_0 \neq 0$ and the filter would *eventually* converge. We'll start our filter with $P_0 = 1$.

4.3.3 The Simulations

To begin with, we randomly chose a scalar constant $z = -0.37727$ (there is no "hat" on the z because it represents the "truth"). We then simulated 50 distinct measurements z_k that had error normally distributed around zero with a standard deviation of 0.1 (remember we presumed that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise). We could have generated the individual measurements within the filter loop, but pre-generating the set of 50 measurements allowed me to run several simulations with the same exact measurements (i.e. same measurement noise) so that comparisons between simulations with different parameters would be more meaningful.

In the first simulation we fixed the measurement variance at $R = (0.1)^2 = 0.01$. Because this is the "true" measurement error variance, we would expect the "best" performance in terms of balancing responsiveness and estimate variance. This will become more evident in the second and third simulation. Figure 4.4 depicts the results of this first simulation. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

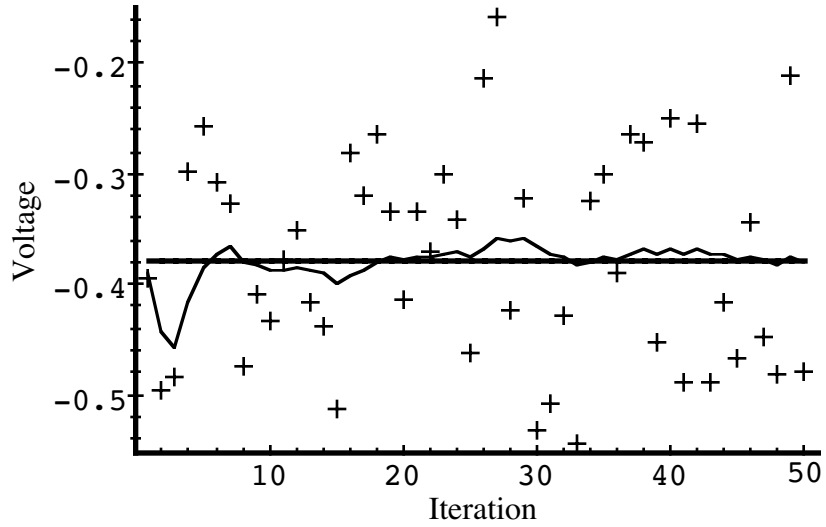

Figure 4.4: The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

When considering the choice for P_0 above, we mentioned that the choice was not critical as long as $P_0 \neq 0$ because the filter would eventually converge. Below in Figure 4.5 we have plotted the value of P_k versus the iteration. By the 50^{th} iteration, it has settled from the initial (rough) choice of 1 to approximately 0.0002 (Volts^2).

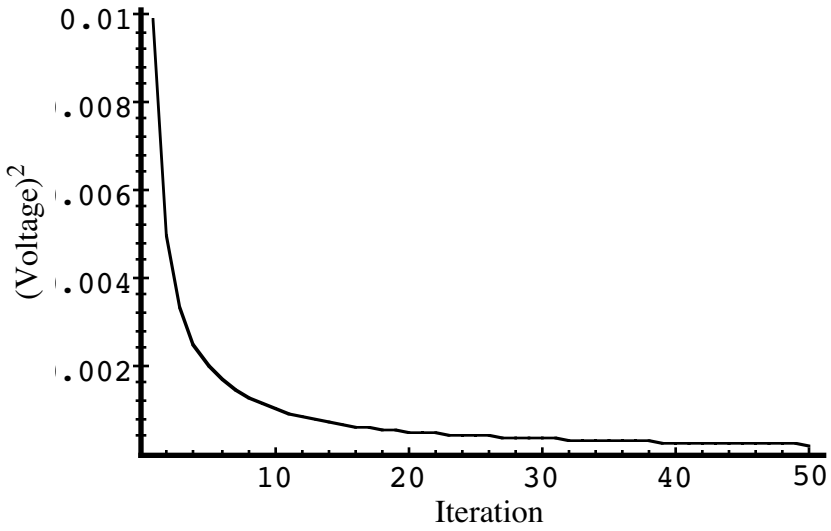

Figure 4.5: After 50 iterations, our initial (rough) error covariance P_k choice of 1 has settled to about 0.0002 (Volts^2).

In Section 5.1 under the topic “Parameter Estimation or Tuning” we briefly discussed changing or “tuning” the parameters Q and R to obtain different filter performance. In Figure 4.6 and Figure 4.7 below we can see what happens when R is increased or

decreased by a factor of 100 respectively. In Figure 4.6 the filter was told that the measurement variance was 100 times greater (i.e. $R = 1$) so it was "slower" to believe the measurements.

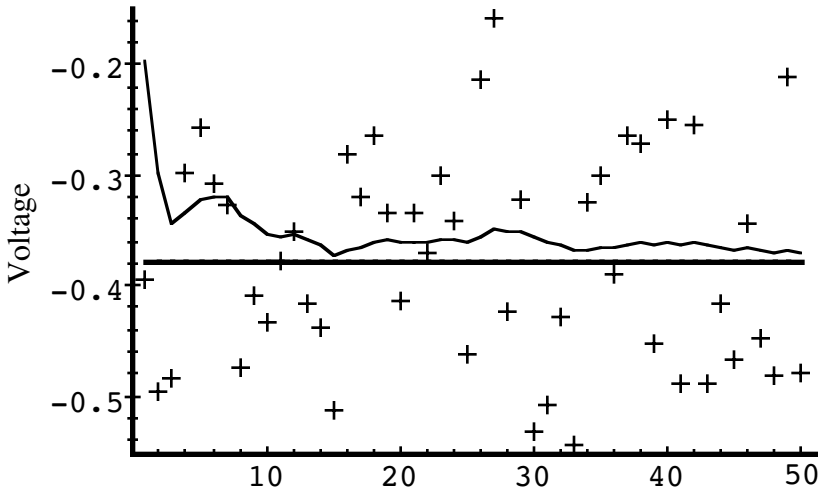

Figure 4.6: Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

In Figure 4.7 the filter was told that the measurement variance was 100 times smaller (i.e. $R = 0.0001$) so it was very "quick" to believe the noisy measurements.

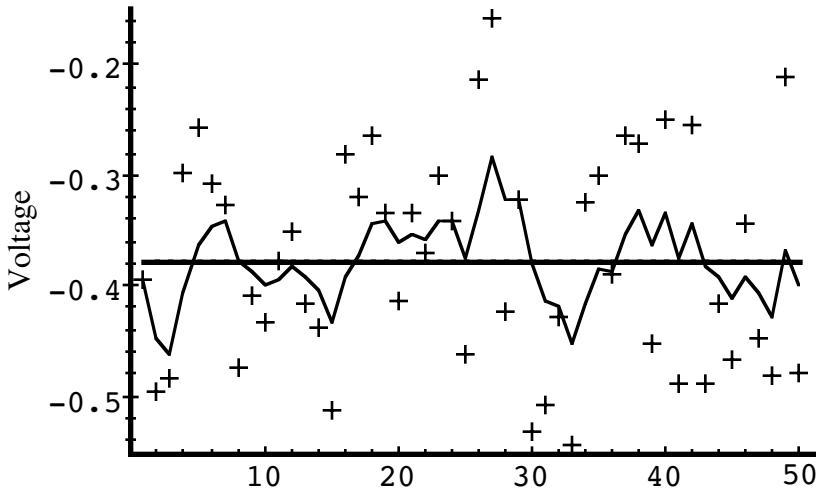

Figure 4.7: Third simulation: $R = 0.0001$. The filter responds to measurements quickly, increasing the estimate variance.

While the estimation of a constant is relatively straight-forward, it clearly demonstrates the workings of the Kalman filter. In Figure 4.6 in particular the Kalman "filtering" is evident as the estimate appears considerably smoother than the noisy measurements.

5. Other Topics

5.1 Parameter Estimation or Tuning

In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. Measuring the measurement error covariance R is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise.

The determination of the process noise covariance Q is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainty into the process via the selection of Q. Certainly in this case one would hope that the process measurements are reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by *tuning* the filter parameters Q and R. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to as *system identification*.

Under conditions where Q and R.are in fact constant, both the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant. If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of P_k as described in (Grewal and Andrews 2001).

It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting beacons in our optoelectronic tracker ceiling panels, the noise in measurements of nearby beacons will be smaller than that in far-away beacons. Also, the process noise Q is sometimes changed dynamically during filter operation—becoming Q_k—in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of Q_k if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases Q_k might be chosen to account for both uncertainty about the user's intentions and uncertainty in the model.

See (Welch 1996) for more information on this topic.

5.2 Multi-Modal (Multiple Model) Filters

5.2.1 Random Processes and the Kalman Filter

The Kalman filter is based on the assumption of a continuous system that can be modeled as a normally distributed random process X, with mean $\hat{\vec{x}}$ (the state) and variance P (the error covariance). In other words,

$$X \sim N(\hat{\vec{x}}, P) .$$

Similarly the Kalman filter is based on the assumption that the output of the system can be modeled as a random process Z which is a linear function of the state $\hat{\vec{x}}$ plus an independent, normally distributed, zero-mean white noise process V,[1]

$$\vec{z} = H\hat{\vec{x}} + \vec{v} \tag{5.1}$$

where $X \sim N(\hat{\vec{x}}, P)$, $V \sim N(0, R)$, and $E\{XV\} = 0$. From equations (2.12) and (2.14) we have

$$Z \sim N(H\hat{\vec{x}}, HPH^T + R) .$$

However, the expression $HPH^T + R$ reflects the filter's *estimate* of the measurement residuals (innovations), not the actual residuals. This becomes clear when one examines the update expressions for P in the Kalman filter: P does not depend on the measurement residual. The effect of this is that the expression $HPH^T + R$ may indicate some small residual variance, when in fact at particular points in time the variance is relatively large. This is indeed exactly the case when one is simultaneously considering multiple models for a process—one of the models, or some combination of them, is "right" and actually has small residuals, while others are "wrong" and will suffer from large residuals. Thus when one is computing the likelihood of a residual for the purpose of comparing model performance, one must consider the liklihood of the *actual* measurement $\vec{z}$ at each time step, given the *expected* performance of each model.

The Likelihood of the Measurements Given a Particular Model

Given equations (2.13) and (2.15), we can use the following conditional probability density function as an indicator of the *likelihood* of a measurement $\vec{z}$ at step k:

$$f(\vec{z}|\mu) = \frac{1}{(2\pi|C_\mu|)^{n_\mu/2}} e^{-\frac{1}{2}(\vec{z} - H_\mu \hat{\vec{x}}_\mu^-)^T C_\mu^{-1} (\vec{z} - H_\mu \hat{\vec{x}}_\mu)} , \tag{5.2}$$

1. Recall that "white" means that the spectral density is constant and the autocorrelation is a delta function. In other words, the output of the noise source at any one instant in time is independent from that at any other instant in time.

where

$$C_\mu = H_\mu P_\mu^- H_\mu^T + R_\mu .$$

We have omitted the subscript k for clarity. Note again that the state vector $\hat{x}_\mu$ and error covariance matrix P_μ^- are the *a priori* (predicted) versions at step k, already computed at each filter prediction step using equation (4.9) and equation (4.10). In other words, the density is conditioned on the model μ and all of its associated *a priori* (predicted) parameters.

5.2.2 Fixed Multiple Models

We begin with the case where we believe that there is *one* correct model for the process, and that the model is fixed or does not change over time, however we don't know what that model is. Over time, as the filter reaches a steady state, we want to converge on a choice for the single most likely model. For this approach let us assume that the *correct* model M is one of r possible *known* fixed models,

$$M \in \{\mu_j\}_{j=1}^r .$$

The Probability of a Particular Model Given the Measurements

Given a new measurement $\hat{z}$ at time step k, and associated *a priori* state and covariance estimates from equation (4.9) and equation (4.10), we can use equation (5.2) to compute the recursive probability $p_j(k)$ that candidate model μ_j is the correct model at that time:

$$p_j(k) = \frac{f(\hat{z}|\mu_j) p_j(k-1)}{\sum_{h=1}^{r} f(\hat{z}|\mu_h) p_h(k-1)} . \tag{5.3}$$

One would initialize $p_j(0)$ with some *a priori* estimate of the probability that μ_j is the correct model. For example, one could consider all models equally likely to begin with, and set

$$p_j(0) = \frac{1}{r}, \; j = 1, 2, \ldots, r .$$

Note that $f(\hat{z}|\mu_j)$ and $p_j(0)$ are scalars, and at every time step k,

$$\sum_{j=1}^{r} p_j(k) = 1 .$$

The Final Model-Conditioned Estimate

The final combined or *model-conditioned* estimate of the state $\hat{x}_k$ and error covariance P_k are computed as a weighted combination of each candidate filter's *a posteriori* state and error covariance estimates. The weight for each candidate model is the model probability given by equation (5.3). The final model-conditioned state estimate is computed as

$$\widehat{x}_k = \sum_{j=1}^{r} p_j(k)\hat{x}_{k,\mu_j}, \text{ and} \tag{5.4}$$

the final model-conditioned error covariance as

$$P_k = \sum_{j=1}^{r} p_j(k)[P_{k,\mu_j} + \varepsilon_{\mu_j}\varepsilon_{\mu_j}^T], \tag{5.5}$$

where $\varepsilon_{\mu_j} = \widehat{x}_k - \hat{x}_{k,\mu_j}$.

The Algorithm

To begin with, one would instantiate r independent Kalman filters, one for each of the r candidate models. Each of these filters would then be run independently, in parallel, with the addition of the necessary individual density and final probability computations.

At each *time update* (see Figure 4.1) one would compute the normal *a priori* Kalman filter elements (see table 4.1), and then

a. using the conditional density function given in equation (5.2), compute the liklihood of the current (actual) measurement $\vec{z}$ for each candidate model μ_j;

b. using the previous probability $p_j(k-1)$ for each candidate model μ_j, use the recursive equation (5.3) to compute the probability $p_j(k)$ that each individual model μ_j is correct;

c. for each candidate model μ_j, compute the *a posteriori* (corrected) state estimate $\hat{x}_{k,\mu_j}$ and error covariance P_{k,μ_j} using equation (4.12) and equation (4.13);

d. given each candidate filter's *a posteriori (corrected)* state estimate $\hat{x}_{k,\mu_j}$, compute the final *model-conditioned* state estimate $\widehat{x}_k$ using equation (5.4); and

e. if desired, given each candidate filter's *a posteriori (corrected)* error covariance estimate P_{k,μ_j}, compute the final model-conditioned error covariance P_k using equation (5.5).

Convergence of the Mode Estimates

As described in (Bar-Shalom and Li 1993), the final mode-conditioned state estimate will converge to agree with one of the models, if one of the models is the correct one. In any case, it will converge to some constant mode represented by a fixed weighting of the individual multiple models.

If the actual mode is not constant, i.e. if the process can be switching or varying between different models, one can use various ad hoc methods to prevent convergence on a single mode. For example,

a. One can impose an artificial lower bound on the model probabilities,

b. impose a finite memory (sliding window) on the likelihood function, or

c. impose an exponential decay on the likelihood function.

A problem with using ad hoc means of varying the blending of fixed multiple models is that the error in the incorrect models (at any moment) can grow unbounded, i.e. the incorrect filters can get lost. Thus the filters might have to be re-initialized.

5.2.3 Dynamic Multiple Model Method

The multiple-model approach described in Section 5.2.2 is appropriate for systems where we believe there is *one* correct model for the process, and that the model is *fixed*. However there are situations where the choice from a set of candidate models varies continuously while the filter is in operation.´In such a case one cannot make a fixed *a priori* choice of filter parameters. Instead one could operate a set of candidate filters in parallel (similar to Section 5.2.2) and use a continuously varying model-conditioned combination of the candidate state and error covariance estimates.

The *dynamic multiple model approach* is virtually identical to the *fixed* approach outlined in Section 5.2.2, with the exception of the model probability given by equation (5.3). In the dynamic case we do not want the probabilities to converge to fixed values, but to remain free to change with each new measurement. Given a new measurement $\vec{z}$ at time

step k, and associated *a priori* state and covariance estimates from equation (4.9) and equation (4.10), one could compute the probability $p_j(k)$ that candidate model μ_j is the correct model at that time simply using

$$p_j(k) = \frac{f(\vec{z}|\mu_j)}{\sum_{h=1}^{r} f(\vec{z}|\mu_h)}. \tag{5.6}$$

The algorithm would remain the same as in Section 5.2.2, except that in step (b) one would use equation (5.6) instead of equation (5.3).

5.3 Hybrid or Multi-Sensor Fusion

Stochastic estimation tools such as the Kalman filter can be used to combine or *fuse* information from different "mediums or sensors for *hybrid systems*. The basic idea is to use the Kalman filter to weight the different mediums most heavily in the circumstances where they each perform best, thus providing more accurate and stable estimates than a system based on any one medium alone. In particular, the *indirect feedback* Kalman filter shown in Figure 5.1 (also called a *complementary* or *error-state* Kalman filter) is often used to combined the two mediums (Maybeck 1979). In such a configuration, the Kalman filter is used to estimate the *difference* between the current inertial and optical (or acoustic) outputs, i.e. it continually estimates the *error* in the inertial estimates by using the optical system as a second (redundant) reference. This error estimate is then used to correct the inertial estimates. The *tuning* of the Kalman filter parameters (see "Parameter Estimation or Tuning" on page 35) then adjusts the weight of the correction as a function of frequency. By slightly modifying the Kalman filter, adaptive velocity response can be incorporated also. This can be accomplished by adjusting (in real time) the expected optical measurement error as a function of the magnitude of velocity. The dashed line in Figure 5.1 also indicates the use of inertial estimates to help a image-based optical system to prevent tracking of moving scene objects (i.e. unrelated motion in the environment).

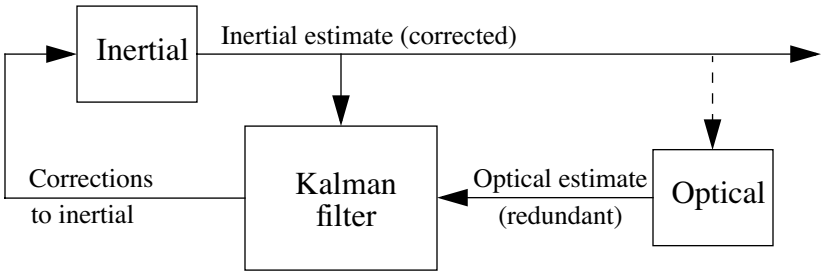

Figure 5.1: The Kalman filter used in an *indirect-feedback* configuration to optimally weight inertial and optical information.

In such a configuration, the Kalman filter uses a common *process model*, but a distinct *measurement model* for each of the inertial and optical subsystems.

5.4 Single-Constraint-at-a-Time (SCAAT)

A conventional approach to pose estimation is to collect a group of sensor measurements and then to attempt to simultaneously solve a system of equations that together completely constrain the solution. For example, the 1991 UNC-Chapel Hill wide-area opto-electronic tracking system (Wang 1990; Ward, Azuma et al. 1992) collected a group of diverse measurements for a variety of LEDs and sensors, and then used a method of simultaneous non-linear equations called *Collinearity* to estimate the pose of the head-worn sensor fixture (Azuma and Ward 1991). There was one equation for each measurement, expressing the constraint that a ray from the front principal point of the sensor lens to the LED, must be collinear with a ray from the rear principal point to the intersection with the sensor. Each estimate made use of typically 20 (or more) measurements that together over-constrained the solution.

This *multiple constraint* method had several drawbacks. First, it had a significantly lower estimate rate due to the need to collect multiple measurements per estimate. Second, the system of non-linear equations did not account for the fact that the sensor fixture continued to move throughout the collection of the sequence of measurements. Instead the method effectively assumes that the measurements were taken simultaneously. The violation of this *simultaneity assumption* could introduce significant error during even moderate motion. Finally, the method provided no means to identify or handle unusually noisy individual measurements. Thus, a single erroneous measurement could cause an estimate to jump away from an otherwise smooth track.

In contrast, there is typically nothing about typical solutions to the observer design problem in general (Section 3.2), or the Kalman filter in particular (see “Parameter Estimation or Tuning” on page 35), that dictates the ordering of measurement information. In 1996 we introduced a new approach to tracking with a Kalman filter, an approach that exploits this flexibility in measurement processing. The basic idea is to update the pose estimate as each new measurement is made, rather than waiting to form a complete collection of measurement. Because single measurements under-constrain the mathematical solution, we refer to the approach as *single-constraint-at-a-time* or SCAAT tracking (Welch 1996; Welch and Bishop 1997). The key is that the single measurements provide *some* information about the tracker state, and thus can be used to incrementally improve a previous estimate. We intentionally fuse each individual “insufficient” measurement immediately as it is obtained. With this approach we are able to generate estimates more frequently, with less latency, with improved accuracy, and we are able to estimate the LED positions on-line concurrently while tracking.

This approach is used in our laboratory-based HiBall Tracking System (Welch, Bishop et al. 1999; Welch, Bishop et al. 2001), the commercial version of the same system (3rdTech 2000), and the commercial systems manufactured by Intersense (Foxlin, Harrington et al. 1998; Intersense 2000). For more information see (Welch 1996; Welch and Bishop 1997), the former which is included at the end of this course pack.

A. Bibliography

3rdTech. (2000, July 15). *3rdTech™* , [HTML]. 3rdTech. Available: http://www.3rdtech.com/ [2000, July 19].

Azarbayejani, A., & Pentland, A. (1994). *Recursive estimation of motion, structure, and focal length* (Technical report 243). Cambridge, MA: Massachusetts Institute of Technology (MIT).

Azuma, R. T. (1995). *Predictive Tracking for Augmented Reality*. Unpublished Ph.D. Dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC USA.

Azuma, R. T., & Bishop, G. (1994). Improving Static and Dynamic Registration in an Optical See-Through HMD, *Computer Graphics* (SIGGRAPH 94 Conference Proceedings ed., pp. 197-204). Orlando, FL USA: ACM Press, Addison-Wesley.

Azuma, R. T., & Ward, M. (1991). *Space-Resection by Collinearity: Mathematics Behind the Optical Ceiling Head-Tracker* (Technical Report 91-048). Chapel Hill, NC USA: University of North Carolina at Chapel Hill.

Bar-Shalom, Y., & Li, X.-R. (1993). *Estimation and Tracking: Principles, Techniques, and Software*: Artec House, Inc.

Brown, R. G., & Hwang, P. Y. C. (1996). *Introduction to Random Signals and Applied Kalman Filtering: with MATLAB Exercises and Solutions* (Third ed.): Wiley & Sons, Inc.

Emura, S., & Tachi, S. (1994a). Compensation of time lag between actual and virtual spaces by multi-sensor integration, *1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* (pp. 363-469). Las Vegas, NV: Institute of Electrical and Electronics Engineers.

Emura, S., & Tachi, S. (1994b). *Sensor Fusion based Measurement of Human Head Motion*. Paper presented at the 3rd IEEE International Workshop on Robot and Human Communication (RO-MAN 94 NAGOYA), Nagoya University, Nagoya, Japan.

Foxlin, E., Harrington, M., & Pfeifer, G. (1998). Constellation™ : A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications. In M. F. Cohen (Ed.), *Computer Graphics* (SIGGRAPH 98 Conference Proceedings ed., pp. 371-378). Orlando, FL USA: ACM Press, Addison-Wesley.

Fuchs (Foxlin), E. (1993). *Inertial Head-Tracking*. Unpublished M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA USA.

Gelb, A. (1974). *Applied Optimal Estimation*. Cambridge, MA: MIT Press.

Grewal, M., S., & Andrews, A., P. (2001). *Kalman Filtering Theory and Practice Using MATLAB* (Second ed.). New York, NY USA: John Wiley & Sons, Inc.

Intersense. (2000). *Intersense IS-900*, [html]. Intersense. Available: http://www.isense.com/ [2000, April 27].

Jacobs, O. L. R. (1993). *Introduction to Control Theory* (Second ed.): Oxford University Press.

Julier, S., J., & Uhlmann, J., K. (1996). *A General Method for Approximating Nonlinear Transformations of Probability Distributions* (Technical Report). Oxford, UK: Robotics Research Group, Department of Engineering Science, University of Oxford.

Kailath, T., Sayed, A., H., & Hassibi, B. (2000). *Linear Estimation*. Upper Saddle River, NJ USA: Prentice Hall.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME—Journal of Basic Engineering, 82*(Series D), 35-45.

Kelly, D. G. (1994). *Introduction to Probability*: Macmillan Publishing Company.

Lewis, F. L. (1986). *Optimal Estimation with an Introductory to Stochastic Control Theory*: John Wiley & Sons, Inc.

Maybeck, P. S. (1979). *Stochastic models, estimation, and control* (Vol. 141).

Mazuryk, T., & Gervautz, M. (1995). Two-Step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments, *Proceedings of EUROGRAPHICS 95* (EUROGRAPHICS 95 ed., Vol. 14 (3), pp. 30-41).

Sorenson, H. W. (1970, July). Least-Squares estimation: from Gauss to Kalman. *IEEE Spectrum, 7,* 63-68.

Van Pabst, J. V. L., & Krekel, P. F. C. (1993). Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space, *7th International Conference on Image Analysis and Processing—* (pp. 174-182). Capitolo, Monopoli, Italy: World Scientific, Singapore.

Wang, J.-F. (1990). *A real-time optical 6D tracker for head-mounted display systems*. Unpublished Ph.D. Dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC USA.

Ward, M., Azuma, R. T., Bennett, R., Gottschalk, S., & Fuchs, H. (1992). A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems, *Symposium on Interactive 3D Graphics* (I3D 99 Conference Proceedings ed., pp. 43-52). Cambridge, MA USA: ACM Press, Addison-Wesley.

Welch, G. (1996). *SCAAT: Incremental Tracking with Incomplete Information*. Unpublished Ph.D. Dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

Welch, G., & Bishop, G. (1997). SCAAT: Incremental Tracking with Incomplete Information. In T. Whitted (Ed.), *Computer Graphics* (SIGGRAPH 97 Conference Proceedings ed., pp. 333-344). Los Angeles, CA, USA (August 3 - 8): ACM Press, Addison-Wesley.

Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., & Colucci, D. n. (1999). The HiBall Tracker: High-Performance Wide-Area Tracking for Virtual and Augmented Environments, *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 1-11). University College London, London, United Kingdom (December 20 - 23): ACM SIGGRAPH, Addison-Wesley.

Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., & Colucci, D. n. (2001). High-Performance Wide-Area Optical Tracking—The HiBall Tracking System. *Presence: Teleoperators and Virtual Environments, 10*(1).

B. Related Papers

This appendix includes a sample of some relevant papers that we have permission to reproduce.

Stochastic models, estimation, and control

VOLUME 1

PETER S. MAYBECK

DEPARTMENT OF ELECTRICAL ENGINEERING
AIR FORCE INSTITUTE OF TECHNOLOGY
WRIGHT-PATTERSON AIR FORCE BASE
OHIO

ACADEMIC PRESS New York San Francisco London 1979
A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1 7DX

Library of Congress Cataloging in Publication Data

Maybeck, Peter S
Stochastic models, estimation and control.

(Mathematics in science and engineering ; v.)
Includes bibliographies.
1. System analysis 2. Control theory. 3. Estimation theory. I. Title. II. Series.
QA402.M37 519.2 78-8836
ISBN 0-12-480701-1 (v. 1)

PRINTED IN THE UNITED STATES OF AMERICA

79 80 81 82 9 8 7 6 5 4 3 2 1

To Beverly

CHAPTER 1
Introduction

1.1 WHY STOCHASTIC MODELS, ESTIMATION, AND CONTROL?

When considering system analysis or controller design, the engineer has at his disposal a wealth of knowledge derived from deterministic system and control theories. One would then naturally ask, why do we have to go beyond these results and propose stochastic system models, with ensuing concepts of estimation and control based upon these stochastic models? To answer this question, let us examine what the deterministic theories provide and determine where the shortcomings might be.

Given a physical system, whether it be an aircraft, a chemical process, or the national economy, an engineer first attempts to develop a mathematical model that adequately represents some aspects of the behavior of that system. Through physical insights, fundamental "laws," and empirical testing, he tries to establish the interrelationships among certain variables of interest, inputs to the system, and outputs from the system.

With such a mathematical model and the tools provided by system and control theories, he is able to investigate the system structure and modes of response. If desired, he can design compensators that alter these characteristics and controllers that provide appropriate inputs to generate desired system responses.

In order to observe the actual system behavior, measurement devices are constructed to output data signals proportional to certain variables of interest. These output signals and the known inputs to the system are the only information that is directly discernible about the system behavior. Moreover, if a feedback controller is being designed, the measurement device outputs are the only signals directly available for inputs to the controller.

There are three basic reasons why deterministic system and control theories do not provide a totally sufficient means of performing this analysis and

design. First of all, *no mathematical system model is perfect*. Any such model depicts only those characteristics of direct interest to the engineer's purpose. For instance, although an endless number of bending modes would be required to depict vehicle bending precisely, only a finite number of modes would be included in a useful model. The objective of the model is to represent the dominant or critical modes of system response, so many effects are knowingly left unmodeled. In fact, models used for generating online data processors or controllers must be pared to only the basic essentials in order to generate a computationally feasible algorithm.

Even effects which are modeled are necessarily *approximated* by a mathematical model. The "laws" of Newtonian physics are adequate approximations to what is actually observed, partially due to our being unaccustomed to speeds near that of light. It is often the case that such "laws" provide adequate system *structures*, but various *parameters* within that structure are not determined absolutely. Thus, there are many sources of uncertainty in any mathematical model of a system.

A second shortcoming of deterministic models is that dynamic systems are driven not only by our own control inputs, but also by *disturbances which we can neither control nor model deterministically*. If a pilot tries to command a certain angular orientation of his aircraft, the actual response will differ from his expectation due to wind buffeting, imprecision of control surface actuator responses, and even his inability to generate exactly the desired response from his own arms and hands on the control stick.

A final shortcoming is that sensors *do not provide perfect and complete data* about a system. First, they generally do not provide all the information we would like to know: either a device cannot be devised to generate a measurement of a desired variable or the cost (volume, weight, monetary, etc.) of including such a measurement is prohibitive. In other situations, a number of different devices yield functionally related signals, and one must then ask how to generate a best estimate of the variables of interest based on partially redundant data. Sensors do not provide exact readings of desired quantities, but introduce their own system dynamics and distortions as well. Furthermore, these devices are also always noise corrupted.

As can be seen from the preceding discussion, to assume perfect knowledge of all quantities necessary to describe a system completely and/or to assume perfect control over the system is a naive, and often inadequate, approach. This motivates us to ask the following four questions:

(1) How do you develop system models that account for these uncertainties in a direct and proper, yet practical, fashion?

(2) Equipped with such models and incomplete, noise-corrupted data from available sensors, how do you optimally estimate the quantities of interest to you?

(3) In the face of uncertain system descriptions, incomplete and noise-corrupted data, and disturbances beyond your control, how do you optimally control a system to perform in a desirable manner?

(4) How do you evaluate the performance capabilities of such estimation and control systems, both before and after they are actually built? This book has been organized specifically to answer these questions in a meaningful and useful manner.

1.2 OVERVIEW OF THE TEXT

Chapters 2-4 are devoted to the stochastic modeling problem. First Chapter 2 reviews the pertinent aspects of deterministic system models, to be exploited and generalized subsequently. Probability theory provides the basis of all of our stochastic models, and Chapter 3 develops both the general concepts and the natural result of static system models. In order to incorporate dynamics into the model, Chapter 4 investigates stochastic processes, concluding with practical linear dynamic system models. The basic form is a linear system driven by white Gaussian noise, from which are available linear measurements which are similarly corrupted by white Gaussian noise. This structure is justified extensively, and means of describing a large class of problems in this context are delineated.

Linear estimation is the subject of the remaining chapters. Optimal filtering for cases in which a linear system model adequately describes the problem dynamics is studied in Chapter 5. With this background, Chapter 6 describes the design and performance analysis of practical online Kalman filters. Square root filters have emerged as a means of solving some numerical precision difficulties encountered when optimal filters are implemented on restricted word-length online computers, and these are detailed in Chapter 7.

Volume 1 is a complete text in and of itself. Nevertheless, Volume 2 will extend the concepts of linear estimation to smoothing, compensation of model inadequacies, system identification, and adaptive filtering. Nonlinear stochastic system models and estimators based upon them will then be fully developed. Finally, the theory and practical design of stochastic controllers will be described.

1.3 THE KALMAN FILTER: AN INTRODUCTION TO CONCEPTS

Before we delve into the details of the text, it would be useful to see where we are going on a conceptual basis. Therefore, the rest of this chapter will provide an overview of the optimal linear estimator, the Kalman filter. This will be conducted at a very elementary level but will provide insights into the

underlying concepts. As we progress through this overview, contemplate the ideas being presented: try to conceive of graphic *images* to portray the concepts involved (such as time propagation of density functions), and to generate a *logical structure* for the component pieces that are brought together to solve the estimation problem. If this basic conceptual framework makes sense to you, then you will better understand the need for the details to be developed later in the text. Should the idea of where we are going ever become blurred by the development of detail, refer back to this overview to regain sight of the overall objectives.

First one must ask, what is a Kalman filter? A Kalman filter is simply an *optimal recursive data processing algorithm*. There are many ways of defining *optimal*, dependent upon the criteria chosen to evaluate performance. It will be shown that, under the assumptions to be made in the next section, the Kalman filter is optimal with respect to virtually any criterion that makes sense. One aspect of this optimality is that the Kalman filter incorporates all information that can be provided to it. It processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with use of (1) knowledge of the system and measurement device dynamics, (2) the statistical description of the system noises, measurement errors, and uncertainty in the dynamics models, and (3) any available information about initial conditions of the variables of interest. For example, to determine the velocity of an aircraft, one could use a Doppler radar, or the velocity indications of an inertial navigation system, or the pitot and static pressure and relative wind information in the air data system. Rather than ignore any of these outputs, a Kalman filter could be built to combine all of this data and knowledge of the various systems' dynamics to generate an overall best estimate of velocity.

The word *recursive* in the previous description means that, unlike certain data processing concepts, the Kalman filter does not require all previous data to be kept in storage and reprocessed every time a new measurement is taken. This will be of vital importance to the practicality of filter implementation.

The "filter" is actually a *data processing algorithm*. Despite the typical connotation of a filter as a "black box" containing electrical networks, the fact is that in most practical applications, the "filter" is just a computer program in a central processor. As such, it inherently incorporates discrete-time measurement samples rather than continuous time inputs.

Figure 1.1 depicts a typical situation in which a Kalman filter could be used advantageously. A system of some sort is driven by some known controls, and measuring devices provide the value of certain pertinent quantities. Knowledge of these system inputs and outputs is all that is explicitly available from the physical system for estimation purposes.

The *need* for a filter now becomes apparent. Often the variables of interest, some finite number of quantities to describe the "state" of the system, cannot

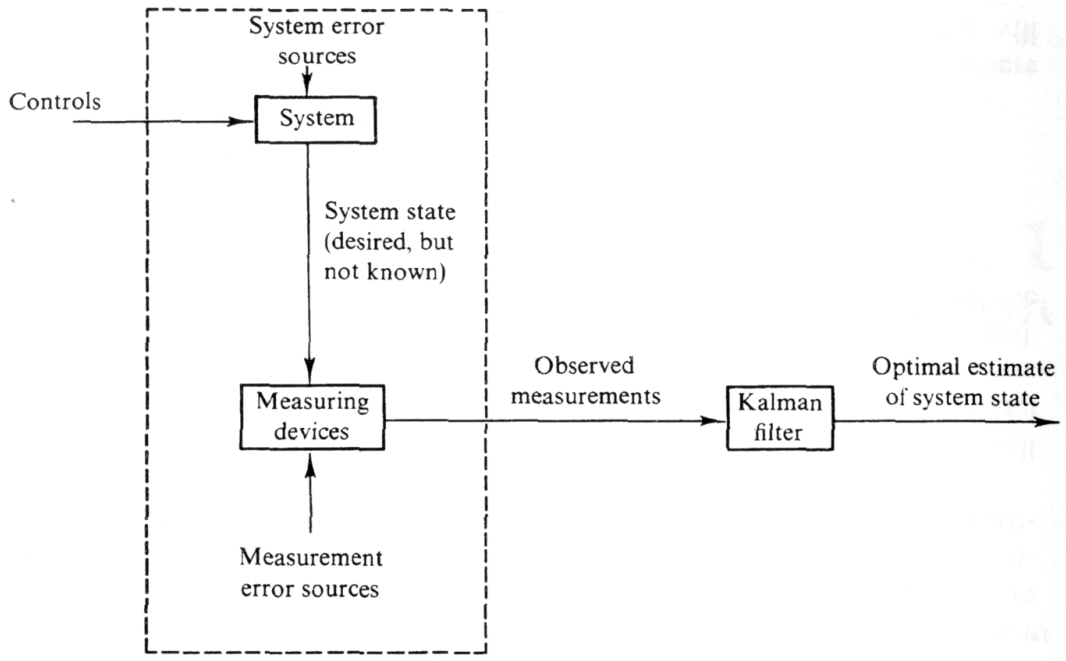

FIG. 1. 1 Typical Kalman filter application

be measured directly, and some means of inferring these values from the available data must be generated. For instance, an air data system directly provides static and pitot pressures, from which velocity must be inferred. This inference is complicated by the facts that the system is typically driven by inputs other than our own known controls and that the relationships among the various "state" variables and measured outputs are known only with some degree of uncertainty. Furthermore, any measurement will be corrupted to some degree by noise, biases, and device inaccuracies, and so a means of extracting valuable information from a noisy signal must be provided as well. There may also be a number of different measuring devices, each with its own particular dynamics and error characteristics, that provide some information about a particular variable, and it would be desirable to combine their outputs in a systematic and optimal manner. A Kalman filter combines all available measurement data, plus prior knowledge about the system and measuring devices, to produce an estimate of the desired variables in such a manner that the error is minimized statistically. In other words, if we were to run a number of candidate filters many times for the same application, then the average results of the Kalman filter would be better than the average results of any other.

Conceptually, what any type of filter tries to do is obtain an "optimal" estimate of desired quantities from data provided by a noisy environment, "optimal" meaning that it minimizes errors in some respect. There are many means of accomplishing this objective. If we adopt a Bayesian viewpoint, then we want the filter to propagate the *conditional probability density* of the desired

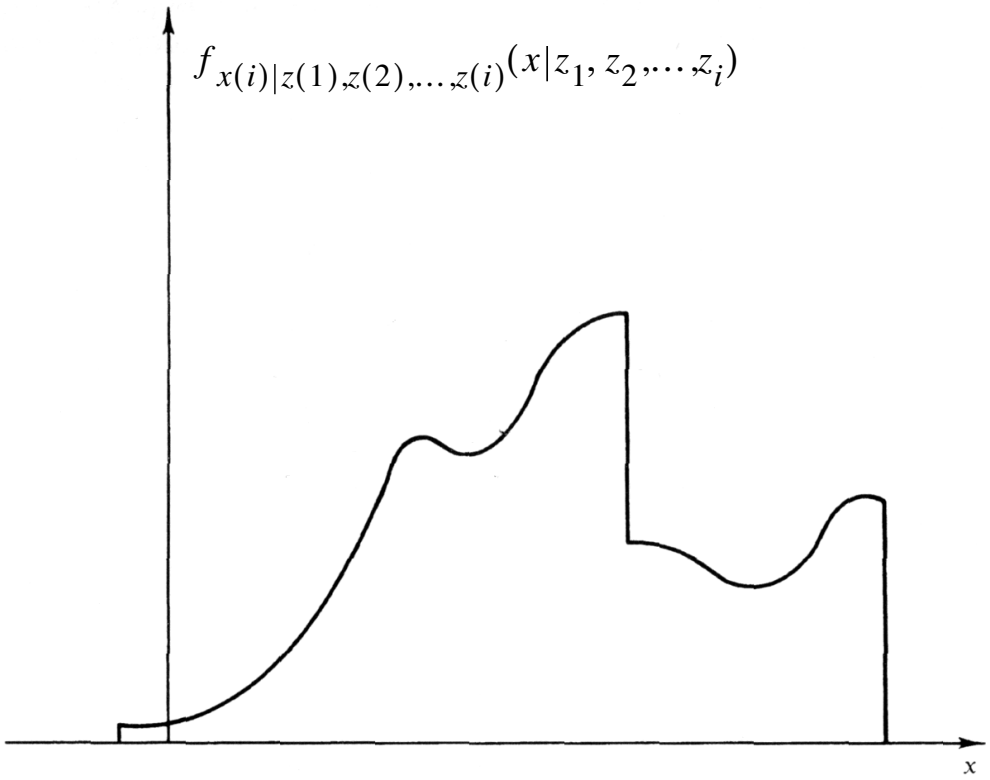

FIG. 1. 2 Conditional probability density.

quantities, conditioned on knowledge of the actual data coming from the measuring devices. To understand this concept, consider Fig. 1.2, a portrayal of a conditional probability density of the value of a scalar quantity x at time instant i ($x(i)$), conditioned on knowledge that the vector measurement $z(1)$ at time instant 1 took on the value z_1 ($z(1) = z_1$) and similarly for instants 2 through i, plotted as a function of possible $x(i)$ values. This is denoted as $f_{x(i)|z(1),z(2),\ldots,z(i)}(x|z_1, z_2,\ldots,z_i)$. For example, let $x(i)$ be the one-dimensional position of a vehicle at time instant 1, and let $z(j)$ be a two-dimensional vector describing the measurements of position at time j by two separate radars. Such a conditional probability density contains all the available information about $x(i)$: it indicates, for the given value of all measurements taken up through time instant i, what the probability would be of $x(i)$ assuming any particular value or range of values.

It is termed a "conditional" probability density because its shape and location on the x axis is dependent upon the values of the measurements taken. Its shape conveys the amount of certainty you have in the knowledge of the value of x. If the density plot is a narrow peak, then most of the probability "weight" is concentrated in a narrow band of x values. On the other hand, if the plot has a gradual shape, the probability "weight" is spread over a wider range of x, indicating that you are less sure of its value.

Once such a conditional probability density function is propagated, the "optimal" estimate can be defined. Possible choices would include

(1) the *mean*—the "center of probability mass" estimate;
(2) the *mode*—the value of x that has the highest probability, locating the peak of the density; and
(3) the *median*—the value of x such that half of the probability weight lies to the left and half to the right of it.

A Kalman filter performs this conditional probability density propagation for problems in which the system can be described through a *linear* model and in which system and measurement noises are *white* and *Gaussian* (to be explained shortly). Under these conditions, the mean, mode, median, and virtually any reasonable choice for an "optimal" estimate all coincide, so there is in fact a unique "best" estimate of the value of x. Under these three restrictions, the Kalman filter can be shown to be the best filter of any conceivable form. Some of the restrictions can be relaxed, yielding a qualified optimal filter. For instance, if the Gaussian assumption is removed, the Kalman filter can be shown to be the best (minimum error variance) filter out of the class of linear unbiased filters. However, these three assumptions can be justified for many potential applications, as seen in the following section.

1.4 BASIC ASSUMPTIONS

At this point it is useful to look at the three basic assumptions in the Kalman filter formulation. On first inspection, they may appear to be overly restrictive and unrealistic. To allay any misgivings of this sort, this section will briefly discuss the physical implications of these assumptions.

A linear system model is justifiable for a number of reasons. Often such a model is adequate for the purpose at hand, and when nonlinearities do exist, the typical engineering approach is to linearize about some nominal point or trajectory, achieving a perturbation model or error model. Linear systems are desirable in that they are more easily manipulated with engineering tools, and linear system (or differential equation) theory is much more complete and practical than nonlinear. The fact is that there are means of extending the Kalman filter concept to some nonlinear applications or developing nonlinear filters directly, but these are considered only if linear models prove inadequate.

"Whiteness" implies that the noise value is not correlated in time. Stated more simply, if you know what the value of the noise is now, this knowledge does you no good in predicting what its value will be at any other time. Whiteness also implies that the noise has equal power at all frequencies. Since this results in a noise with infinite power, a white noise obviously cannot really exist. One might then ask, why even consider such a concept if it does not

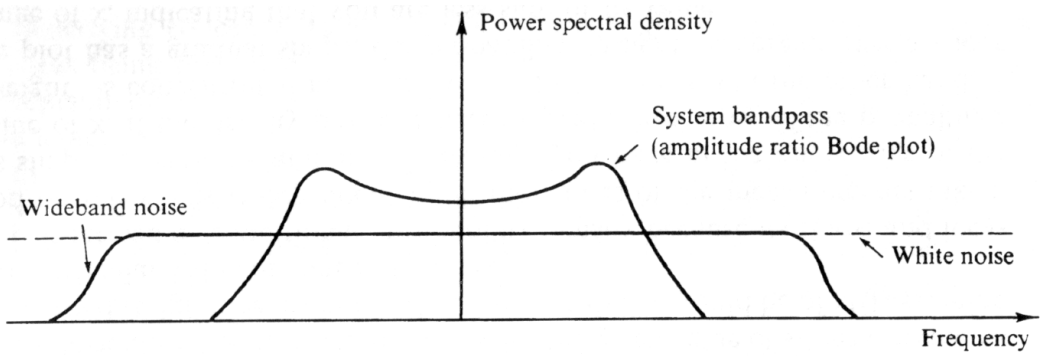

FIG. 1. 3 Power spectral density bandwidths.

exist in real life? The answer is twofold. First, any physical system of interest has a certain frequency "bandpass"—a frequency range of inputs to which it can respond. Above this range, the input either has no effect, or the system so severely attenuates the effect that it essentially does not exist. In Fig. 1.3, a typical system bandpass curve is drawn on a plot of "power spectral density" (interpreted as the amount of power content at a certain frequency) versus frequency. Typically a system will be driven by wideband noise—one having power at frequencies above the system bandpass, and essentially constant power at all frequencies within the system bandpass—as shown in the figure. On this same plot, a white noise would merely extend this constant power level out across all frequencies. Now, within the bandpass of the system of interest, the fictitious white noise looks identical to the real wideband noise. So what has been gained? That is the second part of the answer to why a white noise model is used. It turns out that the mathematics involved in the filter are vastly simplified (in fact, made tractable) by replacing the real wideband noise with a white noise which, from the system's "point of view," is identical. Therefore, the white noise model is used.

One might argue that there are cases in which the noise power level is not constant over all frequencies within the system bandpass, or in which the noise is in fact time correlated. For such instances, a white noise put through a small linear system can duplicate virtually any form of time-correlated noise. This small system, called a "shaping filter," is then added to the original system, to achieve an overall linear system driven by white noise once again.

Whereas whiteness pertains to time or frequency relationships of a noise, Gaussianness has to do with its amplitude. Thus, at any single point in time, the probability density of a Gaussian noise amplitude takes on the shape of a normal bell-shaped curve. This assumption can be justified physically by the fact that a system or measurement noise is typically caused by a number of small sources. It can be shown mathematically that when a number of independent random variables are added together, the summed effect can be described very closely by a Gaussian probability density, regardless of the shape of the individual densities.

There is also a practical justification for using Gaussian densities. Similar to whiteness, it makes the mathematics tractable. But more than that, typically an engineer will know, at best, the first and second order statistics (mean and variance or standard deviation) of a noise process. In the absence of any higher order statistics, there is no better form to assume than the Gaussian density. The first and second order statistics completely determine a Gaussian density, unlike most densities which require an endless number of orders of statistics to specify their shape entirely. Thus, the Kalman filter, which propagates the first and second order statistics, includes *all* information contained in the conditional probability density, rather than only some of it, as would be the case with a different form of density.

The particular assumptions that are made are dictated by the objectives of, and the underlying motivation for, the model being developed. If our objective were merely to build good descriptive models, we would not confine our attention to linear system models driven by white Gaussian noise. Rather, we would seek the model, of whatever form, that best fits the data generated by the "real world." It is our desire to build estimators and controllers based upon our system models that drives us to these assumptions: other assumptions generally do not yield tractable estimation or control problem formulations. Fortunately, the class of models that yields tractable mathematics also provides adequate representations for many applications of interest. Later, the model structure will be extended somewhat to enlarge the range of applicability, but the requirement of model usefulness in subsequent estimator or controller design will again be a dominant influence on the manner in which the extensions are made.

1.5 A SIMPLE EXAMPLE

To see how a Kalman filter works, a simple example will now be developed. Any example of a single measuring device providing data on a single variable would suffice, but the determination of a position is chosen because the probability of one's exact location is a familiar concept that easily allows dynamics to be incorporated into the problem.

Suppose that you are lost at sea during the night and have no idea at all of your location. So you take a star sighting to establish your position (for the sake of simplicity, consider a one-dimensional location). At some time t_1 you determine your location to be z_1. However, because of inherent measuring device inaccuracies, human error, and the like, the result of your measurement is somewhat uncertain. Say you decide that the precision is such that the standard deviation (one-sigma value) involved is σ_{z_1} (or equivalently, the variance, or second order statistic, is $\sigma_{z_1}^2$,). Thus, you can establish the conditional probability of $x(t_1)$, your position at time t_1, conditioned on the observed value of

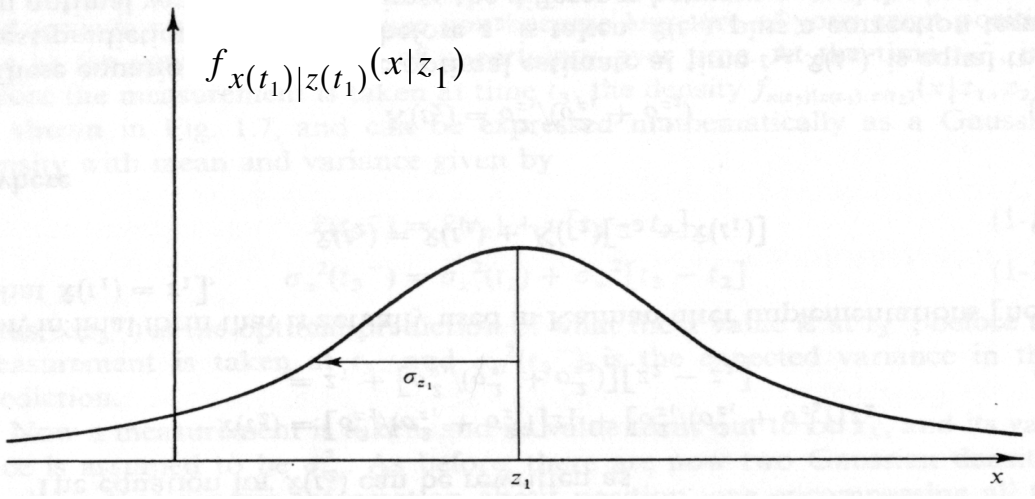

FIG. 1. 4 Conditional density of position based on measured value z_1.

the measurement being z_1, as depicted in Fig. 1.4. This is a plot of $f_{x(t_1)|z(t_1)}(x|z_1)$ as a function of the location x: it tells you the probability of being in any one location, based upon the measurement you took. Note that σ_{z_1} is a direct measure of the uncertainty: the larger σ_{z_1} is, the broader the probability peak is, spreading the probability "weight" over a larger range of x values. For a Gaussian density, 68.3% of the probability "weight" is contained within the band σ units to each side of the mean, the shaded portion in Fig. 1.4.

Based on this conditional probability density, the best estimate of your position is

$$\hat{x}(t_1) = z_1 \tag{1-1}$$

and the variance of the error in the estimate is

$$\sigma_x^2(t_1) = \sigma_{z_1}^2 \tag{1-2}$$

Note that $\hat{x}$ is both the mode (peak) and the median (value with $1/2$ of the probability weight to each side), as well as the mean (center of mass).

Now say a trained navigator friend takes an independent fix right after you do, at time $t_2 \cong t_1$ (so that the true position has not changed at all), and obtains a measurement z_2 with a variance σ_{z_2}. Because he has a higher skill, assume the variance in his measurement to be somewhat smaller than in yours. Figure 1.5 presents the conditional density of your position at time t_2, based only on the measured value z_2. Note the narrower peak due to smaller variance, indicating that you are rather certain of your position based on his measurement.

At this point, you have two measurements available for estimating your position. The question is, how do you combine these data? It will be shown subsequently that, based on the assumptions made, the conditional density of

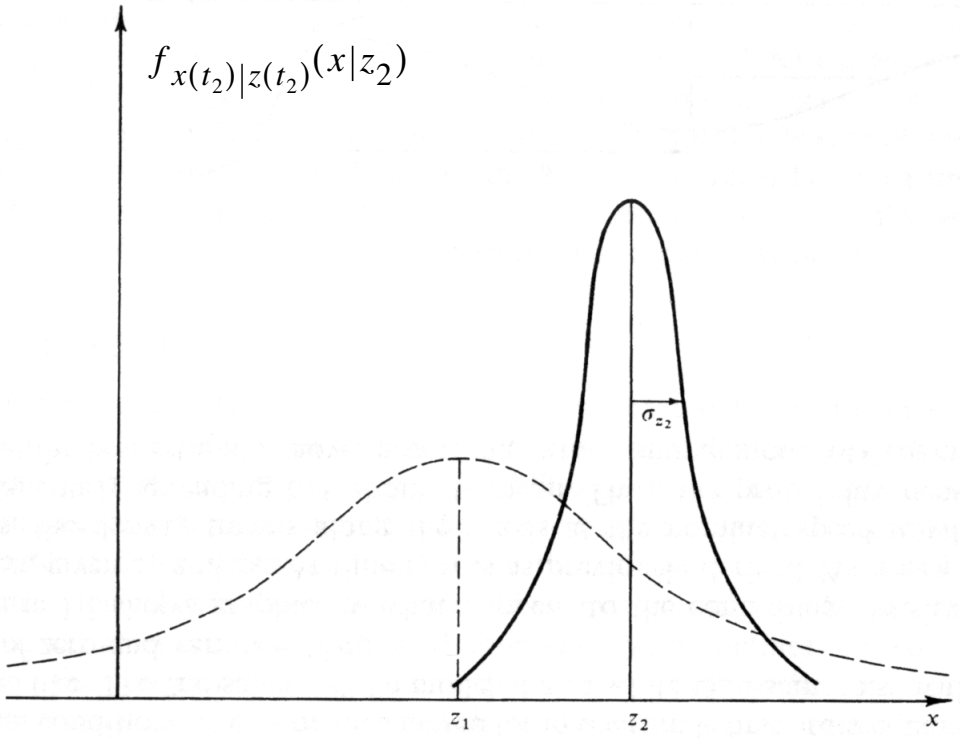

FIG. 1. 5 Conditional density of position based on measurement z_2 alone.

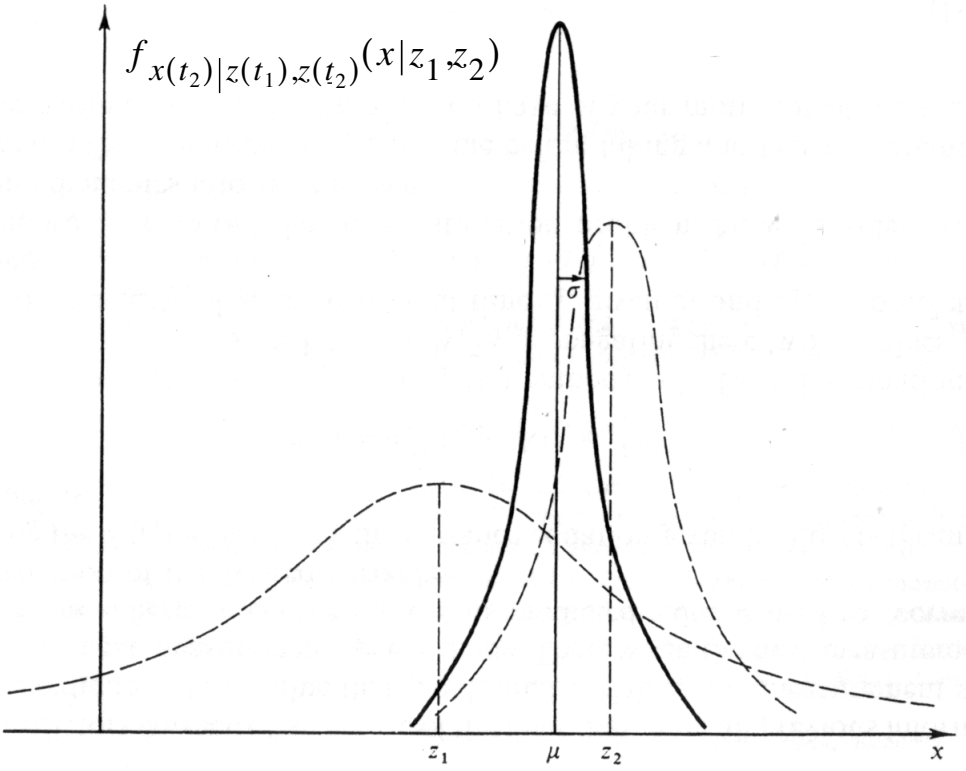

FIG. 1. 6 Conditional density of position based on data z_1 and z_2.

your position at time $t_2 \cong t_1$, $x(t_2)$, given both z_1 and z_2, is a Gaussian density with mean μ and variance σ^2 as indicated in Fig. 1.6, with

$$\mu = [\sigma_{z_2}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_2 \tag{1-3}$$

$$1/\sigma^2 = (1/\sigma_{z_1}^2) + (1/\sigma_{z_2}^2) \tag{1-4}$$

Note that, from (l-4), σ is less than either σ_{z_1} or σ_{z_2}, which is to say that the uncertainty in your estimate of position has been decreased by combining the two pieces of information.

Given this density, the best estimate is

$$\hat{x}(t_2) = \mu \tag{1-5}$$

with an associated error variance σ^2. It is the mode and the mean (or, since it is the mean of a conditional density, it is also termed the conditional mean). Furthermore, it is also the maximum likelihood estimate, the weighted least squares estimate, and the linear estimate whose variance is less than that of any other linear unbiased estimate. In other words, it is the "best" you can do according to just about any reasonable criterion.

After some study, the form of μ given in Eq. (1-3) makes good sense. If σ_{z_1} were equal to σ_{z_2}, which is to say you think the measurements are of equal precision, the equation says the optimal estimate of position is simply the average of the two measurements, as would be expected. On the other hand, if σ_{z_1} were larger than σ_{z_2}, which is to say that the uncertainty involved in the measurement z_1 is greater than that of z_2, then the equation dictates "weighting" z_2 more heavily than z_1. Finally, the variance of the estimate is less than σ_{z_1}, even if σ_{z_2} is very large: even poor quality data provide some information, and should thus increase the precision of the filter output.

The equation for $\hat{x}(t_2)$ can be rewritten as

$$\begin{aligned}\hat{x}(t_2) &= [\sigma_{z_2}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_2 \\ &= z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)][z_2 - z_1]\end{aligned} \tag{1-6}$$

or, in final form that is actually used in Kalman filter implementations [noting that $\hat{x}(t_1) = z_1$]

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)] \tag{1-7}$$

where

$$K(t_2) = \sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2) \tag{1-8}$$

These equations say that the optimal estimate at time t_2, $\hat{x}(t_2)$, is equal to the best prediction of its value before z_2 is taken, $\hat{x}(t_1)$, plus a correction term of an optimal weighting value times the difference between z_2 and the best prediction of its value before it is actually taken, $\hat{x}(t_1)$. It is worthwhile to understand this "predictor-corrector" structure of the filter. Based on all previous

information, a prediction of the value that the desired variables and measurement will have at the next measurement time is made. Then, when the next measurement is taken, the difference between it and its predicted value is used to "correct" the prediction of the desired variables.

Using the $K(t_2)$ in Eq. (l-8), the variance equation given by Eq. (1-4) can be rewritten as

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - K(t_2)\sigma_x^2(t_1) \tag{1-9}$$

Note that the values of $\hat{x}(t_2)$ and $\sigma_x^2(t_2)$ embody all of the information in $f_{x(t_2)|z(t_1),z(t_2)}(x|z_1,z_2)$. Stated differently, by propagating these two variables, the conditional density of your position at time t_2, given z_1 and z_2, is completely specified.

Thus we have solved the static estimation problem. Now consider incorporating dynamics into the problem.

Suppose that you travel for some time before taking another measurement. Further assume that the best mode1 you have of your motion is of the simple form

$$dx/dt = u + w \tag{1-10}$$

where u is a nominal velocity and w is a noise term used to represent the uncertainty in your knowledge of the actual velocity due to disturbances, off-nominal conditions, effects not accounted for in the simple first order equation, and the like. The "noise" w will be modeled as a white Gaussian noise with a mean of zero and variance of σ_w^2.

Figure 1.7 shows graphically what happens to the-conditional density of position, given z_1 and z_2. At time t_2 it is as previously derived. As time progresses, the density travels along the x axis at the nominal speed u, while simultaneously spreading out about its mean. Thus, the probability density starts at the best estimate, moves according to the nominal model of dynamics,

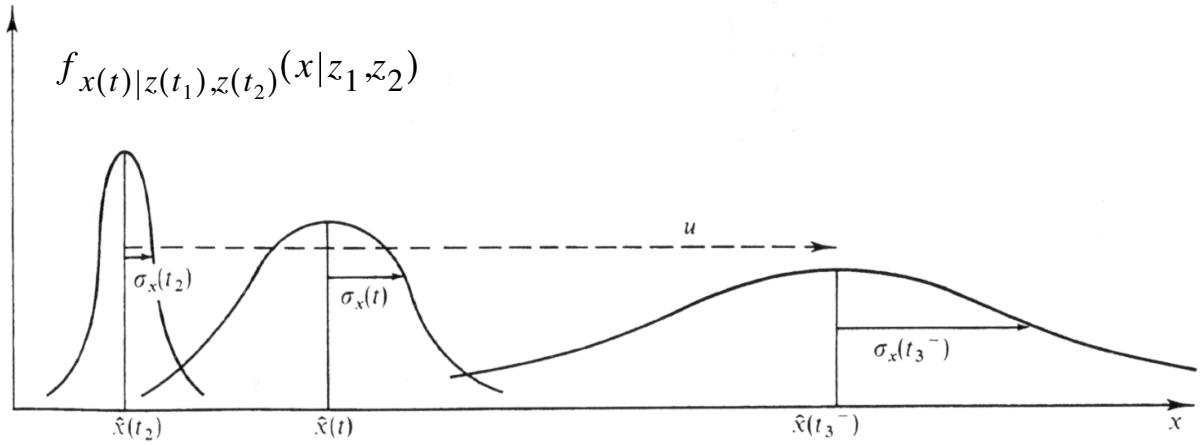

FIG. 1. 7 Propagation of conditional probability density.

and spreads out in time because you become less sure of your exact position due to the constant addition of uncertainty over time. At the time t_3^-, just before the measurement is taken at time t_3, the density $f_{x(t_3)|z(t_1),z(t_2)}(x|z_1,z_2)$ is as shown in Fig. 1.7, and can be expressed mathematically as a Gaussian density with mean and variance given by

$$\hat{x}(t_3^-) = \hat{x}(t_2) + u[t_3 - t_2] \tag{1-11}$$

$$\sigma_x^2(t_3^-) = \sigma_x^2(t_2) + \sigma_w^2[t_3 - t_2] \tag{1-12}$$

Thus, $\hat{x}(t_3^-)$ is the optimal prediction of what the x value is at t_3^-, before the measurement is taken at t_3, and $\sigma_x^2(t_3^-)$ is the expected variance in that prediction.

Now a measurement is taken, and its value turns out to be z_3, and its variance is assumed to be $\sigma_{z_3}^2$. As before, there are now two Gaussian densities available that contain information about position, one encompassing all the information available before the measurement, and the other being the information provided by the measurement itself. By the same process as before, the density with mean $\hat{x}(t_3^-)$ and variance $\sigma_x^2(t_3^-)$ is combined with the density with mean z_3 and variance $\sigma_{z_3}^2$ to yield a Gaussian density with mean

$$\hat{x}(t_3) = \hat{x}(t_3^-) + K(t_3)[z_3 - \hat{x}(t_3^-)] \tag{1-13}$$

and variance

$$\sigma_x^2(t_3) = \sigma_x^2(t_3^-) - K(t_3)\sigma_x^2(t_3^-) \tag{1-14}$$

where the gain $K(t_3)$ is given by

$$K(t_3) = \sigma_x^2(t_3^-)/[\sigma_x^2(t_3^-) + \sigma_{z_3}^2] \tag{1-15}$$

The optimal estimate, $\hat{x}(t_3)$, satisfies the same form of equation as seen previously in (1-7). The best prediction of its value before z_3 is taken is corrected by an optimal weighting value times the difference between z_3 and the prediction of its value. Similarly, the variance and gain equations are of the same form as (1-8) and (1-9).

Observe the form of the equation for $K(t_3)$. If σ_{z_3}, the measurement noise variance, is large, then $K(t_3)$ is small; this simply says that you would tend to put little confidence in a very noisy measurement and so would weight it lightly. In the limit as $\sigma_{z_3}^2 \to \infty$, $K(t_3)$ becomes zero, and $\hat{x}(t_3)$ equals $\hat{x}(t_3^-)$; an infinitely noisy measurement is totally ignored. If the dynamic system noise variance σ_w^2 is large, then $\sigma_x^2(t_3^-)$ will be large [see Eq. (1-12)] and so will $K(t_3)$; in this case, you are not very certain of the output of the system model within the filter structure and therefore would weight the measurement heavily. Note that in the limit as $\sigma_w^2 \to \infty$, $\sigma_x^2(t_3^-) \to \infty$ and $K(t_3) \to 1$, so Eq. (1-13) yields

$$\hat{x}(t_3) = \hat{x}(t_3^-) + 1 \cdot [z_3 - \hat{x}(t_3^-)] = z_3 \tag{1-16}$$

Thus in the limit of absolutely no confidence in the system model output, the optimal policy is to ignore the output and use the new measurement as the optimal estimate. Finally, if $\sigma_x^2(t_3^-)$ should ever become zero, then so does $K(t_3)$; this is sensible since if $\sigma_x^2(t_3^-) = 0$, you are absolutely sure of your estimate before z_3 becomes available and therefore can disregard the measurement.

Although we have not as yet derived these results mathematically, we have been able to demonstrate the reasonableness of the filter structure.

1.6 A PREVIEW

Extending Eqs. (1-11) and (1-12) to the vector case and allowing time varying parameters in the system and noise descriptions yields the general Kalman filter algorithm for propagating the conditional density and optimal estimate from one measurement sample time to the next. Similarly, the Kalman filter update at a measurement time is just the extension of Eqs. (l-13)-(1-15). Further logical extensions would include estimation with data beyond the time when variables are to be estimated, estimation with nonlinear system models rather than linear, control of systems described through stochastic models, and both estimation and control when the noise and system parameters are not known with absolute certainty. The sequel provides a thorough investigation of those topics, developing both the theoretical mathematical aspects and practical engineering insights necessary to resolve the problem formulations and solutions fully.

GENERAL REFERENCES

1. Aoki, M., *Optimization of Stochastic Systems—Topics in Discrete-Time Systems*. Academic Press, New York, 1967.
2. Aström. K. J., *Introduction to Stochastic Control Theory*. Academic Press, New York, 1970.
3. Bryson, A. E. Jr., and Ho. Y., *Applied Optimal Control*. Blaisdell, Wahham, Massachusetts, 1969.
4. Bucy, R. S., and Joseph, P. D., *Filtering for Stochastic Processes with Applications to Guidance*. Wiley, New York, 1968.
5. Deutsch, R., *Estimation Theory*. Prentice-Hall, Englewood Cliffs, New Jersey, 1965.
6. Deyst, J. J., "Estimation and Control of Stochastic Processes," unpublished course notes. M.I.T. Dept. of Aeronautics and Astronautics, Cambridge, Massachusetts, 1970.
7. Gelb, A. (ed.), *Applied Optimal Estimation*. M.I.T. Press, Cambridge, Massachusetts, 1974.
8. Jazwinski, A. H., *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.
9. Kwakernaak. H., and Sivan. R., *Linear Optimal Control Systems*. Wiley, New York, 1972.
10. Lee, R. C. K., *Optimal Estimation, Identification and Control*. M. I. T. Press, Cambridge, Massachusetts, 1964.
11. Liebelt, P. B., *An Introduction to Optimal Estimation*. Addison-Wesley, Reading, Massachusetts, 1967.
12. Maybeck, P. S., "The Kalman Filter—An Introduction for Potential Users," TM-72-3, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, June 1972.

13. Maybeck, P. S., "Applied Optimal Estimation—Kalman Filter Design and Implementation," notes for a continuing education course offered by the Air Force Institute of Technology, Wright-Patterson AFB, Ohio, semiannually since December 1974.
14. Meditch, J. S., *Stochastic Optimal Linear Estimation and Control*. McGraw-Hill, New York, 1969.
15. McGarty, T. P., *Stochastic Systems and State Estimation*. Wiley, New York, 1974.
16. Sage, A. P., and Melsa, J. L., *Estimation Theory with Application to Communications and Control*. McGraw-Hill, New York, 1971.
17. Schweppe, F. C., *Uncertain Dynamic Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
18. Van Trees, H. L., *Detection, Estimation and Modulation Theory*, Vol. 1. Wiley, New York, 1968.

SCAAT: Incremental Tracking with Incomplete Information

Greg Welch and Gary Bishop

University of North Carolina at Chapel Hill†

Abstract

We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods.

Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (*Space Synchro*) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose.

Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations. Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations.

CR Categories and Subject Descriptors: I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism—Virtual reality; I.4.4 [Image Processing] Restoration—Kalman filtering; I.4.8 [Image Processing] Scene Analysis—Sensor fusion; G.0 [Mathematics of Computing] General—Numerical Analysis, Probability and Statistics, Mathematical Software.

Additional Key Words and Phrases: virtual environments tracking, feature tracking, calibration, autocalibration, delay, latency, sensor fusion, Kalman filter.

† CB 3175, Sitterson Hall, Chapel Hill, NC, 27599-3175
welch@cs.unc.edu, http://www.cs.unc.edu/~welch
gb@cs.unc.edu, http://www.cs.unc.edu/~gb

1 INTRODUCTION

The method we present requires, we believe, a fundamental change in the way people think about estimating a set of unknowns in general, and tracking for virtual environments in particular. Most of us have the preconceived notion that to estimate a set of unknowns we need as many constraints as there are degrees of freedom at any particular instant in time. What we present instead is a method to constrain the unknowns *over time*, continually refining an estimate for the solution, a *single constraint at a time*.

For applications in which the constraints are provided by real-time observations of physical devices, e.g. through measurements of sensors or visual sightings of landmarks, the SCAAT method isolates the effects of error in individual measurements. This isolation can provide improved filtering as well as the ability to individually calibrate the respective devices or landmarks concurrently and continually while tracking. The method facilitates user motion prediction, multisensor or multiple modality data fusion, and in systems where the constraints can only be determined sequentially, it provides estimates at a higher rate and with lower latency than multiple-constraint (batch) approaches.

With respect to tracking for virtual environments, we are currently using the SCAAT method with a new version of the UNC wide-area optoelectronic tracking system (section 4). The method could also be used by developers of commercial tracking systems to improve their existing systems or it could be employed by end-users to improve custom multiple modality hybrid systems. With respect to the more general problem of estimating a set of unknowns that are related by some set of mathematical constraints, one could use the method to trade estimate quality for computation time. For example one could incorporate individual constraints, one at a time, stopping when the uncertainty in the solution reached an acceptable level.

1.1 Incomplete Information

The idea that one might build a tracking system that generates a new estimate with each individual sensor measurement or *observation* is a very interesting one. After all, individual observations usually provide only partial information about a user's complete state (pose), i.e. they are "incomplete" observations. For example, for a camera observing landmarks in a scene, only limited information is obtained from observations of any single landmark. In terms of control theory, a system designed to operate with only such incomplete measurements is characterized as *unobservable* because the user state cannot be observed (determined) from the measurements.

The notion of observability can also be described in terms of constraints on the unknown parameters of the system being estimated, e.g. constraints on the unknown elements of the system state. Given a particular system, and the corresponding set of unknowns that are to be estimated, let C be defined as the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, let N be the number actually used to generate a new estimate, and let N_{ind} be the number of *independent* constraints that can be formed from the N constraints. For any $N \geq N_{\text{ind}}$ constraints, if $N_{\text{ind}} = C$ the problem is *well constrained*, if $N_{\text{ind}} > C$ it is *over constrained*, and if $N_{\text{ind}} < C$ it is *under-constrained*. (See Figure 1.)

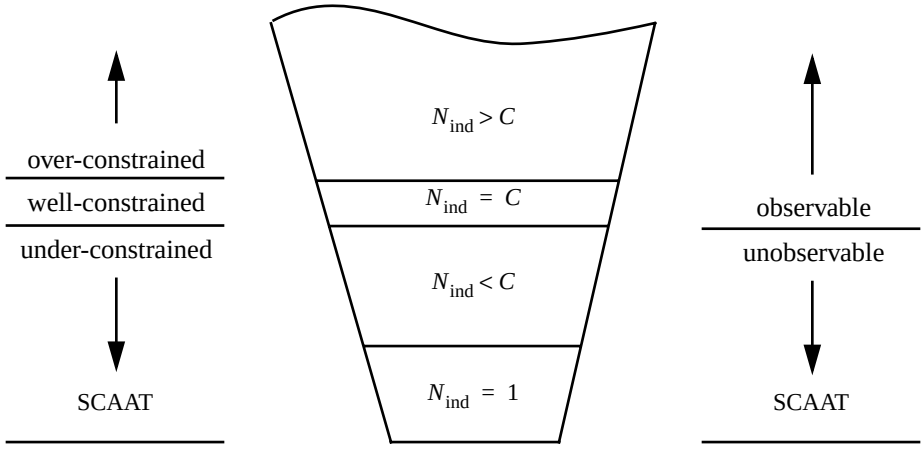

Figure 1: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{ind}$ and $N_{ind} \geq C$, i.e. to use enough measurements to well-constrain or even over-constrain the estimate. The SCAAT approach is to employ the smallest number of constraints available at any one time, generally $N = N_{ind} = 1$ constraint. From this viewpoint, each SCAAT estimate is severely under-constrained.

1.2 Landmark Tracking

Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images.

In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations.

1.3 Putting the Pieces Together

Given a tracker that uses multiple constraints that are each individually incomplete, a *measurement model* for any one of incomplete constraints would be characterized as *locally unobservable*. Such a system must incorporate a sufficient set of these incomplete constraints so that the resulting overall system is observable. The corresponding aggregate measurement model can then be characterized as *globally observable*. Global observability can be obtained over *space* or over *time*. The SCAAT method adopts the latter scheme, even in some cases where the former is possible.

2 MOTIVATION

2.1 The Simultaneity Assumption

Several well-known virtual environment tracking systems collect position and orientation constraints (sensor measurements) sequentially. For example, tracking systems developed by Polhemus and Ascension depend on sensing a sequence of variously polarized electromagnetic waves or fields. A system that facilitated simultaneous polarized excitations would be very difficult if not impossible to implement. Similarly both the original UNC optoelectronic tracking system and the newer HiBall version are designed to observe only one ceiling-mounted LED at a time. Based on the available literature [25,27,37] these systems currently assume (mathematically) that their sequential observations were collected simultaneously. We refer to this as the *simultaneity assumption.* If the target remains motionless this assumption introduces no error. However if the target is moving, the violation of the assumption introduces error.

To put things into perspective, consider that typical arm and wrist motion can occur in as little as 1/2 second, with typical “fast” wrist tangential motion occurring at 3 meters/second [1]. For the current versions of the above systems such motion corresponds to approximately 2 to 6 centimeters of translation *throughout* the sequence of measurements required for a single estimate. For systems that attempt sub-millimeter accuracies, even slow motion occurring during a sequence of sequential measurements impacts the accuracy of the estimates.

The error introduced by violation of the simultaneity assumption is of greatest concern perhaps when attempting any form of system *autocalibration*. Gottschalk and Hughes note that motion during their autocalibration procedure must be severely restricted in order to avoid such errors [19]. Consider that for a multiple-measurement system with 30 milliseconds total measurement time, motion would have to be restricted to approximately 1.5 centimeters/second to confine the translation (throughout a measurement sequence) to 0.5 millimeters. For complete autocalibration of a large (wide-area) tracking system, this restriction results in lengthy specialized sessions.

2.2 Device Isolation & Autocalibration

Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems. While intrinsic sensor parameters can often be determined off-line, e.g. by the manufacturer, this is generally not the case for extrinsic parameters. For example it can be difficult to determine the exact geometric relationship between the various sensors of a hybrid system. Consider that the coordinate system of a magnetic sensor is located at some unknown location inside the sensor unit. Similarly the precise geometric relationship between visible landmarks used in a vision-based system is often difficult to determine. Even worse, landmark positions can change over time as, for example, a patient's skin deforms with pressure from an ultrasound probe. In general, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or *autocalibration* attractive.

The general idea for autocalibration is not new. See for example [19,45]. However, because the SCAAT method *isolates* the measurements provided by each sensor or modality, the method provides a new and elegant means to autocalibrate concurrently while tracking. Because the SCAAT method isolates the individual measurements, or measurement dimensions, individual source and sensor imperfections are more easily identified and dealt with. Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.1 would be removed, and autocalibration could be performed *while concurrently tracking a target.*

The isolation enforced by the SCAAT approach can improve results even if the constraints are obtained simultaneously through multidimensional measurements. An intuitive explanation is that if the elements (dimensions) are corrupted by independent noise, then incorporating the elements independently can offer improved filtering over a batch or ensemble estimation scheme.

2.3 Temporal Improvements

Per Shannon's sampling theorem [24] the measurement or *sampling* frequency should be at least twice the true target motion bandwidth, or an estimator may track an alias of the true motion. Given that common arm and head motion bandwidth specifications range from 2 to 20 Hz [13,14,36], the *sampling* rate should ideally be greater than 40 Hz. Furthermore, the *estimate* rate should be as high as possible so that normally-distributed white estimate error can be discriminated from any non-white error that might be observed during times of significant target dynamics, and so estimates will always reflect the most recent user motion.

In addition to increasing the estimate rate, we want to reduce the latency associated with generating an improved estimate, thus reducing the overall latency between target motion and visual feedback in virtual environment systems [34]. If too high, such latency can impair adaptation and the illusion of presence [22], and can cause motion discomfort or sickness. Increased latency also contributes to problems with head-mounted display registration [23] and with motion prediction [4,15,29]. Finally, post-rendering image deflection techniques are sometimes employed in an attempt to address latency variability in the rendering pipeline [32,39]. Such methods are most effective when they have access to (or generate) accurate motion predictions and low-latency tracker updates. With accurate prediction the best possible position and orientation information can be used to render a preliminary image. With fast tracker updates there is higher probability that when the preliminary image is ready for final deflection, recent user motion has been detected and incorporated into the deflection.

With these requirements in mind, let us examine the effect of the measurements on the estimate latency and rate. Let t_m be the time needed to determine one constraint, e.g. to measure a sensor or extract a scene landmark, let N be the number of (sequential) constraints used to compute a complete estimate, and let t_c be the time needed to actually compute that estimate. Then the estimate latency t_e and rate r_e are

$$t_e = Nt_m + t_c \ , \qquad r_e = \frac{1}{t_e} = \frac{1}{Nt_m + t_c} \ . \tag{1}$$

As the number of constraints N increases, equation (1) shows how the estimate latency and rate increase and decrease respectively. For example the Polhemus Fastrak, which uses the SPASYN (*Space Synchro*) method for determining relative position and orientation, employs $N = 3$ sequential electromagnetic excitations and measurements per estimate [25,27,37], the original University of North Carolina (UNC) optoelectronic tracking system sequentially observed $10 \le N \le 20$ beacons per estimate [3,44], and the current UNC hybrid landmark-magnetic tracking system extracts (from a camera image) and then incorporates $N = 4$ landmarks per update. The SCAAT method seeks to improve the latencies and data rates of such systems by updating the current estimate with each new (individual) constraint, i.e. by fixing N at 1. In other words, it increases the estimate rate to approximately the rate that individual constraints can be obtained and likewise decreases the estimate latency to approximately the time required to obtain a single constraint, e.g. to perform a single measurement of a single sensor, or to extract a single landmark.

Figure 2 illustrates the increased data rate with a timing diagram that compares the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation. In contrast to the SPASYN system, a SCAAT implementation would generate a new estimate after sensing each *individual* excitation vector rather than waiting for a complete pattern.

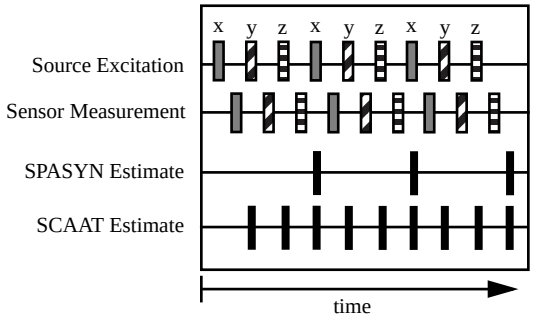

Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

2.4 Data Fusion & Hybrid Systems

The Kalman filter [26] has been widely used for data fusion. For example in navigation systems [17,30], virtual environment tracking systems [5,12,14], and in 3D scene modeling [20,42]. However the SCAAT method represents a new approach to Kalman filter based *multi-sensor data fusion.* Because constraints are intentionally incorporated one at a time, one can pick and choose which ones to add, and when to add them. This means that information from different sensors or modalities can be woven together in a common, flexible, and expeditious fashion. Furthermore, one can use the approach to ensure that each estimate is computed from the most recently obtained constraint.

Consider for a moment the UNC hybrid landmark-magnetic presented at SIGGRAPH 96 [41]. This system uses an off-the-shelf Ascension magnetic tracking system along with a vision-based landmark recognition system to achieve superior synthetic and real image registration for augmented reality assisted medical procedures. The vision-based component attempts to identify and locate multiple known landmarks in a single image before applying a correction to the magnetic readings. A SCAAT implementation would instead identify and locate only one landmark per update, using a new image (frame) each time. Not only would this approach increase the frequency of landmark-based correction (given the necessary image processing) but it would offer the added benefit that unlike the implementation presented in [41], no special processing would be needed for the cases where the number of visible landmarks falls below the number C necessary to determine a complete position and orientation solution. The SCAAT implementation would simply cycle through any available landmarks, one at a time. Even with only one visible landmark the method would continue to operate as usual, using the information provided by the landmark sighting to refine the estimate where possible, while increasing the uncertainty where not.

3 METHOD

The SCAAT method employs a *Kalman filter* (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a *predictor-corrector* fashion, predicting short-term (since the last estimate) changes in the state using a *dynamic model*, and then correcting them with a measurement and a corresponding *measurement model*. The *extended* Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of *nonlinear* systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46].

The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community.

In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration.

Throughout we use the following conventions.

x	=	scalar (lower case)
$\vec{x}$	=	general vector (lower case, arrow) indexed as $\vec{x}[r]$
$\hat{x}$	=	filter estimate vector (lower case, hat)
A	=	matrix (capital letters) indexed as $A[r, c]$
A^{-1}	=	matrix inverse
I	=	the identity matrix
β^-	=	matrix/vector *prediction* (super minus)
β^T	=	matrix/vector transpose (super T)
α_i	=	matrix/vector/scalar identifier (subscript)
$E\{\bullet\}$	=	mathematical expectation

3.1 Tracking

3.1.1 Main Tracker Filter

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time δt we model the target's dynamic motion with the following linear difference equation:

$$\vec{x}(t + \delta t) = A(\delta t)\vec{x}(t) + \vec{w}(\delta t). \quad (2)$$

In the standard model corresponding to equation (2), the n dimensional Kalman filter *state vector* $\vec{x}(t)$ would completely describe the target position and orientation at any time t. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\vec{x}(t)$ we maintain the target position as the Cartesian coordinates (x, y, z), and the *incremental* orientation as small rotations (ϕ, θ, ψ) about the (x, y, z) axis. Externally we maintain the target orientation as the *external quaternion* $\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations (ϕ, θ, ψ) are factored into the external quaternion $\vec{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\vec{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\vec{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \quad (3)$$

and the four-element external orientation quaternion

$$\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)), \quad (4)$$

where the time designations have been omitted for clarity.

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time t to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \tag{5}$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval δt. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{6}$$

Because our implementation is discrete with inter sample time δt, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration δt only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] &= Q(\delta t)[j,i] = \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \tag{7}$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

The $\vec{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.

For each sensor *type* σ we define the $m_\sigma \times 1$ *measurement vector* $\vec{z}_\sigma(t)$ and corresponding *measurement function* $\vec{h}_\sigma(\bullet)$ such that

$$\vec{z}_{\sigma,t} = \vec{h}_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t) + \vec{v}_\sigma(t). \tag{8}$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion) elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\vec{v}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\vec{v}_\sigma(t)\vec{v}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{9}$$

For each measurement function $\vec{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t)[i,j] \equiv \frac{\partial}{\partial \vec{x}[j]}\vec{h}_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t)[i], \tag{10}$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type σ) and source becomes available at time t:

a. Compute the time δt since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^-H^T(HP^-H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\vec{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\vec{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

g. Update the external orientation of equation (4) per the change indicated by the (ϕ, θ, ψ) elements of the state.*

$$\begin{aligned} \Delta\hat{\alpha} &= \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi]) \\ \hat{\alpha} &= \hat{\alpha} \otimes \Delta\hat{\alpha} \end{aligned} \tag{16}$$

h. Zero the orientation elements of the state vector.

$$\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0 \tag{17}$$

The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian H in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of K in (13) is of rank m_σ (2×2 for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\hat{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately 100µs on a 200 MHz PC-compatible computer.)

3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\hat{z}_t = \left[\hat{z}^T_{\sigma_1, t_1} \; \cdots \; \hat{z}^T_{\sigma_N, t_N} \right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time δt between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\hat{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain K which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian H. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\hat{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\hat{b}_t$ and $\hat{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$\widehat{\alpha}$ = augmented matrix/vector (wide hat)

3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\hat{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\hat{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\widehat{x}(t - \delta t) = \left[\hat{x}^T(t - \delta t) \;\; \hat{x}^T_{b,t}(t - \delta t) \;\; \hat{x}^T_{c,t}(t - \delta t) \right]^T, \tag{18}$$

the error covariance matrix

$$\widehat{P}(t - \delta t) = \begin{bmatrix} P(t - \delta t) & 0 & 0 \\ 0 & P_{b,t}(t - \delta t) & 0 \\ 0 & 0 & P_{c,t}(t - \delta t) \end{bmatrix}, \tag{19}$$

the state transition matrix

$$\widehat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{20}$$

and the process noise matrix

$$\widehat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \tag{21}$$

* The operation $\alpha \otimes \Delta\alpha$ is used to indicate a quaternion multiply [9].

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\check{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\check{b}_t$ and $\check{c}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{b,t}(t) &= \widehat{x}(t)[i\ldots j] \\ P_{b,t}(t) &= \widehat{P}(t)[i\ldots j, i\ldots j] \\ \hat{x}_{c,t}(t) &= \widehat{x}(t)[k\ldots l] \\ P_{c,t}(t) &= \widehat{P}(t)[k\ldots l, k\ldots l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and n, n_b, and n_c are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3\times3 = 9$ words for the noise covariance matrix, and $3\times3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000\times8\times(3+9+9) = 504{,}000$ bytes.

3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter η of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132):

a. The filter must be uniformly completely observable,

b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and

c. the dynamic behavior represented by $A(\delta t)$ in equation (2) must be bounded from above.

As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints *over time*. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47].

3.4 Measurement Ordering

Beyond a simple round-robin approach, one might envision a measurement scheduling algorithm that makes better use of the available resources. In doing so one would like to be able to monitor and control uncertainty in the state vector. By periodically observing the eigenvalues and eigenvectors of the error covariance matrix $P(t)$, one can determine the directions in state-space along which more or less information is needed [21]. This approach can be used to monitor the stability of the tracker, and to guide the source/sensor ordering.

4 EXPERIMENTS

We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their $\alpha1$ modeling environment.

Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system.

In this section we present results from both simulations performed during the design and development of the HiBall, and

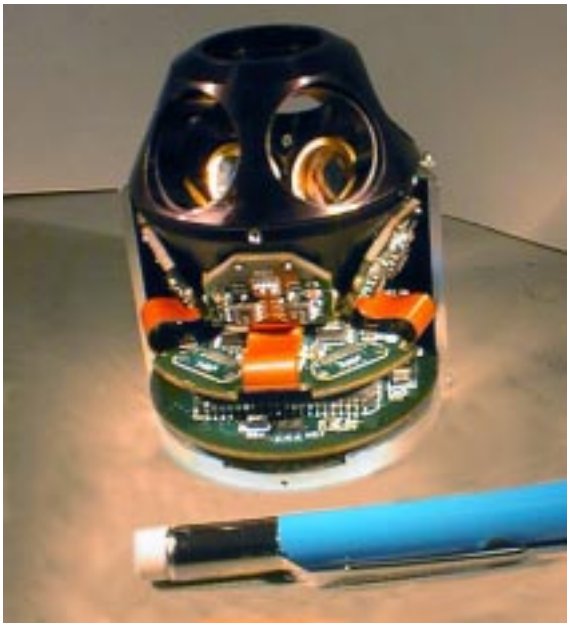

Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit.

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the (u, v) image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $\grave{z}_\sigma = [u, v]^T$. The measurement function $\grave{h}_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\grave{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\grave{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\grave{b}_t$ (position in world coordinates) were then sent to the tracker.

For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate N. In each case the respective estimates were compared with the truth data set for performance evaluation.

4.1 Tracker Filter

The 12 element state vector $\hat{x}(t)$ for the main tracker filter contained the elements shown in (3). Each of the 3000 beacon filters was allocated a 3 element state vector

$$\hat{x}_b = \begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^T$$

where (x_b, y_b, z_b) represents the beacon's estimated position in cartesian (world) coordinates. The 12×12 state transition matrix for the main tracker filter was formed as discussed section 3.1, and for each beacon filter it was the 3×3 identity matrix. The 12×12 process noise matrix for the main tracker was computed using (7), using elements of $\grave{\eta}$ that were determined off-line using Powell's method and a variety of real motion data. For each beacon filter we used an identical noise covariance matrix

$$Q_b(\delta t)[i, j] = \begin{cases} \eta_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$, with beacon position variance η_b also determined off-line. (See [47] for the complete details.) At each estimate step, the *augmented* 15 element state vector, 15×15 process noise matrix, 15×15 state transition matrix, and 15×15 error covariance matrix all resembled (18)-(21) (without the camera parameter components). The measurement noise model was distance dependent (beacon light falls-off with distance) so $R_\sigma(t)$ from (9) was computed prior to step (d), by using a beacon distance estimate (obtained from the user and beacon positions in the predicted state $\grave{\bar{x}}$) to project a distance-dependent electrical variance onto the camera.

4.2 Initialization

The position and orientation elements of the main tracker state were initialized with the true user position and orientation, and the velocities were initialized to zero. The 3000 beacon filter state vectors were initialized with (potentially erroneous) beacon position estimates. The main tracker error covariance matrix was initialized to the null matrix. All beacon filter error covariance matrices were initialized to

$$P_b(0)[i, j] = \begin{cases} (0.001)^2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$, to reflect 1 millimeter of uncertainty in the initial beacon positions.

While for the presented simulations we initialized the filter state with the true user pose information, we also performed (but will not show here) simulations in which the state elements were initialized to arbitrary values, e.g. all zeros. It is a testament to the stability of the method that in most cases the filter completely converged in under a tenth of a second, i.e. with fewer than 100 measurements. (In a few cases the camera was facing away from the beacon, a condition not handled by our simulator.)

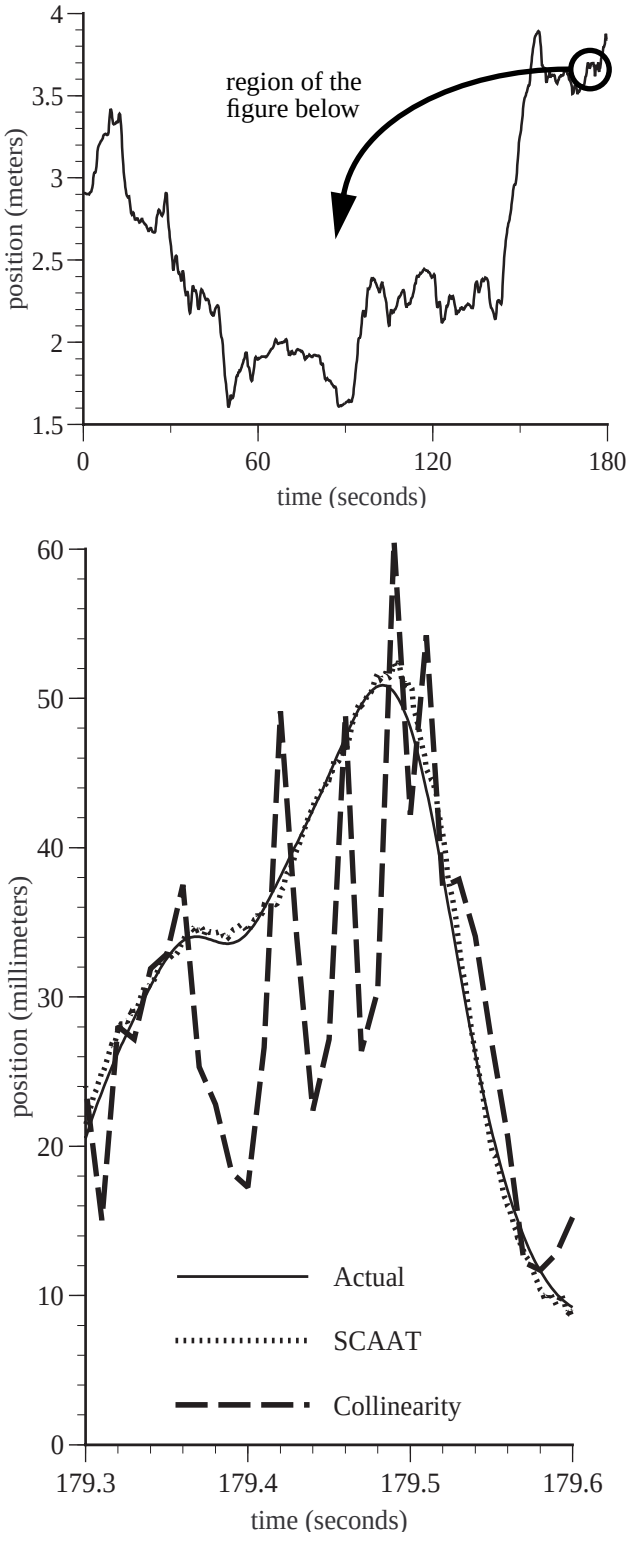

Figure 4: The upper plot depicts the entire 3 minutes of *x*-axis position data from user motion data set ‘a’ of sets ‘a’-’f’. The lower plot shows a close-up of a short portion of the simulation. Collinearity here used $N = 10$ beacons per observation, hence its lower estimate rate. On the other hand, notice that the SCAAT estimates and the actual (truth) data are almost indistinguishable.

4.3 Simulation Results

We present here only comparisons of the SCAAT method with the Collinearity method, the “conventional approach” mentioned in the accompanying video. More extensive simulation results can be found in [47], including tests for stability under “cold starts” and periodic loss of data. All error measures reflect the RMS position error for a set of three imaginary points located approximately at arms length. This approach combines both position and orientation error into a metric that is related to the error a user would encounter in [HMD] screen space.

Figure 4 contains two related plots. The upper plot shows the entire three minutes (180 seconds) of the *x*-axis position for the first of seven data sets, data set ‘a’. The lower plot shows a close-up of a particular segment of 300 milliseconds near the end. Notice that the Collinearity estimates appear very jerky. This is partially a result of the lower estimate rate, it is using $N = 10$ beacon observations to compute an estimate, and partially due to the method’s inability to deal with the erroneous beacon position data. In contrast, the SCAAT method hugs the actual motion track, appearing both smooth and accurate. This is partly a result of the higher update rate (10 times Collinearity here), and partly the effects of Kalman filtering, but mostly the accuracy is a result of the SCAAT autocalibration scheme. With the autocalibration turned on, the initially erroneous beacon positions are being refined at the same high rate that user pose estimates are generated.

Figure 5 shows progressively improving estimates as the number of beacons N is reduced from 15 (Collinearity) down to 1 (SCAAT), and a clear improvement in the accuracy when autocalibration is on. Consider for a moment that the motion prediction work of Azuma and Bishop [4] was based on jerky Collinearity estimates similar to those in Figure 4. The smooth and accurate SCAAT estimation should provide a much better basis for motion prediction, which could in turn provide a more effective means for addressing other system latencies such as those in the rendering pipeline. The improved accuracy should also improve post-rendering warping or image deflection [32,39].

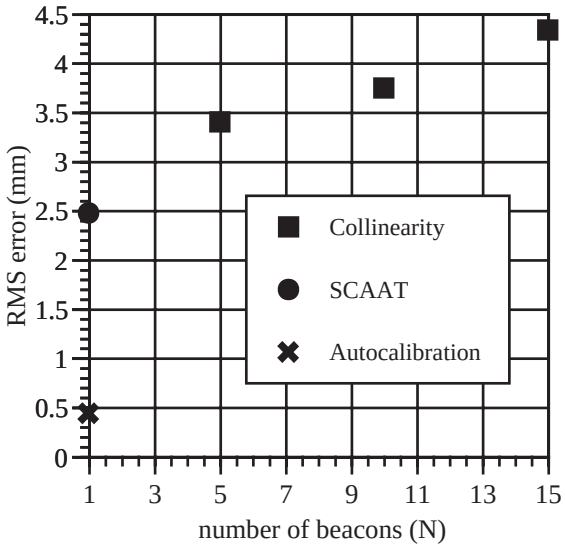

Figure 5: As the number of beacons N is reduced from 15 to 5, the Collinearity results improve slightly. (The Collinearity algorithm generally becomes unstable with $N \leq 4$.) The SCAAT results, with $N = 1$ beacons, are better, and especially good once autocalibration is turned on.

As further evidence of the smoothing offered by the SCAAT approach, Figure 6 presents an error spectra comparison between a Collinearity implementation with $N = 10$, and a SCAAT implementation with and without autocalibration. Even without autocalibration the SCAAT output has significantly less noise than collinearity, and with autocalibration it is better by more than a factor of 10. These reductions in noise are clearly visible to the HMD user as a reduction in the amount of jitter in virtual-world objects.

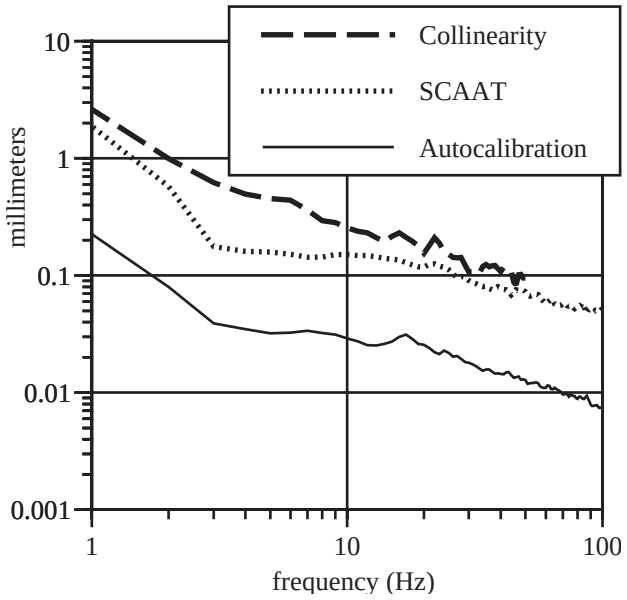

Figure 6: Here we show an error spectra comparison for the Collinearity method with $N = 10$ beacons, and the SCAAT method with and without autocalibration.

Figure 7 provides results for all seven of the real-user motion data sets. Again the Collinearity implementations observe $N = 10$ beacons per estimate, while the SCAAT implementations observe only $N = 1$. Because the beacon positions were being autocalibrated during the SCAAT run, we repeated each run, the second time using the beacon position estimation results from the first simulation. The more beacons are sighted during tracking, the better they are located. The second-pass simulation results are identified with the dagger (†) in Figure 7.

Figure 8 presents results that support the claim that the beacon location estimates are actually improving during tracking with autocalibration, as opposed to simply shifting to reduce spectral noise. Note that in the case of data set 'd', the beacon error was reduced nearly 60%.

Finally, we simulated using the SCAAT approach with tracking hardware that allowed truly simultaneous observations of beacons. For the Collinearity and other multiple-constraint methods we simply used the methods as usual, except that we passed them truly simultaneous measurements. For the SCAAT method we took the N simultaneous observations, and simply processed them one at a time with $\delta t = 0$. (See equation (2).) We were, at first, surprised to see that even under these ideal circumstances the SCAAT implementation could perform better, even significantly better than a multiple-constraint method with simultaneous constraints. The reason seems to be autocalibration. Even though the multiple-constraint methods were "helped" by the truly simultaneous observations, the SCAAT method still had the advantage in that it could still autocalibrate the beacons more

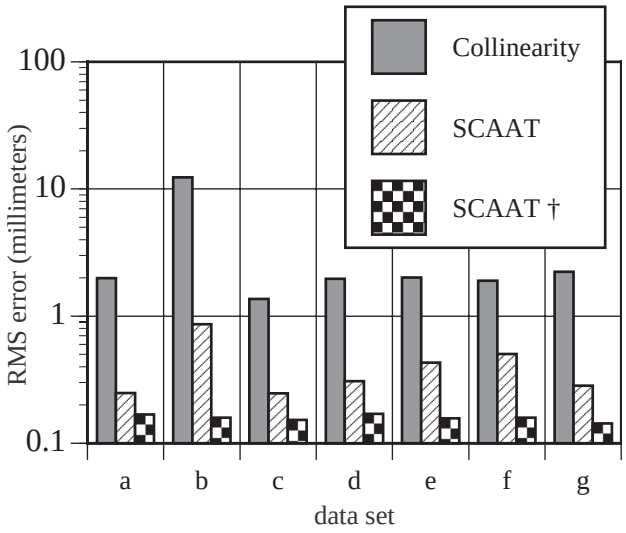

Figure 7: RMS error results for simulations of all seven real user motion data sets. The † symbol indicates a second pass through the motion data set, this time using the already autocalibrated beacons.

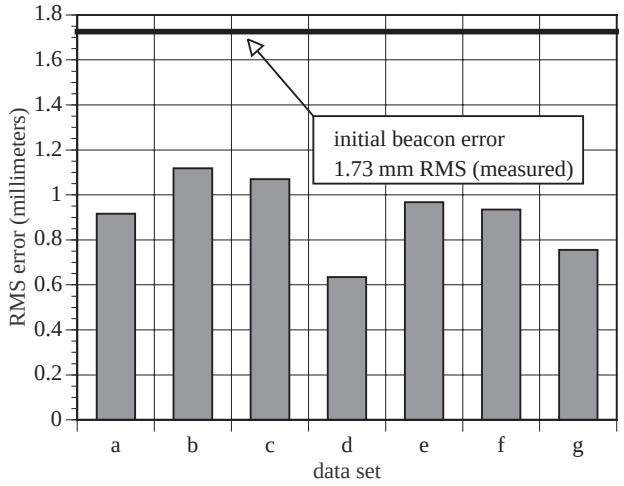

Figure 8: Autocalibration in action. Here we show the final beacon position error for runs through each of the seven user motion data sets.

effectively that any multiple-constraint method. This again arises from the method's inherent isolation of individual observations.

4.4 Real Results

We have demonstrated the SCAAT algorithm with the HiBall tracker, a head-mounted display, and a real application. However, at the time of the submission of this publication we have yet to perform extensive optimization and analysis. As such we present here only limited, albeit compelling results.

The SCAAT code runs on a 200 MHz PC-compatible computer with a custom interface board. With unoptimized code, the system generates new estimates at approximately 700 Hz. We expect the optimized version to operate at over 1000 Hz. Out of the approximately 1.4 millisecond period, the unoptimized SCAAT code takes approximately 100 microseconds and sampling of the sensors takes approximately 200 microseconds. The remaining

time is spent on overhead including a significant amount of unoptimized code to choose an LED and to gather results.

In one experiment we set the HiBall on a flat surface under the ceiling beacons and collected several minutes worth of data. Given that the platform was relatively stable, we believe that the deviation of the estimates provides an indication of the noise in the system. Also, because the HiBall was not moving, we were able to observe the progressive effects of the autocalibration. The standard deviation of the position estimates for the first 15 seconds is shown in Figure 9. With autocalibration off, the estimates deviate approximately 6.0 millimeters in translation and 0.25 degrees in orientation (not shown). With autocalibration on, notice in Figure 9 how the deviation decreases with time, settling at approximately 0.3 millimeters in translation and 0.01 degrees in orientation (again not shown).

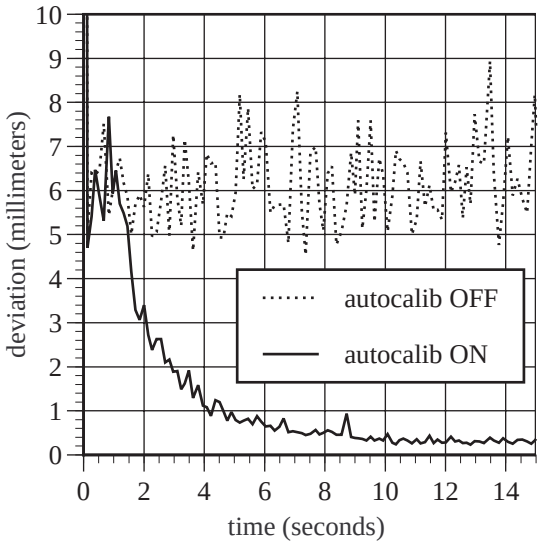

Figure 9: SCAAT position (only) estimate deviation for a Hiball sitting still on a flat surface, with and without autocalibration.

In another experiment we mounted the HiBall on a calibrated translation rail of length one meter, and slid (by hand) the HiBall from one end to the other and then back again. The disagreement between the HiBall and the calibrated position on the rail was less than 0.5 millimeters. The deviation of the measured track from co-linearity was 0.9 millimeters. Because the tolerances of our simple test fixture are of similar magnitude, we are unable to draw conclusions about how much of this disagreement should be attributed to error in the tracking system.

5 CONCLUSIONS

Stepping back from the details of the SCAAT method, we see an interesting relationship: Because the method generates estimates with individual measurements, it not only avoids the simultaneity assumption but it operates faster; by operating faster, it decreases the elapsed time since the previous state estimate; the more recent the previous estimate, the better the prediction in (12); the better the prediction, the more likely we can discriminate bad measurements; if we can discriminate bad measurements, we can autocalibrate the measurement devices; and if we can calibrate the measurement devices, we can improve the individual measurements, thus improving predictions, etc. In other words, the faster, the better.

Looking more closely, it is amazing that such a tracker can function at all. Consider for example the system presented in section 4. Any single beacon sighting offers so few constraints—the user could be theoretically *anywhere*. Similarly, knowledge about where the user was a moment ago is only an indicator of where the user *might* be now. But used together, these two sources of information can offer more constraints than either alone. With a Kalman filter we can extract the information from the previous state and a new (individual) measurement, and blend them to form a better estimate than would be possible using either alone.

The SCAAT method is accurate, stable, fast, and flexible, and we believe it can be used to improve the performance of a wide variety of commercial and custom tracking systems.

Acknowledgements

We would like to thank the tracker team at UNC, in particular Vernon Chi, Steve Brumback, Kurtis Keller, Pawan Kumar, and Phillip Winston. This work was supported by DARPA/ETO contract no. DABT 63-93-C-0048, “Enabling Technologies and Application Demonstrations for Synthetic Environments”, Principle Investigators Frederick P. Brooks Jr. and Henry Fuchs (University of North Carolina at Chapel Hill), and by the National Science Foundation Cooperative Agreement no. ASC-8920219: “Science and Technology Center for Computer Graphics and Scientific Visualization”, Center Director Andy van Dam (Brown University). Principle Investigators Andy van Dam, Al Barr (California Institute of Technology), Don Greenberg (Cornell University), Henry Fuchs (University of North Carolina at Chapel Hill), Rich Riesenfeld (University of Utah).

References

[1] C.G. Atkeson and J.M. Hollerbach. 1985. “Kinematic features of unrestrained vertical arm movements,” Journal of Neuroscience, 5:2318-2330.

[2] Ali Azarbayejani and Alex Pentland. June 1995. “Recursive Estimation of Motion, Structure, and Focal Length,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, June 1995, 17(6).

[3] Ronald Azuma and Mark Ward. 1991. “Space-Resection by Collinearity: Mathematics Behind the Optical Ceiling Head-Tracker,” UNC Chapel Hill Department of Computer Science technical report TR 91-048 (November 1991).

[4] Ronald Azuma and Gary Bishop. 1994. “Improving Static and Dynamic Registration in an Optical See-Through HMD,” SIGGRAPH 94 Conference Proceedings, Annual Conference Series, pp. 197-204, ACM SIGGRAPH, Addison Wesley, July 1994. ISBN 0-201-60795-6

[5] Ronald Azuma. 1995. “Predictive Tracking for Augmented Reality,” Ph.D. dissertation, University of North Carolina at Chapel Hill, TR95-007.

[6] Ted J. Broida and Rama Chellappa. 1986. “Estimation of object motion parameters from noisy images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, January 1986, 8(1), pp. 90-99.

[7] R. G. Brown and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, 2nd Edition*, John Wiley & Sons, Inc.

[8] Vernon L. Chi. 1995. “Noise Model and Performance Analysis of Outward-looking Optical Trackers Using Lateral Effect Photo Diodes,” University of North Carolina, Department of Computer Science, TR 95-012 (April 3, 1995)

[9] Jack C.K. Chou. 1992. “Quaternion Kinematic and Dynamic Differential Equations,” IEEE Transactions on Robotics and Automation, Vol. 8, No. 1, pp. 53-64.

[10] J. L. Crowley and Y. Demazeau. 1993. “Principles and Techniques for Sensor Data Fusion,” *Signal Processing (EURASIP)* Vol. 32. pp. 5-27.

[11] J. J. Deyst and C. F. Price. 1968. “Conditions for Asymptotic Stability of the Discrete Minimum-Variance Linear Estimator,” *IEEE Transactions on Automatic Control*, December, 1968.

[12] S. Emura and S. Tachi. 1994. “Sensor Fusion based Measurement of Human Head Motion,” *Proceedings 3rd IEEE International Workshop on Robot and Human Communication, RO-MAN’94 NAGOYA* (Nagoya University, Nagoya, Japan).

[13] P. Fischer, R. Daniel and K. Siva. 1990. “Specification and Design of Input Devices for Teleoperation,” *Proceedings of the IEEE Conference on Robotics and Automation* (Cincinnati, OH), pp. 540-545.

[14] Eric Foxlin. 1993. “Inertial Head Tracking,” Master’s Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[15] M. Friedman, T. Starner, and A. Pentland. 1992. “Synchronization in Virtual Realities,” *Presence: Teleoperators and Virtual Environments*, 1:139-144.

[16] S. Ganapathy. November 1984. “Camera Location Determination Problem,” AT&T Bell Laboratories Technical Memorandum, 11358-841102-20-TM.

[17] G. J. Geier, P. V. W. Loomis and A. Cabak. 1987. “Guidance Simulation and Test Support for Differential GPS (Global Positioning System) Flight Experiment,” National Aeronautics and Space Administration (Washington, DC) NAS 1.26:177471.

[18] A. Gelb. 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.

[19] Stefan Gottschalk and John F. Hughes. 1993. “Autocalibration for Virtual Environments Tracking Hardware,” Proceedings of ACM SIGGRAPH 93 (Anaheim, CA, 1993), Computer Graphics, Annual Conference Series.

[20] A Robert De Saint Vincent Grandjean. 1989. “3-D Modeling of Indoor Scenes by Fusion of Noisy Range and Stereo Data,” *IEEE International Conference on Robotics and Automation* (Scottsdale, AZ), 2:681-687.

[21] F. C. Ham and R. G. Brown. 1983. “Observability, Eigenvalues, and Kalman Filtering,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-19, No. 2, pp. 269-273.

[22] R. Held and N. Durlach. 1987. *Telepresence, Time Delay, and Adaptation*. NASA Conference Publication 10023.

[23] Richard L. Holloway. 1995. “Registration Errors in Augmented Reality Systems,” Ph.D. dissertation, The University of North Carolina at Chapel Hill, TR95-016.

[24] O. L. R. Jacobs. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.

[25] Roy S. Kalawsky. 1993. *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley Publishers.

[26] R. E. Kalman. 1960. “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

[27] J. B. Kuipers. 1980 “SPASYN—An Electromagnetic Relative Position and Orientation Tracking System,” *IEEE Transactions on Instrumentation and Measurement*, Vol. IM-29, No. 4, pp. 462-466.

[28] Richard Lewis. 1986. *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.

[29] J. Liang, C. Shaw and M. Green. 1991. “On Temporal-spatial Realism in the Virtual Reality Environment,” *Fourth Annual Symposium on User Interface Software and Technology*, pp. 19-25.

[30] R. Mahmoud, O. Loffeld and K. Hartmann. 1994. “Multisensor Data Fusion for Automated Guided Vehicles,” *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 2247, pp. 85-96.

[31] Peter S. Maybeck. 1979. *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.

[32] Thomas Mazuryk and Michael Gervautz. 1995. “Two-Step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments,” *EUROGRAPHICS ‘95*, Vol. 14, No. 3, pp. 30-41.

[33] K. Meyer, H. Applewhite and F. Biocca. 1992. A Survey of Position Trackers. *Presence*, a publication of the *Center for Research in Journalism and Mass Communication*, The University of North Carolina at Chapel Hill.

[34] Mark Mine. 1993. “Characterization of End-to-End Delays in Head-Mounted Display Systems,” The University of North Carolina at Chapel Hill, TR93-001.

[35] National Research Council. 1994. “Virtual Reality, Scientific and Technological Challenges,” National Academy Press (Washington, DC).

[36] P.D. Neilson. 1972. “Speed of Response or Bandwidth of Voluntary System Controlling Elbow Position in Intact Man,” *Medical and Biological Engineering*, 10:450-459.

[37] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. 1979. “Magnetic Position and Orientation Tracking System,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-15, 709-718.

[38] Selspot Technical Specifications, Selcom Laser Measurements, obtained from Innovision Systems, Inc. (Warren, MI).

[39] Richard H. Y. So and Michael J. Griffin. July-August 1992. “Compensating Lags in Head-Coupled Displays Using Head Position Prediction and Image Deflection,” *AIAA Journal of Aircraft*, Vol. 29, No. 6, pp. 1064-1068

[40] H. W. Sorenson. 1970. “Least-Squares estimation: from Gauss to Kalman,” *IEEE Spectrum*, Vol. 7, pp. 63-68, July 1970.

[41] Andrei State, Gentaro Hirota, David T. Chen, Bill Garrett, Mark Livingston. 1996. “Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking,” SIGGRAPH 96 Conference Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison Wesley, August 1996.

[42] J. V. L. Van Pabst and Paul F. C. Krekel. “Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space,” TNO Physics and Electronics Laboratory, The Hague, The Netherlands. [cited 19 November 1995]. Available from http://www.bart.nl/~lawick/index.html.

[43] J. Wang, R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. 1990. “Tracking a head-mounted display in a room-sized environment with head-mounted cameras,” *Proceeding: SPIE'90 Technical Symposium on Optical Engineering & Photonics in Aerospace Sensing* (Orlando, FL).

[44] Mark Ward, Ronald Azuma, Robert Bennett, Stefan Gottschalk, and Henry Fuchs. 1992. “A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems,” *Proceedings of 1992 Symposium on Interactive 3D Graphics* (Cambridge, MA, 29 March - 1 April 1992), pp. 43-52.

[45] Wefald, K.M., and McClary, C.R. “Autocalibration of a laser gyro strapdown inertial reference/navigation system,” *IEEE PLANS '84*. Position Location and Navigation Symposium Record.

[46] Greg Welch and Gary Bishop. 1995. “An Introduction to the Kalman Filter,” University of North Carolina, Department of Computer Science, TR 95-041.

[47] Greg Welch, 1996. “SCAAT: Incremental Tracking with Incomplete Information,” University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051.

[48] H. J. Woltring. 1974. “*New possibilities for human motion studies by real-time light spot position measurement*,” Biotelemetry, Vol. 1.