SIGGRAPH 2001

Course 11

Tracking: Beyond 15 Minutes of Thought

B. Danette Allen [2]
Gary Bishop [1,2]
Greg Welch [1,2]

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

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

1. Organizer
2. Presenter

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
6D	six-dimensional
A/D	analog-to-digital
AR	Augmented Reality
cm	centimeter
CS	coordinate system
DLP	digital light projector
DOF	degree of freedom
EKF	extended Kalman filter
ft	feet
GPS	Global Positioning System
Hz	Hertz
HMD	head-mounted display
kHz	kilohertz
KF	Kalman filter
LCD	liquid crystal display
LED	light emitting diode
m	meter
mm	millimeter
ms	millisecond
MCAAT	multiple-constraints-at-a-time
PV	position-velocity
RMS	root-mean-square
s	second
SCAAT	single-constraint-at-a-time
UNC-CH	University of North Carolina at Chapel Hill
VE	virtual environment
VR	Virtual Reality

Preface

Nearly everyone of a technical bent who has thought about the problem of tracking for graphics for 15 minutes or so believes they have an easy answer—“Why don’t you just…” This course (new for SIGGRAPH 2001) is designed to take you beyond those first few minutes of thought with an “under the hood” look at how these systems work. Our goal is to convey a basic understanding of the characteristics of the available technologies, their fundamental limitations, in what cases those limitations hurt you, and some things you can do to improve your results.

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 an electronic bibliographic version of “Tracking Bibliography” on page 97 are available at

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

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

Course Syllabus

Time	Speaker	Topic	**Time**
1:30 PM	Bishop	Welcome and Introduction	0:15
1:45 PM	Allen	Tracking technologies	0:25
2:10 PM	Bishop	Source/sensor configurations	0:20
2:30 PM	Welch	User and sensor uncertainty/information	0:30
3:00 PM	-	Break	0:15
3:15 PM	Bishop	Traditional approaches	0:20
3:35 PM	Welch	Stochastic approaches	0:25
4:00 PM	Welch	Error sources (spatial and temporal)	0:25
4:25 PM	Bishop	Motion prediction	0:20
4:45 PM	Bishop	Conclusions (summary, resources, etc.)	0:15
5:00 PM			
		Total time	3:30

1. Introduction

One of the important problems in Virtual Environment (VE) research today is that of providing a fast, accurate, and unobtrusive method for reliably *tracking* a computer user's real-world position and orientation or *pose*. Such tracking is necessary in VE systems because a user must continually be provided with two-dimensional computer generated images that match the user's three-dimensional real-world position and orientation. Similarly, human *motion capture* systems are often used for physiological studies (e.g., analysis of athletic motion) or special effects in motion pictures. Usually if the user's position and orientation are not tracked accurately or fast enough, disturbing or even harmful effects can be observed.

Systems for tracking and motion capture for interactive computer graphics have been explored for over 30 years (Sutherland, 1968). Throughout the years commercial and research teams have explored mechanical, magnetic, acoustic, inertial, and optical technologies. Complete historical surveys include (Bhatnagar, 1993; Burdea & Coiffet, 1994; Meyer, Applewhite, & Biocca, 1991; Meyer, Applewhite, & Biocca, 1992; Mulder, 1994, 1998). Commercial magnetic tracking systems for example (Ascension, 2000; Polhemus, 2000) have enjoyed popularity as a result of a small user-worn component and relative ease of use. Optical systems have been developed for 3D *motion capture* (MAC, 2000; Woltring, 1974, 1976) and 6D tracking for visual simulations via head-worn displays (Wang, 1990; Wang et al., 1990; Ward, Azuma, Bennett, Gottschalk, & Fuchs, 1992; Welch et al., 1999, 2001). Recently inertial *hybrid systems* have been gaining popularity because of the reduced high-frequency noise and direct measurements of derivatives (Foxlin, Harrington, & Pfeifer, 1998; Intersense, 2000).

1.1 Course Description

Every year, dozens of vendors display different systems for motion capture and tracking at the SIGGRAPH exhibition, while researchers continue to pursue new approaches in the laboratory. Why are there so many different approaches to this seemingly simple problem? How do the systems differ? What are the strong and weak points of each? How can you decide which is appropriate for your application?

We will attempt to answer these questions and more in this course on some fundamental technologies behind tracking and motion-capture systems. We will use actual systems (commercially available) as examples, describing some algorithms used in popular magnetic, inertial, and optical tracking systems, relating the pros and cons of the systems to the fundamental technologies and the algorithms. While there have been previous SIGGRAPH courses on motion capture, they have primarily concentrated on the graphics application. Instead we will take you “under the hood” of several systems so that you might better understand the performance (or lack thereof) that you experience with different applications, and perhaps improve your results by adjusting your setup to better match the technology.

1.2 Speaker/Author Biographies

Danette Allen is a Ph.D. Candidate in the Department of Computer Science at the University of North Carolina at Chapel Hill. Her primary research interest is tracking technologies but her interests also include hardware and software for man-machine interaction and virtual environments. Allen graduated from North Carolina State University with degrees in Electrical Engineering and Computer Engineering in 1988 and 1989. She received her Master’s Diploma in Business Administration from Manchester Business School, U.K. in 1990. In 1997, she received an M.E. in computer engineering from Old Dominion University. Allen began working at NASA Langley Research Center (LaRC) in Hampton, Virginia in 1991 where she is currently employed. She is a recipient of NASA’s Silver Snoopy award, the astronauts’ award for outstanding performance in flight safety and mission success. She 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.

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.

1.3 Acknowledgements

We thank Leandra Vicci (UNC-Chapel Hill), Richard Holloway (3rdTech, Inc.), and Warren Robinett for their valuable contributions to this course pack. In particular we thank Leandra for her contributions to Section 3.1 on classifications of tracking approaches by physical medium, and Rich for permission to make use of material from his Ph.D. dissertation (Holloway, 1995) in Section 5.1 on tracking error.

2. Background

2.1 Basic Coordinate Transforms

This section briefly discusses the basic 2D and 3D geometrical transformations used in computer graphics and tracking. Most of the information is taken from (Foley, van Dam, Feiner, & Hughes, 1997) and (Robinett & Holloway, 1994), the latter of which is included in Appendix C of this course pack.

We assume a right-handed coordinate system as shown in Figure 2.1.

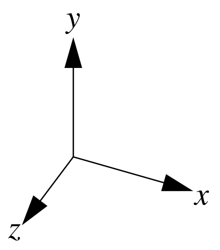


Figure 2.1: Right-handed coordinate system

Translation, *rotation* and *scaling* are the essential transformations in computer graphics and tracking. Translation displaces points by a fixed distance in a given direction. Scaling increases or decreases the size of an object. Rotation revolves a point around a specified axis. Figure 2.2 illustrates these transformations in 2D space.

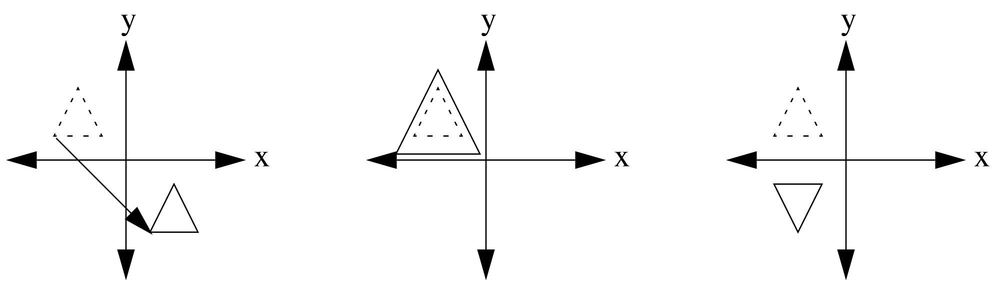


Figure 2.2: Translation, scaling and rotation (around the x-axis) in 2D

In a right-handed coordinate system, positive rotations are defined such that a 90° counterclockwise rotation transforms one positive axis into another. table 2.1 (Foley et al., 1997) follows from this convention.

Axis of Rotation	Direction of Positive Rotation
x	y to z
y	z to x
z	x to y

Table 2.1: Positive Rotations

2.1.1 Coordinate Systems

Coordinate systems serve as a 6D basis to which all points, lines, objects, etc. are referenced. For example, in describing the human head, we might declare a head coordinate system with its origin at the extreme tip of the nose. From this origin, we can determine through a series of transformations (translation, rotation and scaling) where the eyes, ears, mouth, etc. are located. Beyond this we can think about a world coordinate system which references the origin of the head coordinate system in the world.

Tracker measurements are provided in the *tracker coordinate system*. Consider an acoustic tracker that provides a range measurement d [m] from room-mounted transmitters to a sensor mounted on the user. This d, measured from the transmitter to the sensor, is referenced to the tracker's coordinate system. What we *want* is the user's head position in room or world coordinates. This is further complicated if we want to know the position of the user's eyes. We must transform from the trackers's coordinate frame to the coordinate frame of the user's head. If we are rendering in stereo, from the user's head coordinate frame, we transform to the both of the user's eyes.

2.1.2 Affine Transformations

With respect to computer graphics, we use the term *transform* to refer to the mathematical operation of modifying a graphics primitive by adding, multiplying, and even dividing its numerical elements achieve such effects as translation, rotation, and perspective projection. In particular, translation, rotation and scaling are classified as *affine* transformations. They preserve parallelism of lines but not angles and lengths. Another less often used affine transformation is a form of distortion known as *shearing*.

2.1.2.1 2D Transformations

We define a point in 2D space as a column vector,

$$\vec{P} = \begin{bmatrix} x \\ y \end{bmatrix}$$

and a transformed point as

$$\vec{P'} = \begin{bmatrix} x' \\ y' \end{bmatrix}.$$

We left-multiply a point vector, $\vec{P}$, by a transformation matrix, M, to acquire a new point vector, $\vec{P'}$, such that $\vec{P'} = M\vec{P}$.

We can translate points to new positions by adding distances, d, to the coordinates of the original points. We define a translation vector, T, such that

$$T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

and the transformed point $\vec{P'} = \vec{P} + T$. where

$$x' = x + d_x$$

and

$$y' = y + d_{y.}$$

Points can be scaled (stretched or shrunk) in the x and y directions by a scaling matrix, S, such that

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

and

$$x' = x \cdot s_x$$

and

$$y' = y \cdot s_{y.}$$

Points can be rotated around the origin by an angle, θ, with a rotation matrix, R, where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

and

$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$

and

$$y' = x \cdot \sin\theta + y \cdot \cos\theta .$$

Any and all of the transformations can be applied multiple times and in succession. For example, we can translate a point, scale it, rotate it and translate it again with

$$\vec{P'} = T2 + (R \cdot S \cdot (T1 + \vec{P})) .$$

Suppose we have the following object as shown in Figure 2.3 (Foley et al., 1997)

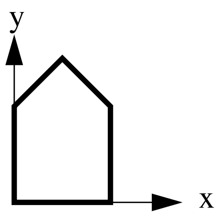


Figure 2.3: Object to be transformed

.

The series of transformations described above might appear as shown in Figure 2.4.

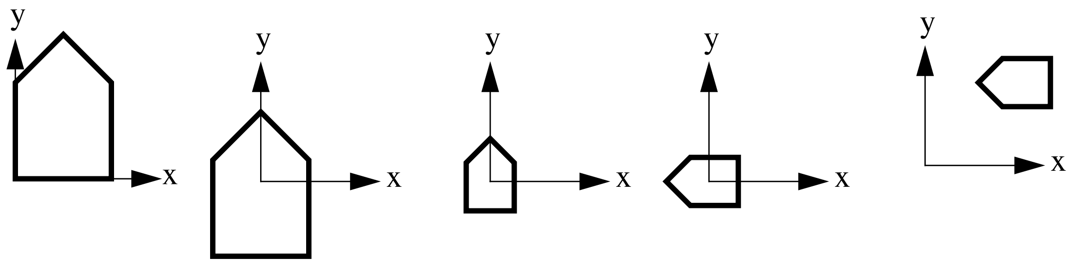


Figure 2.4: Series of transformations on an object

2.1.2.2 Homogeneous Coordinates

Notice that rotation, scaling and shearing are all multiplicative transforms while translation is an additive transform. We would like to be able to treat these transformations consistently to simplify transformation combinations. Homogeneous coordinates allow us to apply all four transformations multiplicatively. In homogeneous coordinates, we add a third coordinate in 2D space, w. Now a single point, $\vec{P}$, is represented as a three-element column vector where

$$\vec{P} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \text{ where } w \neq 0 \text{ and typically } w = 1.$$

When we add an "extra" dimension to the 2D coordinates, every 2D point $[x, y]^{\mathrm{T}}$ in 3D space represents a point along a line that passes through $[x, y, w]^{\mathrm{T}}$ where we want the specific point $[x_1, y_1, w_1]^{\mathrm{T}}$ where $w_1 = 1$. If w does not equal 1, we simply divide all three elements by w to force $w = 1$.

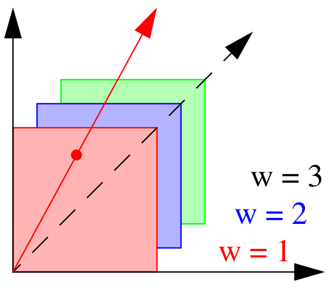


Figure 2.5: 2D homogeneous coordinate space

Any two sets of points $[x_i, y_i, w_i]^{\mathrm{T}}$ and $[x_j, y_j, w_j]^{\mathrm{T}}$ represent the same point if one is a multiple of the other. Dividing P_i and P_j by w_i and w_j respectively will result in the same coordinates $[x, y, 1]^{\mathrm{T}}$. Points at $w = 0$ are points at infinity.

In 2D homogeneous coordinates, we have the following transformation matrices for translation, scaling and rotation:

$$T = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now we can translate a point, scale it, rotate it and translate it again with

$$\vec{P'} = T2 \cdot R \cdot S \cdot T1 \cdot \vec{P}.$$

2.1.2.3 3D Transformations

We extend the idea of homogeneous coordinates to 3D space and represent a point as a 4-element vector, $\vec{P}$, such that

$$\vec{P} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \text{ where } w \neq 0 \text{ and typically } w = 1.$$

and we have the following transformation matrices for translation and scaling:

$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Rotation around a three-coordinate axis is described in the following section.

2.1.3 Representing and working with orientation and rotations

We represent orientation using a rotation from some known orientation just like we represent positions as a translation from some known position origin. The important difference between orientation and position is that orientation space is wrapped on itself in a way that linear position space is not. For example a rotation about the X axis of 45 degrees will produce the same orientation as a rotation of 405 degrees about the X axis. To make matters more confusing there are also combinations of rotations about Y and Z that can produce the same final orientation.

This wrapped nature of orientations is a constant source of difficulty when we attempt to implement even simple operations such as linear interpolation and filtering. We must be prepared to deal with apparent discontinuities in orientation that are really not discontinuities at all but rather differing ways to get to the same place.

Rotation Matrix

The most commonly used representation for rotations is the rotation matrix as described in the previous section on 3D transforms. The upper 3 by 3 sub matrix of the 4 by 4 homogenous rotation transform is a 3D rotation matrix. Its inverse is equal to its transpose and its determinant is 1. We can usefully interpret the columns of a rotation matrix as the new coordinate axes projected onto the old (or origin) coordinate axes.

The rotation matrix is the representation of choice for transforming points because only a simple and efficiently implemented matrix multiply is required. On the other hand, they are inappropriate for filtering or interpolation. Addition and subtraction of elements will almost certainly result in a matrix that is not a proper rotation.

Euler Angles

General rotations in 3D can be expressed as three successive rotations about different axes. For example, a transformation from reference axes to a new coordinate frame may be expressed as follows:

$$\begin{aligned}
\text{rotation } \psi \text{ about } z \text{ axis, } R_1 &= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\text{rotation } \theta \text{ about } y \text{ axis, } R_2 &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \\
\text{rotation } \phi \text{ about } x \text{ axis, } R_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}
\end{aligned} \tag{2.1}$$

Finally, the full transformation can be expressed as the product of these three separate transformations.

$$R = R_3 R_2 R_1$$

$$R = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \sin\phi\cos\psi\sin\theta - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \sin\phi\cos\theta \\ \cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi & \cos\phi\sin\psi\sin\theta - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.2}$$

Notice the asymmetry in the above matrix in relationship to the three angles. The order of the composition matters because matrix multiplication is not commutative. The mathematics is trying to tell us that the axes interact. Unfortunately the three Euler angles don't indicate this to us at all. If we attempt to filter or interpolate the three angles independently we are ignoring exactly this critical interaction.

Yaw, Pitch, and Roll

People often used the words *yaw*, *pitch*, and *roll* to refer to orientations about a self-referenced coordinate frame. (Historically these terms have been used in navigation such as on ships and in planes.) Specifically if you are sitting upright, looking straight ahead, *yaw* would refer to rotating your head to the left or right around the axis of your neck and spine, *pitch* would refer to elevating or declining your chin up or down, and *roll* would refer to leaning your head toward one shoulder or the other. In other words if you place a right-handed coordinate system at the base of your head such that the Z axis is up and you are looking down the Y axis, yaw, pitch, and roll would correspond to rotation about the Z, X, and Y axes respectively.

Quaternions

Hamilton (Hamilton, 1853) invented quaternions to enable division of vectors. You can find excellent introductions to quaternions in a SIGGRAPH paper by Shoemake (Shoemake, 1985) and in the book "Quaternions and Rotation Sequences" by Kuipers (Kuipers, 1998). We have also included (Vicci, 2001) in Appendix C.

A quaternion Q consists of a vector augmented by a real number to make a four-element entity. It has a real part Q_r and a vector part Q_v. If Q_r is zero, Q represents an ordinary vector; if Q_v is zero, it represents an ordinary real number. A unit quaternion has the sum of the squares of its four elements equal to 1.

Unit quaternions represent rotations. The vector part of the quaternion specifies the axis of rotation. The real part of the quaternion is the cosine of half the rotation angle. Thus, a quaternion $\{Q_r, Q_v\}$ represents a rotation of $2\operatorname{acos} Q_r$ about the axis Q_v following the right-hand rule.

From the above, we can see that the identity rotation is represented by the quaternion $\{1, [0, 0, 0]\}$ which specifies a rotation of 0 degrees about an unspecified axis. Here are some other simple rotations:

$$\text{90 degrees about Y} = \left\{\frac{1}{\sqrt{2}}, \left[0, \frac{1}{\sqrt{2}}, 0\right]\right\},$$

$$\text{270 degrees about Z} = \left\{\frac{-1}{\sqrt{2}}, \left[0, 0.\frac{1}{\sqrt{2}}\right]\right\}.$$

Quaternion addition is accomplished simply by adding like parts; real part to real part and vector part to vector part.

Multiplication of quaternions $P = QR$ is defined as

$$\{P_r, P_v\} = \{Q_r R_r - Q_v \cdot R_v, Q_r R_v + R_t Q_v + Q_v \otimes R_v\} \tag{2.3}$$

This equation says that the real part of the result is the product of the real parts minus the inner product of the vector parts. The vector part of the result is the real part of Q times the vector part of R, plus the real part of R times the vector part of Q, plus the cross product of the vector parts. Quaternion multiplication composes the rotations of the two quaternions.

The inverse of a quaternion has the same real part and the negative of the vector part.

In order to rotate a point S by a quaternion Q we evaluate QSQ^{-1} where the multiplication is quaternion multiplication. In this multiplication, think of the vector S as the vector part of a quaternion with zero real part. Though it isn't obvious, this triple product will always result in a quaternion with a zero real part. The vector part will be the rotated point.

To convert a quaternion $\{w, [x, y, z]\}$ to an equivalent rotation matrix we can evaluate:

$$R = \begin{bmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy \\ 2xy-2wz & 1-2x^2-2z^2 & 2yz+2wx \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 \end{bmatrix} \tag{2.4}$$

Watch out when you are given four numbers said to represent a quaternion rotation; there is no agreement on the order of the elements. Some systems specify the real part followed by the vector part, others specify the real part last. It is impossible to tell by examining the four elements which is the real part unless the rotation is known.

The unit quaternions can be thought of as points on a 4D sphere. Each point represents a rotation. Each point would represent a unique orientation except for the difficulty that points connected by a line through the center of the sphere (that is Q and -Q) represent the same rotation. In a time sequence of quaternions we sometimes see apparent jumps that are actually just a result of this ambiguity. When filtering or interpolating in a sequence it is important to handle these apparent discontinuities.

Interpolations among quaternions are properly carried out with spherical interpolation on the 4-sphere (Shoemake, 1985). Linear operations are equivalent to moving along a chord of the sphere rather than on the surface. In a small region, the sphere appears to be flat so the locally linear operations are a good approximation. Just be sure to renormalize the quaternions that result. Don't attempt linear operations on quaternions that are far apart in orientation. Vicci (Vicci, 2001) introduces a new approach to averaging rotations and orientations represented as quaternions.

Small Euler Angles

When the angles are very small, the Euler angle rotation matrix above takes on a particularly simple and useful form. For small angles (expressed in radians) $\sin\alpha \rightarrow \alpha$ and $\cos\alpha \rightarrow 1$. The sine approximation has relative error of about 0.5% at 10 degrees and the error decreases quadratically as the angle gets smaller. The relative error of the cosine approximation is about 3 times that in the sine approximation. The rotation matrix in equation (2.2) reduces with the small angle approximation to

$$R = \begin{bmatrix} 1 & \psi & -\theta \\ \psi & 1 & \phi \\ \theta & -\phi & 1 \end{bmatrix}$$

In this form, the angles behave linearly allowing simple interpolation and filtering. This has become our representation of choice. As described in (Welch & Bishop, 1997), we represent orientation with two terms, a global orientation represented as a quaternion and a local perturbation of that orientation represented as small Euler angles. All filtering operations are done on the small angles making use of their local linearity. The filtered angles are used to update the global orientation so that the small angle property is preserved.

Small Euler angles also have a simple relationship to quaternions. For small angles the approximate quaternion is:

$$Q = \{1 - \phi^2 - \theta^2 - \psi^2, [\phi, \theta, \psi]\}$$

2.1.4 Coordinate System Transforms

Let M_{A_B} denote a transform (scale, rotation, translation) from coordinate system B to coordinate system A. We represent points as column vectors, therefore $\vec{p}_A = M_{A\_B} \cdot \vec{p}_B$ is the transform of the point $\vec{p}_B$ in coordinate system B, by M_{A_B}, to point $\vec{p}_A$ in coordinate system A. The composition of two transforms is given by $M_{A\_C} = M_{A\_B} \cdot M_{B\_C}$ and will transform a point in coordinate system C into coordinate system A. Figure 2.6 (Robinett & Holloway, 1994) shows a diagram of a point $\vec{p}$ and its coordinates in coordinate systems A and B.

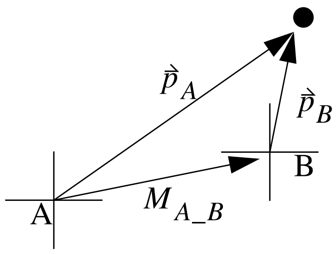


Figure 2.6: Transformation M_{A_B}

We need a sequence of transformations to express the target position in the world coordinate system, a *Head_World* transform. As described in (Robinett & Holloway, 1994), the primary function of the Head_World transform is to contain the measurement made by the tracker of head position and orientation, which is updated each display frame as the user's head moves around. The tracker hardware measures the position and orientation of a small movable sensor with respect to the fixed tracker coordinate frame.

The two components of tracker hardware, the tracker's base and the tracker's sensor, have native coordinate systems associated with them by the tracker's hardware and software. If the tracker base is bolted onto the ceiling of the room, this defines a coordinate system for the room with the origin up on the ceiling and with the X, Y, and Z axes pointing whichever way it was mechanically convenient to mount the tracker base onto the ceiling. In a VR environment, the sensor is mounted somewhere on the rigid structure of a head-mounted display, and the HMD inherits the native coordinate system of the sensor.

So, the Head_World Transform is actually comprised of a series of transforms from World to Tracker Base to Head Sensor to Head. The transformation is decomposed into

$$M_{H\_W} = M_{H\_HS} \cdot M_{HS\_TB} \cdot M_{TB\_W} .$$

We can transform to the user's eyes or any other point in the user's coordinate system with an additional transformation such as M_{LE_H} to transform from the head to the left eye.

2.2 Probability and Random Variables

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

2.2.1 Probability and Random Variables

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

$$P(A \cup B) = P(A) + P(B) . \tag{2.5}$$

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:

$$P(A \cap B) = P(A)P(B) . \tag{2.6}$$

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

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{2.7}$$

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 t (time is the sample space) $X(t)$ 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*:

$$F_X(x) = P(-\infty, x]. \tag{2.8}$$

This function represents the cumulative probability of the continuous random variable X for all (uncountable) events up to and including a. Important properties of the cumulative distribution 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 (2.8) is its derivative, known as the *probability density function*:

$$f_X(x) = \frac{d}{dx}F_X(x). \tag{2.9}$$

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 (2.5), for continuous random variables one integrates the probability density function over the interval of interest.

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

$$\text{Expected value of } X = E(X) = \sum_{i=1}^{n} p_i x_i \tag{2.10}$$

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

$$\text{Expected value of } X = E(X) = \int_{-\infty}^{\infty} x f_X(x)dx . \tag{2.11}$$

Finally, we note that equation (2.10) and equation (2.11) can be applied to *functions* of the random variable X as follows:

$$E(g(X)) = \sum_{i=1}^{n} p_i g(x_i) \tag{2.12}$$

and

$$E(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x) dx . \tag{2.13}$$

The expected value of a random variable is also known as the *first statistical moment*. We can apply the notion of equation (2.12) or (2.13), 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

$$E(X^k) = \int_{-\infty}^{\infty} x^k f_X(x) dx . \tag{2.14}$$

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

$$E(X^2) = \int_{-\infty}^{\infty} x^2 f_X(x) dx . \tag{2.15}$$

When we let $g(X) = X - E(X)$ and apply equation (2.15), 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

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

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 proven 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* (Brown & Hwang, 1996; Maybeck, 1979). Finally, the normal distribution has some nice properties that make it mathematically tractable and even attractive.

The normal distribution is characterized by the following probability density function:

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x - m_x)^2\right]$$

where the expected value is

$$m_x = \int_{-\infty}^{\infty} x f_X(x) dx$$

and the squared-variance is

$$\sigma^2 = \int_{-\infty}^{\infty} (x - m_x)^2 f_X(x) dx .$$

Graphically, the normal distribution is what is likely to be familiar as the "bell-shaped" curve shown below in Figure 2.7.

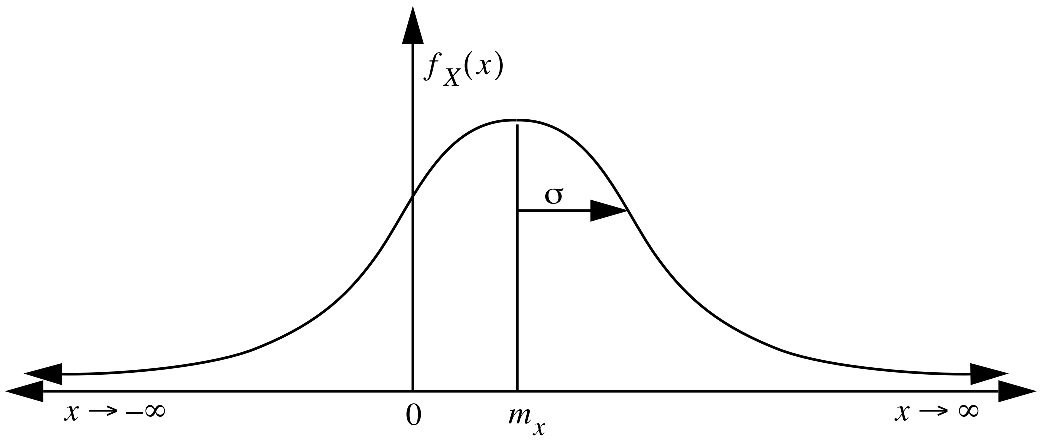


Figure 2.7: The Normal or Gaussian probability distribution function.

Independence and Conditional Probability for Continuous Variables

As with the discrete case and equations (2.6) and (2.7), 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

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

In addition, Bayes' rule follows from (2.7), 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

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

Spatial vs. Spectral Signal Characteristics

In the previous section 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

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)] \tag{2.16}$$

for sample times t_1 and t_2. If the process is *stationary* (the density is invariant with time) then equation (2.16) depends only on the difference $\tau = t_1 - t_2$. In this common case the autocorrelation can be re-written as

$$R_X(\tau) = E[X(t)X(t+\tau)]. \tag{2.17}$$

Two hypothetical autocorrelation functions are shown below in Figure 2.7. 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.

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.

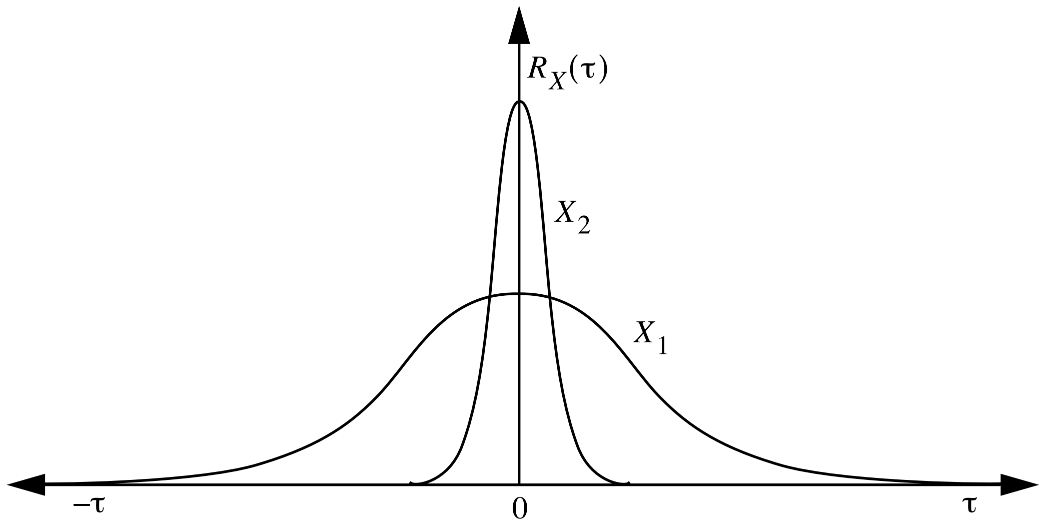


Figure 2.8: Two example (hypothetical) autocorrelation functions X_1 and X_2.

White Noise

An important case of 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 2.9. This is in fact a description of *white noise*, which be thought of both as having power at all

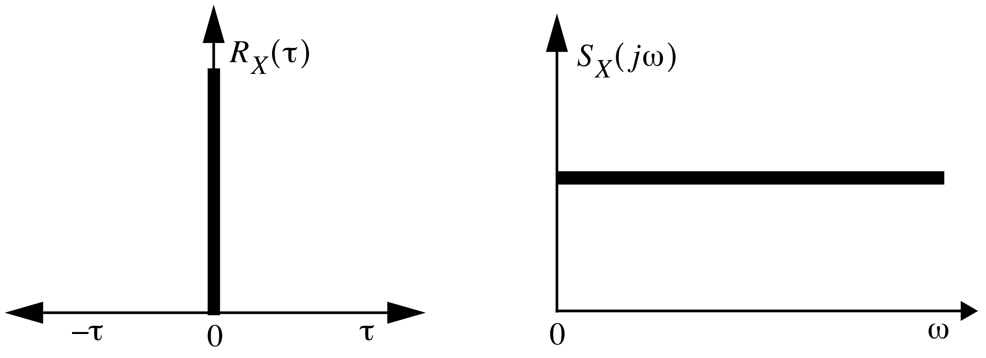


Figure 2.9: White noise shown in both the time (left) and frequency domain (right).

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. Classifications of Devices and Systems

There are many dimensions to the design space of tracking and motion capture systems, and you can consider the design in the context of any one or more of these dimensions. We like to think about this classification as "many ways to slice the problem." For example, there is the dimensionality of the information provided by the sources and sensors; the geometric arrangement of the sources and sensors; whether they offer absolute or relative references; passive vs. active; signal to noise ratio; accuracy; resolution; bandwidth; latency; update rate; reliability/repeatability; required infrastructure, and on and on. Here we look primarily at two classifications: by physical medium and by geometric configuration of the devices. We then end this chapter by considering a combination of mediums in *hybrid systems*.

3.1 By Physical Medium

Here we describe the most practical (hence common) physical mediums employed in tracking and motion capture. We thank Leandra Vicci (UNC-Chapel Hill) for her valuable contributions to this section.

3.1.1 Acoustic Tracking

Acoustic trackers typically use ultrasonic sound waves to sense range. Ultrasonic waves have a frequency above the audible range of the human ear of approximately 20,000 [Hz].

3.1.1.1 The Geometry

A single transmitter/receiver pair provides a distance measurement of the target from a fixed point. In the absence of further information, this defines a sphere on whose surface the target is located. A shown in Figure 3.1 (Oceanographers, 2001), the addition of a second receiver or transmitter restricts this surface to the circle of intersection between the two spheres. A third receiver or transmitter restricts this circle to two points, one of which can normally be rejected, and determines a 3D position. Therefore, either three transmitters and one receiver or three receivers and one transmitter are required to find 3D position. To estimate position and orientation, three transmitters and three receivers are required.[1]

1. The reader might be familiar with GPS navigation units in which *four* satellites are used for position estimation. The fourth satellite is used to constrain timing differences in the other three.

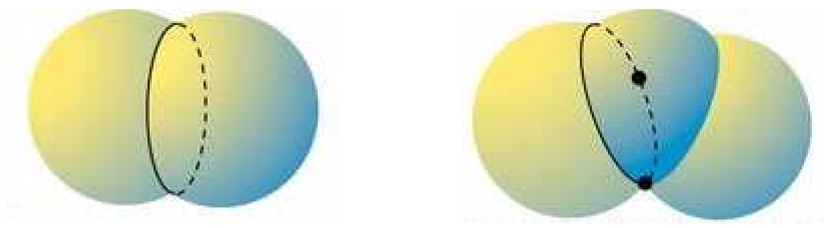


Figure 3.1: Intersection of two spheres (a circle) and three spheres (two points)

3.1.1.2 Techniques

Acoustic trackers typically employ one of two techniques to determine position and orientation:

1. Time of Flight (TOF), and
2. Phase Coherence.

In both methods, the speed of sound is used to convert the time to distance. The drawbacks to any time-of-flight or phase difference method is the inherent delay in waiting for the signal to travel from the source to the destination and this is exaggerated by the slow speed of sound. At 0˚ C the speed of sounds in air is 331 [m/s], approximately 1.1 [ft/ms]. To complicate matters, the speed of sound varies with temperature and pressure and cannot be treated as a constant. More generally, the speed of sound in a gas is

$$speed = \sqrt{\frac{\gamma RT}{M}}$$

where γ is a thermodynamic constant of air, R is the ideal gas constant, M is the molecular weight and T is absolute temperature.

Time of Flight (TOF)

The TOF method measures the time t it takes for an ultrasonic pulse to travel from a transmitter to a receiver to provide an absolute distance, d. It takes the time required for a sound wave to travel form the source to the destination and multiplies that by the speed of sound v to get distance.

$$d[m] = v\left[\frac{m}{s}\right] \times t[s]$$

An absolute position is estimated from this distance.

Phase Coherence

The phase coherence method measures the phase difference between the sound wave at the receiver and the transmitter to provide a change in distance, δ. All signals can be represented as a sum of sinusoidal functions of the form $A\cos(\omega t - \phi)$ where ϕ is the phase shift of the signal and A is its amplitude. From this phase difference between the emitted and received signals of a known wavelength or frequency, distance can be computed.

Figure 3.2 below depicts two cosine waves with amplitude of 1 and frequency of 1 Hz. The solid curve is $\cos(x)$ with no phase shift, and the dashed curve is $\cos(x - \pi/4)$ where $\pi/4$ is the phase shift. A phase angle of 360° or 2π [radians] is equal to one cycle.

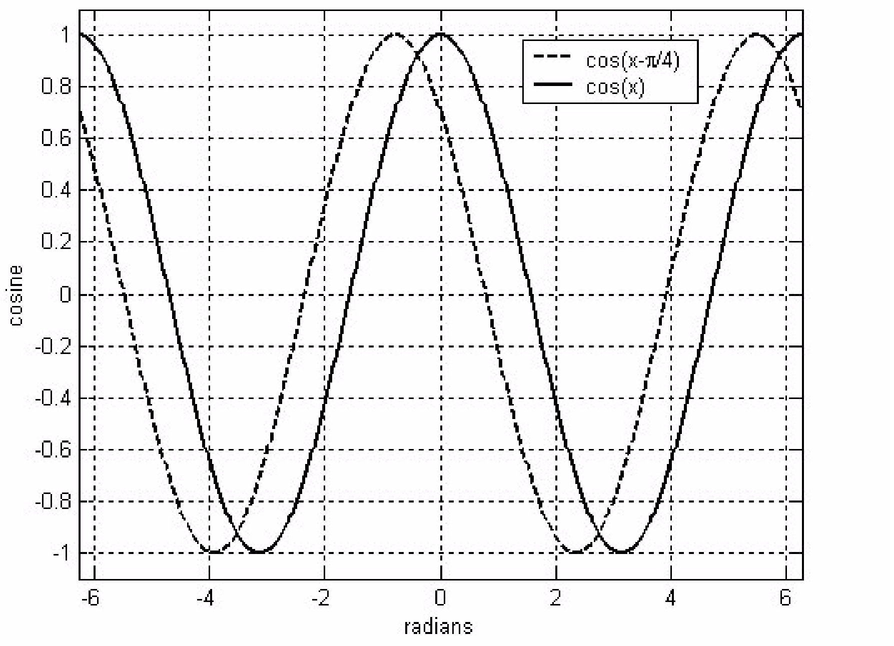


Figure 3.2: Cosine functions with phases of 0 and $\pi/4$ [radians]

The frequency of an acoustic wave, f [Hz], where [Hz] is [1/s], is related to its speed, c [m/s], and wavelength, λ [m], by the equation $c = \lambda f$ where c varies with temperature and pressure as described previously. The fraction δ of the wavelength λ corresponding to the phase shift can be computed by

$$\delta[\mathrm{m}] = \lambda[\mathrm{m}] \cdot \frac{\phi_{delay}[\mathrm{radians}]}{2\pi[\mathrm{radians}]}$$

$$= \frac{c[\mathrm{m/s}]}{f[\mathrm{Hz}]} \cdot \frac{\phi_{delay}[\mathrm{radians}]}{2\pi[\mathrm{radians}]}$$

If the speed of sound is 331 [m/s] and, as in the graph above, the transmitted signal had zero phase and the received signal had a phase of $\pi/4$ [radians], we calculate a phase difference of $+\pi/4$ [radians] and know that the signal traveled

$$\begin{aligned}\delta[\mathrm{m}] &= \frac{c[\mathrm{m/s}]}{f[\mathrm{Hz}]} \cdot \frac{\phi_{delay}[\mathrm{radians}]}{2\pi[\mathrm{radians}]} \\ &= \frac{331[\mathrm{m/s}]}{1[\mathrm{Hz}]} \cdot \frac{\frac{\pi}{4}[\mathrm{radians}]}{2\pi[\mathrm{radians}]} \\ &= 331[\mathrm{m}] \cdot \frac{1}{8} \\ &= 41.375[\mathrm{m}].\end{aligned}$$

If we assume an ultrasonic frequency commonly used in acoustic tracking of 40 [kHz] with the same measured phase shift (1/8 of a cycle), we find

$$\begin{aligned}\delta[\mathrm{m}] &= \frac{c[\mathrm{m/s}]}{f[\mathrm{Hz}]} \cdot \frac{\phi_{delay}[\mathrm{radians}]}{2\pi[\mathrm{radians}]} \\ &= \frac{331[\mathrm{m/s}]}{40[\mathrm{kHz}]} \cdot \frac{1}{8} \\ &= 1.034[\mathrm{mm}].\end{aligned}$$

Notice that a phase of $\phi + (n \cdot 2\pi)$ looks just like a phase of ϕ at the receiver, resulting in an ambiguity in distance. This is usually resolved by assuming phase changes are small between measurement updates. For example, given an acoustic frequency of 40 [kHz], we calculate a wavelength of 8.275 [mm] and measure a phase difference of 0.125 as above. Without previous position information, we cannot determine whether the target is at a distance of 0.125 wavelengths, 17.125 wavelengths or $n + 0.125$ wavelengths. However, if we know the current position estimate is at 100 wavelengths, 0.8275 [m], we assume that the new position is at the closest wavelength count to the current count, 100.125 wavelengths. Therefore the target has moved $\delta = 1.034[\mathrm{mm}]$ and the new position estimate of the target is at

$$\begin{aligned}d' &= d + \delta \\ &= 0.8275[\mathrm{m}] + 1.034 \times 10^{-3}[\mathrm{m}] \cdot \\ &= 0.82753[\mathrm{m}]\end{aligned}$$

or, simply

$$\begin{aligned}d' &= 100.125[\mathrm{wavelengths}] \cdot 8.275\left[\frac{\mathrm{mm}}{\mathrm{wavelength}}\right] \\ &= 828.53[\mathrm{mm}] \\ &= 0.82753[\mathrm{m}].\end{aligned}$$

Note that acoustic energy diminishes with the square of the distance between the transmitter and receiver.

3.1.1.3 Commercial and Research Products

Commercial products that employ acoustic sensors include Infusion Systems' FarReach, Intersense's IS-600 Mark 2 and Mark 2 PLUS (inertial hybrid).

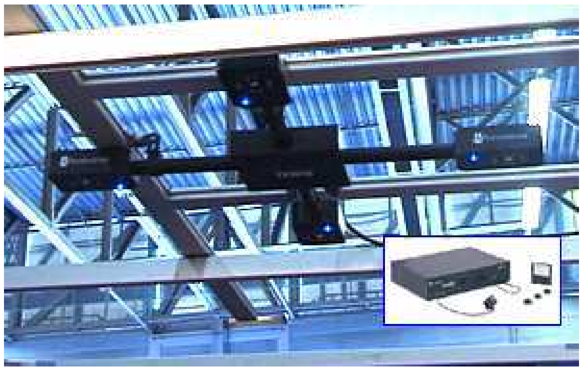


Figure 3.3: Intersense's IS-600 Mark 2

Other acoustic trackers include M.I.T.'s Lincoln Wand (1966). See also (Mulder, 1994b) for more examples.

3.1.2 Inertial Tracking

Inertial trackers use accelerometers to measure the acceleration for object position and gyros to measure the orientation of object orientation. They are passive, relying on Newton's second law of motion, $F = ma$, and its rotational equivalent $M = I\alpha$, which means there are no physical limits on the working volume and the user is able to move around unencumbered in the environment. Ideally, both are deployed in orthogonal triples (for 3D position in *x*, *y* and *z* and 3D orientation in *roll*, *pitch*, and *yaw*) in order to estimate 6D pose.

3.1.2.1 Accelerometers

Accelerometers actually measure the force exerted on mass since acceleration cannot be measured directly. This measured force, F, for a given mass, m, is transformed into a measure of acceleration, a, by the relationship $F = ma$. The primary transducer of an accelerometer converts acceleration into displacement. Since

$$a = \frac{d^2 r}{dt^2},$$

position r is calculated as

$$r = \iint a \, dt^2 .$$

Accelerometers use a known mass (sometimes called the proof-mass) attached to one end of a damped spring. The other end of the spring is attached to the accelerometer housing. When there is no acceleration imposed upon the accelerometer, the spring is at rest and exhibits zero displacement.

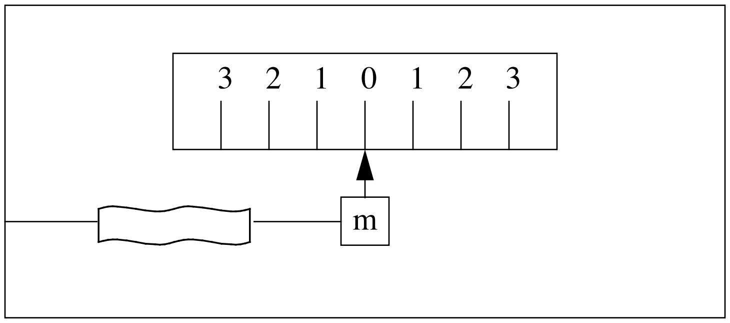


Figure 3.4: Spring at rest with zero displacement.

If a force is applied to the housing, the housing will accelerate but inertia causes the suspended mass to lag behind, resulting in a displacement.

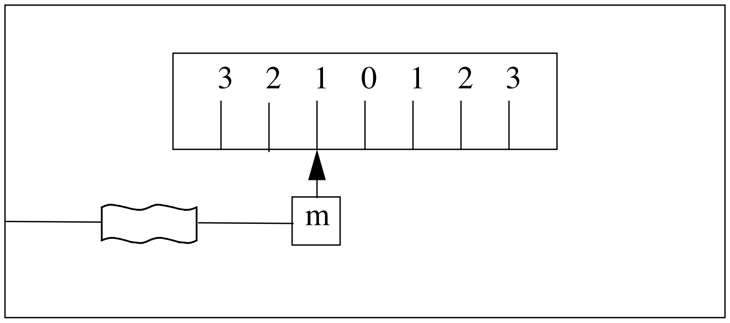


Figure 3.5: Spring under acceleration with displacement.

The displacement of the mass and extension/compression of the spring is proportional to the acceleration of the housing or, in the case or head tracking, the acceleration of the wearer.

A secondary transducer converts this displacement into a usable signal. Two common such transducer types are potentiometric and piezoelectric. Potentiometric devices attach the displacement of the mass to the slider of a potentiometer. The output voltage of a potentiometer is linearly proportional to the slider position. Voltage varies directly with current and the electrical resistance supplied by the potentiometer by $V = IR$ where V is voltage, I is current and R is resistance. Piezoelectric devices attach the displacement to a piezoelectric element (a piezoelectric crystal that produces an electric charge when a force is exerted upon it) that generates a voltage proportional to the displacement.

3.1.2.2 Gyroscopes

Gyroscopes employ the principle of conservation of angular momentum. If torque is exerted on a spinning mass, its axis of rotation will precess at right angles to both itself and the axis of exerted torque. If the mass spins very fast, it will have a large angular momentum that will strongly resist changes in direction.

Figure 3.6 illustrates the phenomenon of precession. If we fix the axis of rotation on one end of the gyroscope allowing it to pivot around this point, the force of gravity simply causes the gyroscope to fall down in the direction of the gravity vector if it is not spinning. However if the gyroscope is spinning, gravity exerts a torque on the gyroscope about an orthogonal axis to the axis of rotation. If the gyroscope is spinning at a sufficient rate, the gyroscope does not fall in the direction of gravity. Instead it rotates around the axis that is orthogonal to both the gyroscope's axis of rotation and gravity's torque axis.

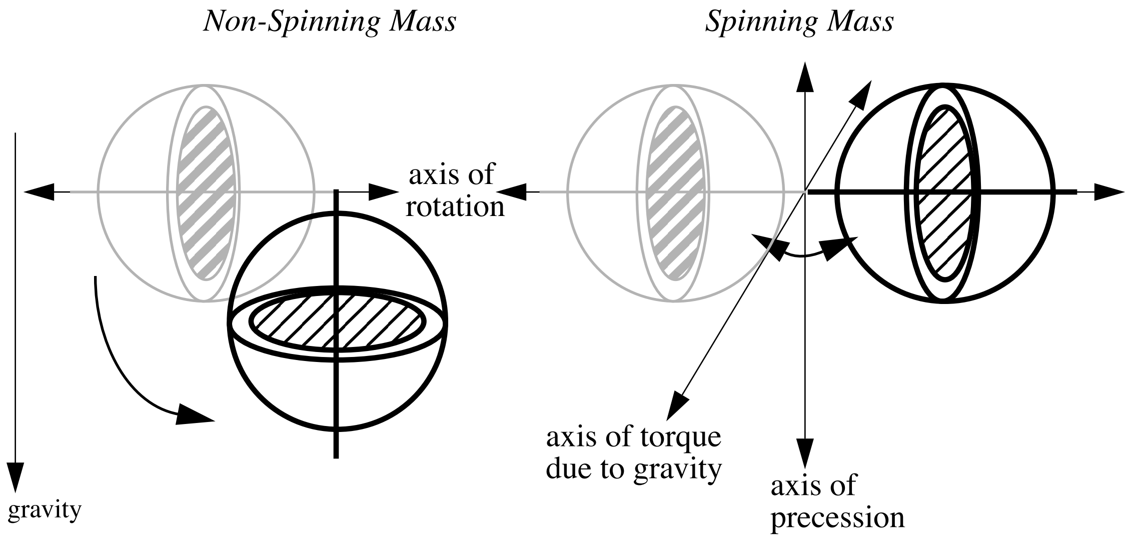


Figure 3.6: Precession

If the spinning mass is mounted on gimbals, these principles can be used to measure changes in direction. The angle that the gyroscope makes with it's housing (gimbal deflection) is a measure of the angular momentum or angular velocity. If the gimbals are constrained with springs, the rate of change of direction can be measured. This configuration is known as a rate gyro. Examples of mechanical rate gyroscopes are shown in Figure 3.7 (Britannica, 1994).

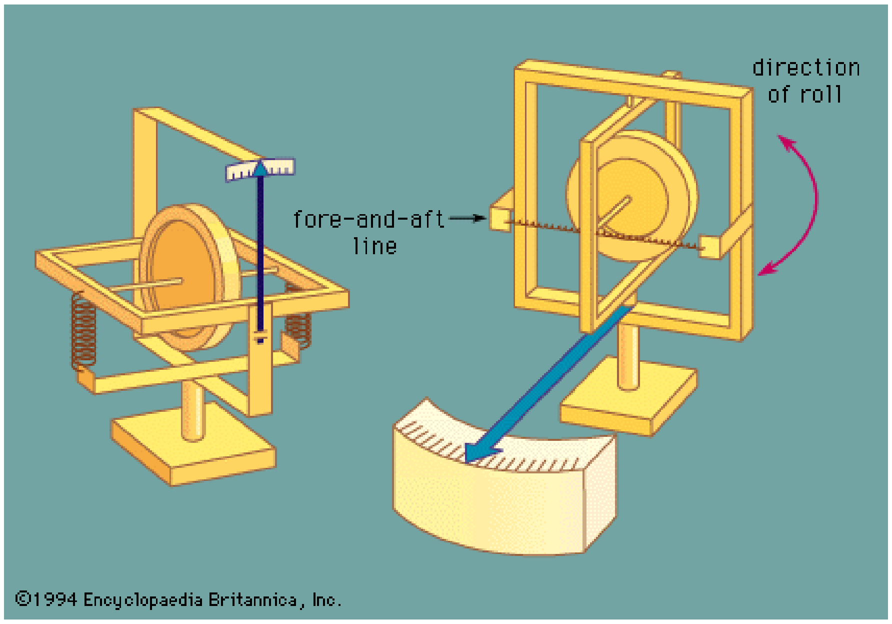


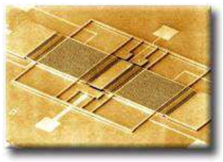
Figure 3.7: Rate Gyroscopes for measuring rate of turn (left) and rate of roll (right)

For 3D orientation (roll, pitch and yaw), three rate gyroscopes are typically fitted to a platform with their axes mutually perpendicular. Two of the gyroscopes provide for horizontal stabilization of the platform--an essential requirement to eliminate the influence of accelerations due to gravity--while the third is responsible for the north-south alignment. Pitch, roll, and yaw are detected by the three gyroscope input axes. The gimbal deflection of each of the gyroscopes is converted into a signal.

The physics supporting inertial measuring devises are most easily illustrated using mechanical devices. While extremely accurate, these devices are not used for human tracking because of their size and mass. Instead micromechanical devices are often employed. An example of is BEI Systron Donner Inertial Division’s GyroChip that uses a vibrating piezoelectric quartz tuning fork to sense rate.

Figure 3.8 illustrates the micromechanical gyro designed as an electronically-driven resonator. When an angular rate is applied to its drive tines, Coriolis or torsional forces are exerted on its drive tines which cause its vibrations to couple to the pickup tines. The pickup fork vibrations are detected and used to measure the angular rate.

Draper Labs' MEMS
Tuning Fork Gyro



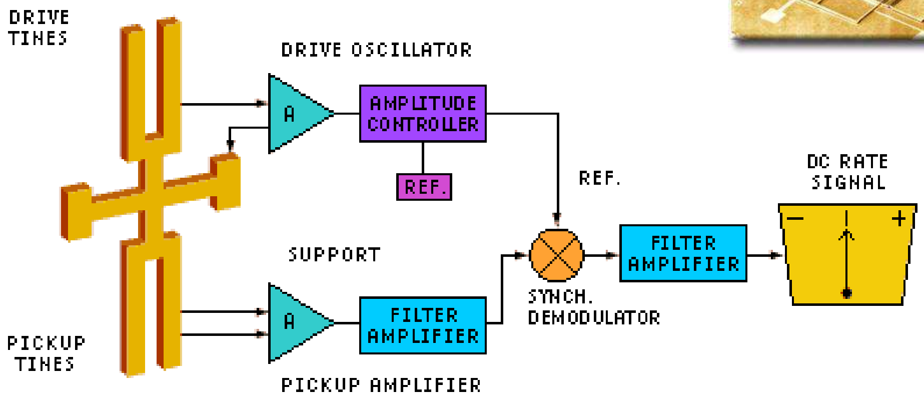


Figure 3.8: BEI Systron Donner Inertial Division's GyroChip technology

Systron Donner (Donner, 2001) offers the following explanation.

> The piezoelectric drive tines are driven by an oscillator to vibrate at a precise amplitude, causing the tines to move toward and away from one another at a high frequency. This vibration causes the drive fork to become sensitive to angular rate about an axis parallel to its tines, defining the true input axis of the sensor.
>
> Vibration of the drive tines causes them to act like the arms of a spinning ice skater, where moving them in causes the skater's spin rate to increase, and moving them out causes a decrease in rate. For vibrating tines ("arms"), an applied rotation rate causes a sine wave of torque to be produced, resulting from "Coriolis Acceleration," in turn causing the tines of the Pickup Fork to move up and down (not toward and away from one another) out of the plane of the fork assembly.
>
> The pickup tines thus respond to the oscillating torque by moving in and out of plane, causing electrical output signals to be produced by the Pickup Amplifier. Those signals are amplified and converted into a DC signal proportional to rate by use of a synchronous switch (demodulator) which responds only to the desired rate signals.
>
> The DC output signal of the GyroChip is directly proportional to input rate, reversing sign as the input rate reverses, since the oscillating torque produced by Coriolis reverses phase when the input rate reverses.

Regardless of the technology, both accelerometers and gyros provide derivative measurements. Linear accelerations must be integrated twice and angular rates need to be integrated once to derive position and orientation, respectively. This integration causes these inertial measurements to be sensitive to drift. Position error diverges with time as errors accumulate. This drift can be combated with periodic recalibration by the use of another tracking method that corrects the accumulated error periodically. Conversely inertial tracking performance is very good at high frequency and over short time intervals.

Models useful in explaining the frequency characteristics of an inertial system are shown in Figure 3.9. The dashed box marked “User” contains a question mark to indicate that the user’s acceleration (motion) is unknown. The dashed box marked “Accelerometers” shows acceleration measurement noise $\varepsilon_{a(t)}$ being summed with the ideal user acceleration signal, and the sum then being integrated twice to obtain a position estimate. The lower part of the figure shows the corresponding transfer function coefficients for spectral (frequency) analysis.

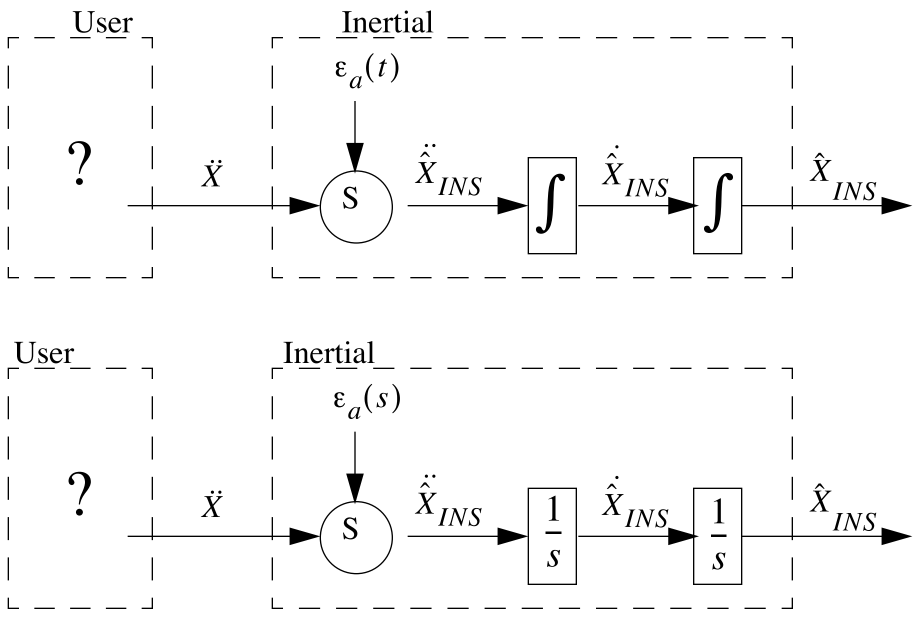


Figure 3.9: Integration in time domain and division in frequency domain of inertial measurements.

The user’s true acceleration is their position twice differentiated, i.e. the user’s acceleration is weighted by the square of the frequency (s) of their motion. This acceleration signal is then weighted by the inverse square of the frequency (s) as it is integrated twice in the inertial system to obtain a position estimate. The end result is a unity frequency weighting of the position in the final estimate.

From Figure 3.9 we also see that any electrical noise $\varepsilon_{a(t)}$ incurred during the measurement of the accelerometer output is weighted solely by the inverse square of the frequency (s) as it is integrated twice in the inertial system. The end result is an inverse square frequency weighting of electrical measurement noise in the final position estimate.

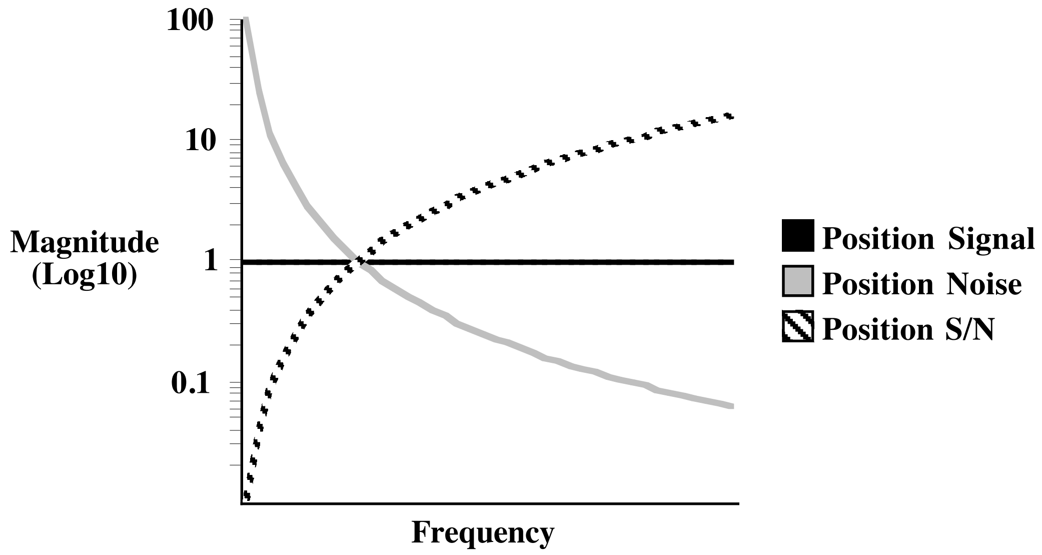


Figure 3.10: Logarithmic plots of typical accelerometer-based position signal vs. noise (for constant velocity).

In Figure 3.10, the estimated position signal, noise, and signal-to-noise ratio for an inertial system using accelerometers are plotted together against frequency. This figure demonstrates why the practical application of a solely inertial-based tracking system is impractical. At low frequencies the position estimate noticeably diverges as measurement noise is erroneously interpreted as acceleration. The most common sources of low frequency noise are the unavoidable and often time-dependent random “DC” (or very low frequency) biases. Such bias errors can cause an inertial-based tracker to report that a subject is moving even when that subject is completely still, or conversely to report that a subject is still when in fact they are slowly moving. The result is that in general for inertial devices to be practical they must be combined with some other mediums as described in Section 3.3.

An additional source of error is misalignment with the gravity vector whose effects must be subtracted out of inertial measurements. The effect of gravity can be significant. One degree of tilt error over ten seconds can cause nine meters of position error.

Noise and quantization error in the signals from the inertial sensors is another important source of error. Figure 3.11 shows how error accumulates in inertial systems. The curves are contours of time to 0.1 [m] of accumulated error for accelerometer (vertical axis) and rate gyro (horizontal axis) signals with the indicated number of useful bits. For example, with 15 bits of useful dynamic range on the rate gyro signal and about 12 bits of useful range on the accelerometer signal, the graph predicts about 20 seconds of operation before 10 [cm] of error accumulates. Increasing the range of the accelerometers will not improve the system performance because the rate gyros are the limiting factor. The shape of the curves can be explained by realizing that rate gyro errors will result in misestimation of the system tilt. As explained earlier, tilt errors cause fractions of the gravity vector to be integrated as actual acceleration resulting in potentially large position errors. Region 1 includes inertial units readily available today. Region 2 includes today's peak performance. Region 3 reflects arguably unachievable performance.

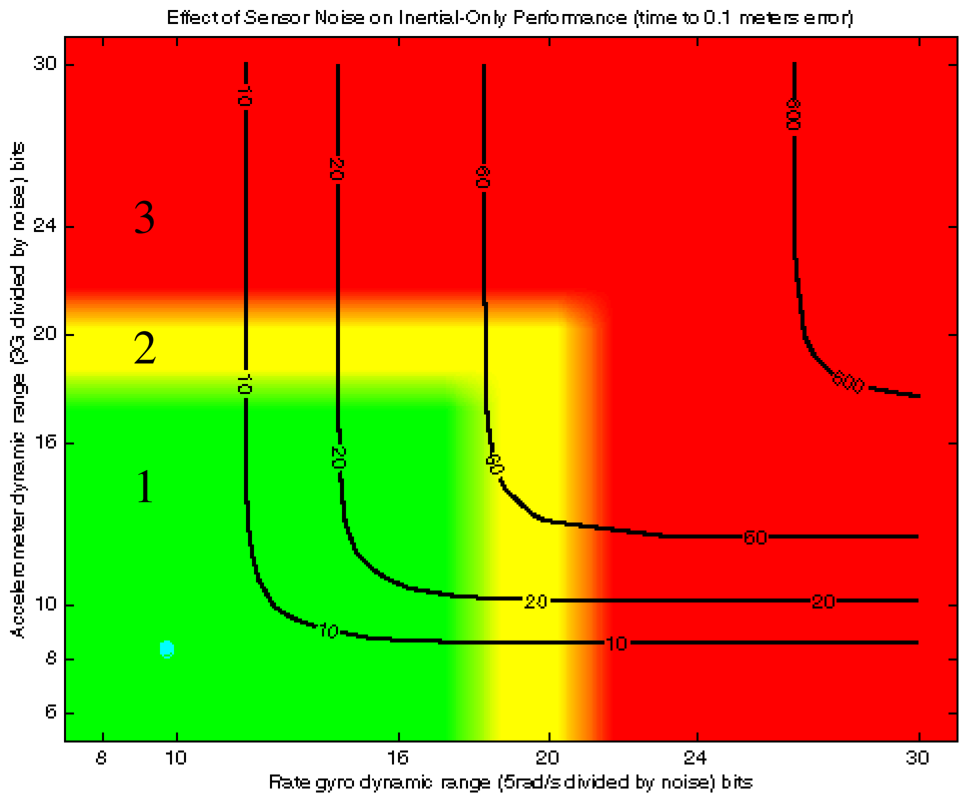
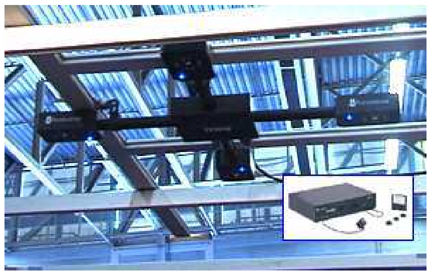


Figure 3.11: Time to 0.1 [m] error

3.1.2.3 Commercial and Research Products

Inertial trackers have become more common over the last few years due to IC technologies allowing for significant reduction in size. Because they require some form of periodic calibration to control drift they are typically used in hybrid tracking products. Commercial products that employ inertial sensors Ascension's 3D-BIRD, Intersense's IS-300/600 and InterTrax 2. See also (Mulder, 1994b) for more examples.

Ascension's 3D-BIRD



Intersense's IS-300/600

Intersense's InterTrax

Figure 3.12: Some example inertial tracking systems.

3.1.3 Magnetic Tracking

Magnetic trackers use magnetic fields to measure range and orientation. These magnetic fields can be low frequency AC fields or pulsed DC fields and three orthogonal triaxial coils are used at both the transmitter and receiver to produce position and orientation measurements.

3.1.3.1 Magnetic Fields

Generating magnetic fields

Current carrying coils are used to generate the source magnetic fields. The magnetic field produced by a circular coil of wire carrying a current, I, at a distance d and off-axis angle, θ, is described by

$$H_r = \frac{M}{2\pi d^3}\cos(\theta) \qquad \text{(radial component)}$$

$$H_\phi = \frac{M}{4\pi d^3}\sin(\theta) \qquad \text{(tangential component in } \theta \text{ direction)}$$

$$H_\phi = 0 \qquad \text{(tangential component in } \phi \text{ direction)}$$

where H_r and H_θ are the radial and tangential components of the field, M is the magnetic moment of the loop ($M = NIA$), A and N are the area enclosed by the current loop and number of the turns of the loop or winding (Raab, Blood, Steiner, & Jones, 1979) and I is the current. Figure 3.13 illustrates the magnetic field of a single-turn winding.

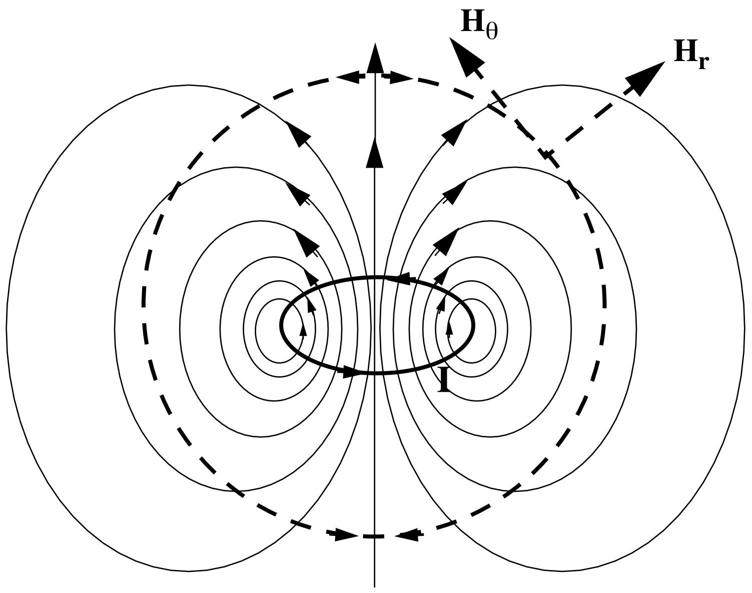


Figure 3.13: Magnetic Dipole

Detecting magnetic fields

A time varying magnetic field will induce a voltage in a coil that can be measured electrically. The magnitude of the voltage is proportional to the area circumscribed by the coil, the rate of change of the field, and varies as $\cos\theta$ where θ is the angle between the direction of the field lines and the axis of the coil.

3.1.3.2 System Configuration

A magnetic tracking system consists of a transmitter and a receiver in the form of coils. A 1D sensor for estimating the position in z (i.e. direction of gravity) is made up of a single coil transmitter oriented in the z-direction. When current is applied to the coil a magnetic field is generated. At the receiver, this induces a maximum voltage proportional to the sensed magnetic field strength in a receiving coil oriented in the same direction as the field.

The induced voltage level provides information about the both distance from the transmitter to the receiver and the axis-alignment between them. Boundaries of equal accuracy are found along a hemisphere or sphere around the transmitter. In Figure 3.14

(Burdea & Coiffet, 1994) the accuracy and amplitude $A2$ is less than $A1$ where the radius $R2$ is greater than $R1$. Here the radius (distance form the transmitter) is the determining factor.

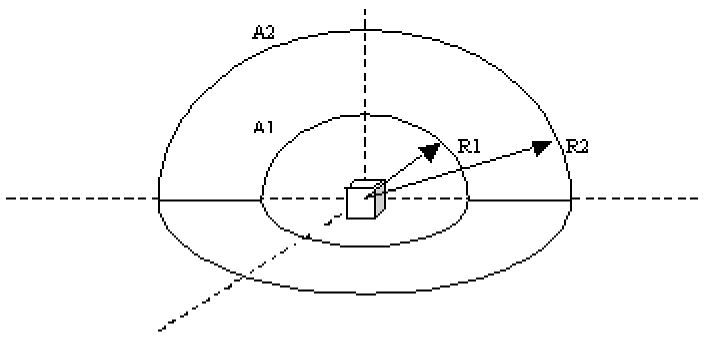


Figure 3.14: Accuracy contours around a coil

Three separate coils wound orthogonally around a core are used to generate and measure the magnetic field strength in x, y and z. When three orthogonal coils are used, the three source coils are activated serially and the induced signal in each of the three receiving coils is measured. A full measurement cycle contains three measured values for each of the three source coils and this nine-element measurement is used to calculate the position and orientation of the receive coils relative to the source coils. The signal strength per receive coil decreases cubically with distance and with the cosine of the angle between its axis and the local magnetic field direction. The strength of the induced signals can be compared to the known strength of the transmitted signals to find distance. The strength of the induced signals are compared to each other to find orientation.

One disadvantage of AC magnetic sensors is that ferromagnetic and other conducting objects within the sensor space can distort the magnetic field geometry. An eddy current is induced in conducting materials by the source magnetic field (and other fields such as the Earth's magnetic field) and these currents produce small magnetic fields around the conducting materials. The fields cause distortions in the source fields shown above resulting in erroneous pose estimates. This is particularly a problem when using AC transmitters because of the continuously varying nature of AC signals.

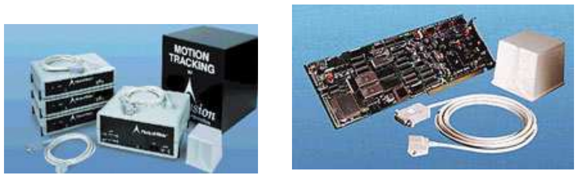
The use of DC transmitters overcomes the eddy current interference problem. Eddy currents are generated only at the beginning of a measurement cycle and a steady state can be reached where the effect of interfering magnetic fields is minimized as the eddy current values approach zero. However, distortions due to ferromagnetism, mainly in steel or iron objects, are still a problem for DC systems.

3.1.3.3 Research and Commercial Products

Despite some shortcomings, magnetic tracking systems have historically enjoyed popularity as a result of a small user-worn component and relative ease of use. Commercial products that employ magnetic sensors include Polheumus' 3SPACE FASTRAK and ISOTRAK II (AC electromagnetic) and Ascension's Flock of Birds and PcBIRD (Pulsed-DC electromagnetic). See also (Mulder, 1994b) for more examples.

Polheumeus' Stylus Option

Polheumeus' Long Ranger Tracker


Ascension's Flock of Birds and PcBIRD transmitter

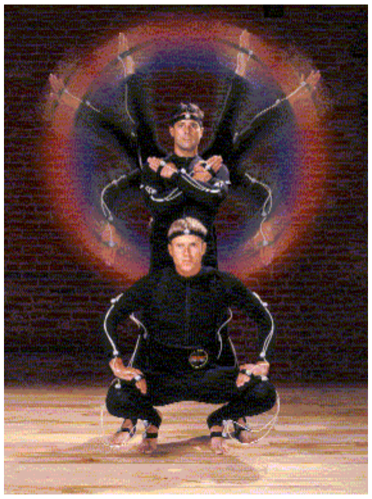

Polheumeus' Star*Trak

Figure 3.15: Examples of magnetic tracking systems.

3.1.4 Mechanical Tracking

Mechanical trackers measure joint angles and lengths between joints. Given one known position, all other absolute positions can be derived from the relative joint measurements. They are used to measure all parts of the body and have been historically employed in motion capture. In implementation, they range from whole body suits that measure the position of all the major joints to mechanical arms to gloves that measure the location of the hands and fingers.

Mechanical tracking systems can be ground-based in which one point of the tracker is affixed to the floor at a known location. This limits the user's range of motion. They can also be body-based in which the system is attached only to the user, typically in the form of an exoskeleton. This does not limit physical range of motion but can be prohibitive if the suit is heavy or bulky.

The rotations and lengths can be measured by gears, potentiometers, and bend sensors as shown in Figure 3.16.

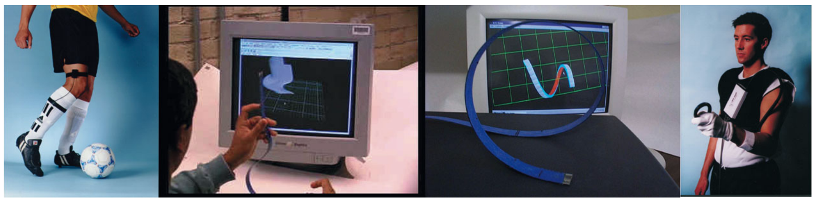
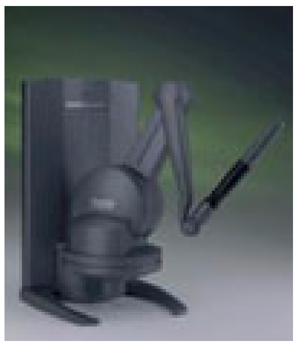


Figure 3.16: Mechanical tracking sensors

A potentiometer is a device that transduces a rotation or displacement to a voltage. A bend sensor is typically a thin strip of plastic whose resistance changes as it bends. The more it bends, the higher the resistance. Alternatively, optical fiber can be treated for a short distance on one side to lose light proportional to the angle through which it is bent.

An advantage of mechanical trackers is the elegant addition of force feedback as illustrated in Sensable's Phantom and the EXOS dexterous hand master pictured in Figure 3.17.



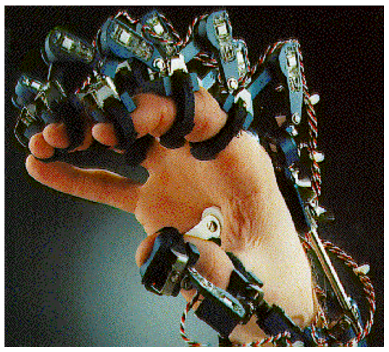
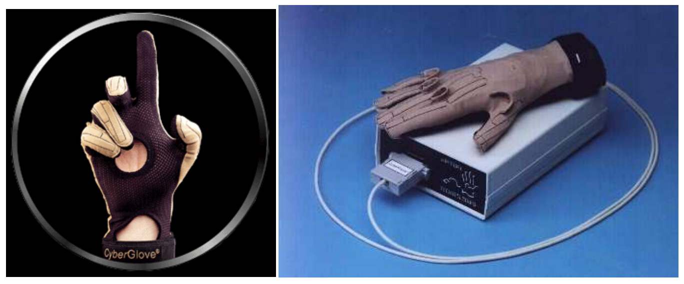


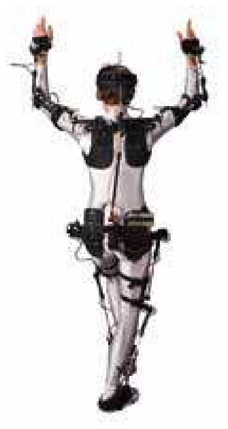
Figure 3.17: Sensable's Phantom and EXOS dexterous hand

They typically provide good accuracy and low latency but can be cumbersome for the user. The user may be constrained by the suit or arm and, therefore, may not have full freedom of movement.

3.1.4.1 Research and Commercial Products

Commercial products that employ mechanical sensors are Fakespace's Boom HF, Virtual Technologies' CyberGlove and MetaMotion's Gypsy (motion capture only). See also (Mulder, 1994b) for more examples.


Virtual Technologies' CyberGlove


MetaMotion's Gypsy

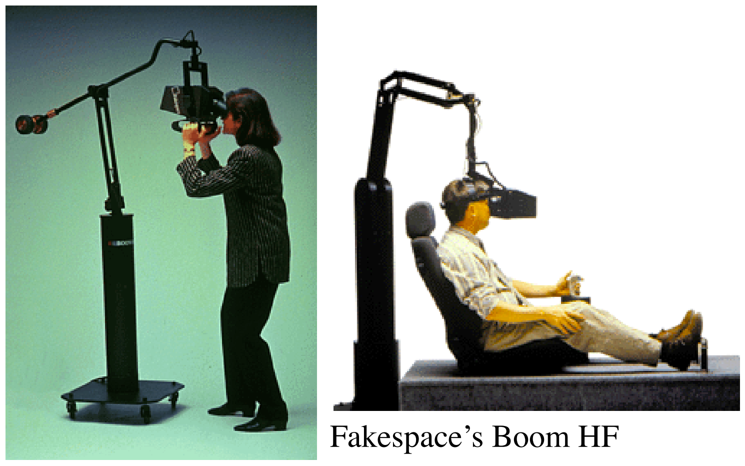

Fakespace's Boom HF

Figure 3.18: Some example mechanical tracking systems.

3.1.5 Optical Tracking

Optical trackers use light to measure angles. As shown in Figure 3.19, a single point on the detector, an optical sensor or image plane, provides a ray defined by that pixel and the center of projection, C. As is the case with the acoustic medium, optical energy diminishes with the square of the distance between the transmitter and receiver.

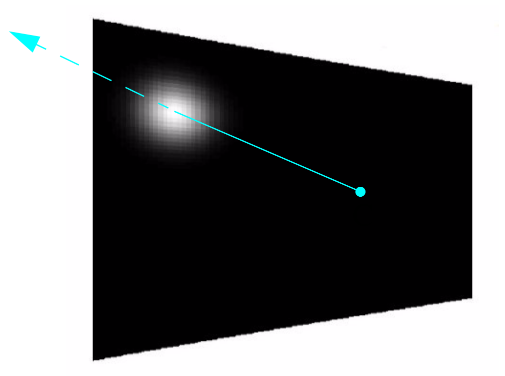


Figure 3.19: Centroid of light.

3.1.6 Targets

Optical tracking systems (also called image-based systems) can use two types of targets: active targets and passive targets. Active targets are powered such as an light-emitting diode (ILED). Infrared LEDs (ILEDs) are used to combat noise due to ambient light. Passive targets are not powered such as reflective materials or high contrast patterns. Regardless of categorization, a detector is used to record the object being tracked (the target) and from this angle measurement, position and orientation can be derived. Some systems use no artificial targets and, instead, use elements of the natural scene.

3.1.7 Detectors

Detectors can be simple video and CCD cameras (typically used with passive targets), or lateral-effect photodiodes (typically used with active targets) that provide the location of the centroid of light on the image plane as illustrated in Figure 3.19. Video cameras and CCDs require imaging techniques to determine position while photodiodes produce currents that are directly proportional to the light center's position.

3.1.7.1 Lateral Effect PhotoDiodes (LEPDs)

1D LEPDs place two terminals on either side of a silicon photosensitive region. An incident light beam produces electrons that flow laterally towards the terminals on either side of the region. The amount of current measured at each terminal is dependent on the distance of the centroid of the incident beam from the terminals. If the centroid occurs at the center of the region, equal current values will be measured at each terminal. A 2D LEDP sensor is made up of 2 1D sensors rotated 90 degrees from each other. LEPDs can be used to detect both the intensity and position of an incident light beam.

3.1.7.2 Quad Cells

Quad cells are made up of four photosensitive cells. When a light beam is incident upon a quad cell, a current is generated in each of the four quadrants proportional to the amount of light seen by each. A perfectly circular beam illuminating the exact center of the quad cell will produce equal photocurrents in the four quadrants. As shown in Figure 3.20, if the beam is too small such that it falls in between quad cell or too large such that is covers all four cells, there is no information to be gained from the voltage measurements of each cell.

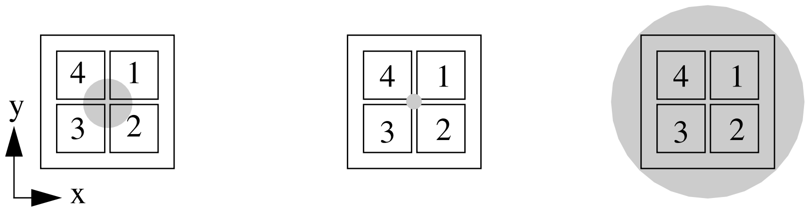


Figure 3.20: A Quad Cell

The x and y displacements of the beam relative to the center of the quad cell can be calculated using the following formulas:

$$x = \frac{(i_1 + i_2) - (i_3 + i_4)}{i_1 + i_2 + i_3 + i_4}$$

$$y = \frac{(i_1 + i_4) - (i_2 + i_3)}{i_1 + i_2 + i_3 + i_4}.$$

3.1.7.3 Charge Coupled Devices (CCDs)

A CCD array can be a 1D or 2D collection of light-sensitive cells.

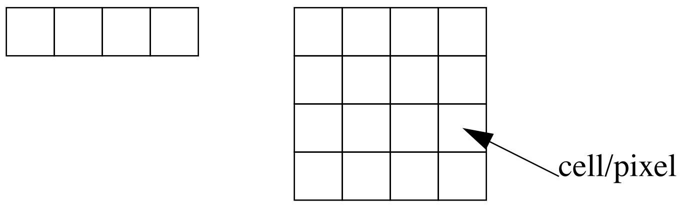


Figure 3.21: 1D and 2D CCD Detector Arrays

When light is incident upon a CCD cell, electrons are produced and each cell accumulates electrons proportional to the amount of incident light for the duration of the dwell time. The electron count per array cell is read out as a voltage to provide a per pixel luminance value. An “empty” cell corresponds to zero volts (black). This collection of pixel values produces a digital image that is similar to the analog images captured with film. The array of luminance values can be analyzed to pinpoint the cell (or pixel) of highest intensity and sub-pixel accuracy can be achieved by interpolation between discrete pixel values. The dwell time dictates how long the CCD will accumulate a charge and must be set large enough so that sufficiently high SNR is achieved for pinpointing the target. However, care should be taken in fixing the dwell time because it affects the update rate of the CCD. Long dwell times delay measurements which affect the rate at which position estimates can be determined.

While optical sensors fundamentally provide angle measurements, they can also be used to determine range. The amount of defocus (i.e. blur) due to the limited depth of field of lenses can provide information about distance. Structures which are closer to the plane of focus for an image will appear sharper. Conversely, structures farther from the plane of focus will appear more blurred. As shown in Figure 3.22 (from (Goshtasby, 2001; Wood et al., 2000)) there is ambiguity between two points that lie equidistantly on either side of the plane of focus. This ambiguity can be eliminated with the addition of a second image plane.

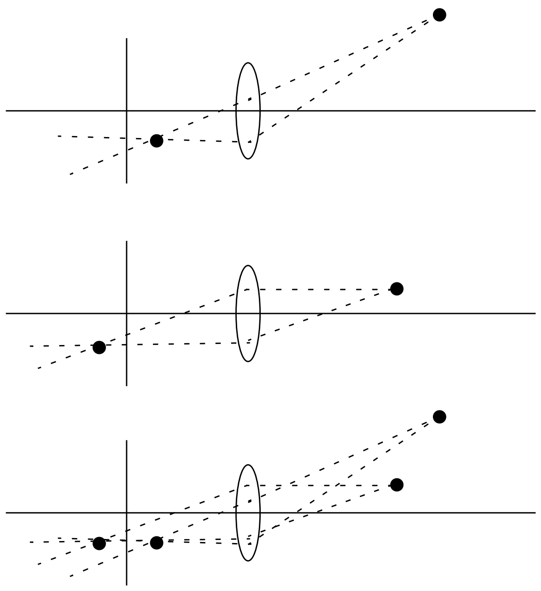


Figure 3.22: Two objects produce the same position and blur on the image

In general, optical tracking systems exhibit high accuracy and resolution and are well suited to real-time systems in terms of update rate (light is a fast medium). However, there is much variety among optical tracking systems due to technical specification difference (focal length, FOV, etc.), system configuration (outside-in, inside-out) and target type. This accuracy depends on a clear line of sight between the sensor and the target. If something in the environment or the object itself blocks this line-of-sight, optical tracking systems suffer from obscuration difficulties.

In Section 4.1 we describe some traditional approaches to using optical sensors for pose estimation. In Appendix C we have also included a copy of (Welch et al., 2001), which describes the UNC-Chapel Hill HiBall optical tracking system.

3.1.7.4 Commercial and Research Products

Commercial products that employ optical sensors include InMotion's CODA, 3rdTech's HiBall-3000, University of Iowa's Selspot II, Arcsecond's Vulcan, Ascension's laserBIRD, and Phoenix Technologies' Visualeyez (motion capture only). Other optical trackers include Omniplanar's Virtual Tracker and the Minnesota scanner. See also (Mulder, 1994b) for more examples.

3.2 Sensor Configurations

Choosing the physical medium and sensors for a tracking system determines only a part of its capability. The geometric configuration of the sources and sensors also has a profound effect. For example, in an optical tracker we can have fixed sensors observing moving targets or moving sensors observing fixed targets. This has lead to the descriptions "outside looking in" and "inside looking out" as described in (Welch et al., 2001), but the direction of "looking" is not the determining factor. Rather, as we shall see, the distinguishing factor is the coordinate frame in which the measurements are made.

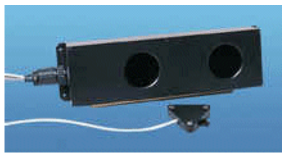
3.2.1 Measurements in the Laboratory Frame

The CODA system (BL, 2000) is a commercial example of a system that makes its measurements in the laboratory coordinate frame as illustrated in Figure 3.24. It uses three or more stationary 1D optical sensors that observe LED beacons that are free to move (a typical "outside looking in" configuration). Each of the 1D sensors narrows the possible location of the LED to a plane in three space. The three planes intersect in a mathematical point which is the system's estimate of the three-dimensional coordinates of the LED. The FlashPoint 5000 (IGT, 2000) from Image Guided Technologies uses the same fundamental measurement strategy though the sensors and optics are quite different. The Selspot and OPTOTRAK (NDI, 2001) systems are also essentially the same though they use at least two 2D sensors (each determining a line) rather than at least three 1D sensors.

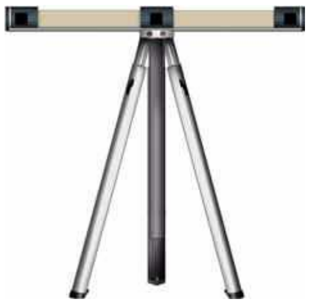
The Minnesota Scanner (Sorensen, Donath, Yang, & Starr, 1989) uses spinning mirrors at fixed locations in the laboratory to project planes of light into the working volume. The tracked target is a photo-detector that detects the time at which it is illuminated by the swept plane of light. The time is used along with the precisely known rotation rate of the mirror to determine the equation of a plane that passes through the detector and the center

InMotion Systems' CODA

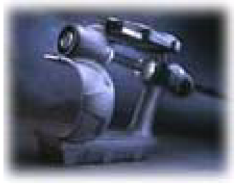
3rd Tech's HiBall-3000



Ascension's LaserBIRD



Phoenix's Visualeyez



Arcsecond's Vulcan

Figure 3.23: Some example optical tracking systems.

of rotation of the mirror. Just as in the CODA system, the multiple planes are then intersected to produce an estimate of the 3D coordinate of the target. The commercially available ArcSecond system works in the same way.

Even though the sensors and sources have apparently swapped places in the CODA mxp30 and the Minnesota Scanner, they are both fundamentally making an angle measurement in the laboratory coordinate system. Thus, while the moving sensor on the Minnesota Scanner is "looking out", it shares all the characteristics of a "outside-looking-in" system. What matters is the coordinate frame in which the measurements are made.

The positional sensitivity of the systems above is determined by the angular resolution of the optical sensors, the distance between target and sensor, and on the geometric configuration of the sensors. The angular resolution is determined by the field of view of

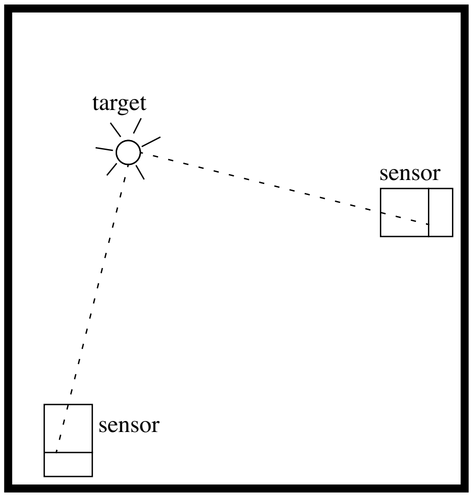


Figure 3.24: Two sensors are shown fixed in lab space. They observe a moving target. Here in 'Flatland' the position of the image on the 1D image-line of the sensor determines the equation of a line that passes through the target. The intersection of two of these lines determines the 2D position.

the sensor and the resolution of measurements on the image plane. The designer of such a system always has a trade off between positional accuracy and working volume; higher accuracy requires a smaller working volume while larger working volume implies lower accuracy.

As shown in Figure 3.25 below, when the geometric configuration of the system is such that the sensors are physically close together compared to the distance to the target, the planes (or lines) they determine are nearly parallel. This results in near singularity of the resulting equations and poor positional accuracy in the direction aligned with the planes. This problem is aggravated by uncertainty in the sensor estimates. Rather than being planes or lines, the constraint provided by a single sensor is better modeled by a cone or a wedge. For maximum positional accuracy the sensors should be far apart and at nearly right angles to one another. Unfortunately placing the sensors far apart may give rise to line-of-sight problems because all sensors must have a clear view of each target.

In order to determine orientation, multiple targets must be arranged in a rigid configuration. Then the relative positions of the targets can be used to derive orientation. The sensitivity of the resulting system to small rotations is determined both by its positional sensitivity and the distance between targets. Orientation sensitivity is maximized by increasing the distance between targets but this can quickly result in a physically unwieldy device.

3.2.2 Measurements in the User's Frame

The HiBall Tracker (Welch et al., 1999, 2001) uses a golf-ball sized cluster of six 2D optical sensors looking out at LED beacons fixed in the environment. In the HiBall, angular measurements are made in the moving coordinate system of the user as shown in Figure 3.26, rather than in the fixed coordinate system of the lab.

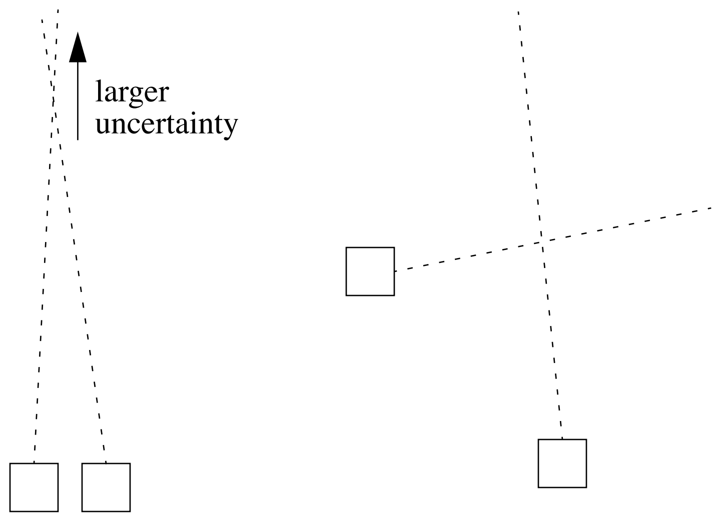


Figure 3.25: When sensors are close together as on the left, the equations of the lines they determine are nearly parallel resulting in less positional sensitivity in the direction parallel to the lines When the sensors are far apart and at nearly right angles the sensitivity is greatest.

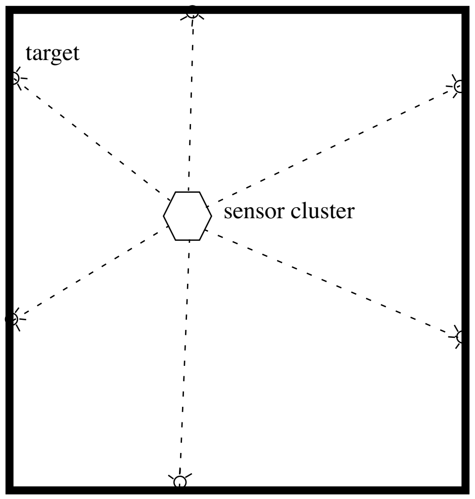


Figure 3.26: A cluster of six sensors observes targets that are at fixed locations in 'Flatland'. Each observation determines a constraint between the position of the moving cluster and its orientation. A minimum of three observations are required to determine the position and orientation but to ensure uniqueness of the solution and to allow for measurement errors more are better.

Unlike the systems above for which one sighting determines a mathematically simple constraint such as a plane or line equation, sighting one target with one of the cameras in the cluster provides a difficult to visualize constraint on the relationship between the cluster's position and its orientation. Given as few as three sightings it is possible to solve a nonlinear system of equations to determine the position and orientation of the cluster (Azuma & Ward, 1991). With only three sightings there are up to four solutions that can be arbitrarily close together (Fischler & Bolles, 1981). Larger numbers of sightings provide a unique solution and also allow the use of least-squares estimation to reduce the effects of noise on the measurements.

The HiBall does not solve such nonlinear equations to determine it position. Instead it uses the SCAAT algorithm described in Section 4.2.5.

The orientation sensitivity of the HiBall is determined by the angular sensitivity of the optical sensors in the cluster. The HiBall achieves high angular sensitivity using very narrow fields of view (approximately 6 degrees) and high-resolution analog sensors (lateral-effect photo diodes (Wallmark, 1957) with approximately 1 part in 2000 resolution). This requires that the room-mounted LEDs are densely packed to ensure that sufficient numbers are continuously visible.

The positional sensitivity of the HiBall is similar to the systems described above. It varies with distance to the target and with sensor resolution in the same way.

Comparing systems that measure in the laboratory frame with those that measure in the user's frame we see that user-centered measurements can provide higher orientation accuracy and comparable positional accuracy for systems of practical size. Systems with moving targets have the advantage that the targets are smaller and lighter and thus easier to attach to the user. Lab-based sensors are probably preferred when position is the only or most important measurement required.

3.3 Hybrid Systems

As described in Section 3.1, every type of sensor has fundamental limitations related to the associated physical medium. In addition there are practical limitations imposed by the measurement systems, and application-specific limitations related to the motion characteristics of the target being tracked. These limitations continuously affect the *quantity* and *quality* of the information. The result is that no single medium or sensor type provides the necessary performance over the wide spectrum of temporal and spatial characteristics desired for many applications. Happily several mediums exhibit complementary behavior, and these systems can be combined to leverage the strengths of each medium as needed. Systems that employ such mixed mediums are called *hybrid systems*.

A number of research and commercial groups have recognized that hybrid systems are necessary for some applications, and have constructed hybrids that combine multiple sensors including inertial, video, and GPS (Azuma, 1995b; Azuma & Bishop, 1994; Azuma et al., 1998; Azuma, Hoff, & Neely, 1999; Azuma, Lee et al., 1999; Behringer, 1999; Foxlin, 1996; Foxlin & Durlach, 1994; Foxlin et al., 1998; Golding & Lesh, 1999; Pasman, van der Schaaf, Lagendijk, & Jansen, 1999; Verplaetse, 1996; Verplaetse, 1997; You, Neumann, & Azuma, 1999a, 1999b).

Inertial Tracking

One of the most popular technologies (mediums) used to improve or *augment* the performance of other mediums is to incorporate accelerometers and gyros in some form of an *inertial navigation system* as in (Azuma, 1995b; Azuma & Bishop, 1994; Azuma, Hoff et al., 1999; Azuma, Lee et al., 1999; Emura & Tachi, 1994b; List, 1983). The reason is that inertial navigation systems exhibit relatively low error at high frequencies and velocities, and are very responsive. This is due in part to the fundamental medium, and in part to the nature of inertial devices and our ability to sample their signals at a relatively high rate, typically on the order of thousands of samples per second. (Contrast this with magnetic, acoustic, and optical system can typically only be sampled hundreds of times per second.) Unfortunately as described in Section 3.1.2, they also exhibit high error at lower frequencies and velocities. At low velocities (very slow or no movement) one must in practice contend with pronounced bias and drift error (noise). As movement slows, such noise begins to grow with respect to the true signal, resulting in unbounded error growth.

Two examples of inertial hybrids are inertial-acoustic, and inertial-optical. The most well known example of the former is the commercial system described by (Foxlin et al., 1998) and marketed by Intersense (Intersense, 2000). This system seeks to overcome the temporal and spatial shortcomings of purely acoustic systems (Section 3.1.1) by adding a relatively fast and robust inertial system. Again, the inertial system could not be used alone, but in combination with the acoustic system one can cover a wider spectrum of motion and performance. As a side note, the system described in (Foxlin et al., 1998) actually uses *three* mediums—it uses pulsed infrared light to aid in the timing (triggering) of acoustic signals.

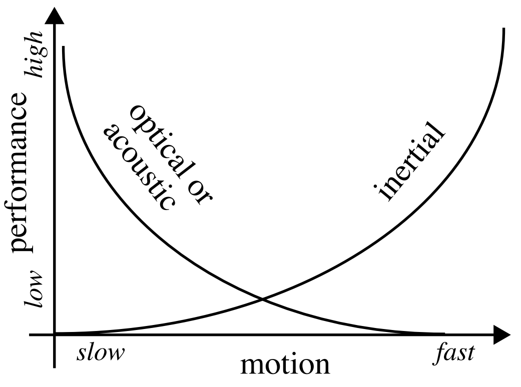


Figure 3.27: A qualitative comparison of inertial vs. optical or acoustic

A second example is an inertial-optical hybrid. As with any hybrid, the complementary behavior of each system is leveraged to obtain more accurate and stable tracking information than either system alone. With an inertial system, bias and drift errors dominate (grow unbounded) during periods of slow movement. However, during such periods the error can be controlled by an optical system, which would typically exhibit its best behavior under such conditions. Conversely an optical typically performs worst during very rapid movement, precisely the conditions where the inertial signal-to-noise ratio is high (see Figure 3.10). In addition, while a typical vision-based optical system using multi-pixel cameras *is* affected by unrelated motion in an environment, an inertial system is *not* and can provide assistance with static visual feature discrimination. Figure 3.27 offers a qualitative comparison of the complementary relationship between the performance of inertial and optical (or acoustic) mediums.

4. Approaches

4.1 Traditional Closed-Form Approaches

We get the intuition from linear algebra that we need at least as many equations (or constraints) as there are unknowns to solve a system of equations. More equations might be required for a non-linear system. In this section we will examine a few solutions to particular tracking problems using this traditional approach. In Section 4.2.5 we examine a new approach that sequentially applies a single constraint at a time.

The key thing to notice is the variety of the mathematical approaches. These are only a few of the many formulations for these problems.

4.1.1 Range Trackers

For 3D position tracking using range common arrangements are three fixed microphones and a single moving source or three fixed sources and a single moving microphone. Mathematically, we must solve three simultaneous sphere equations where we know the origin (e.g. x_0, y_0, z_0) and the radius (e.g. r_0) for each sphere.

$$\begin{aligned}(x-x_0)^2+(y-y_0)^2+(z-z_0)^2 &= r_0^2\\(x-x_1)^2+(y-y_1)^2+(z-z_1)^2 &= r_1^2\\(x-x_2)^2+(y-y_2)^2+(z-z_2)^2 &= r_2^2\end{aligned}\qquad(4.1)$$

The solution of these equations is very complicated but we can simplify it greatly by aligning the three microphones with the axes of the coordinate system as follows:

a. put microphone 0 at the origin of the coordinate system

b. put microphone 1 out the X axis by one unit

c. put microphone 2 out the Y axis by one unit

d. all three microphones are in the Z = 0 plane.

This amounts to allowing the microphones to define the coordinate system rather than attempting to embed them in an existing coordinate system. We can, of course, get from these *microphone coordinates* to laboratory coordinates with a separate linear transform. The simplified equations are now:

$$\begin{aligned} x^2 + y^2 + z^2 &= r_0^2 \\ (x-1)^2 + y^2 + z^2 &= r_1^2 \\ x^2 + (y-1)^2 + z^2 &= r_2^2 \end{aligned} \tag{4.2}$$

And their solution is:

$$\begin{aligned} x &= \frac{r_0^2 - r_1^2 + 1}{2} \\ y &= \frac{r_0^2 - r_2^2 + 1}{2} \\ z &= \pm\sqrt{r_0^2 - x^2 - y^2} \end{aligned} \tag{4.3}$$

Notice that the sign ambiguity on z prevents us from determining which side of the plane the target is on. This is a common problem with nonlinear systems of equations. The three spheres actually intersect in 2 points as shown in Figure 3.1 on page 32. Usually some design constraint is used to restrict z to the positive or negative subspace.

Another subtle point in this formulation and the others in this section is the assumption that the multiple measurements correspond to a single position of the target. For an acoustic tracker with fixed microphones and a moving source, this assumption is likely valid because the microphones respond to a single acoustic burst from the source. On the other hand, for a tracker with a moving microphone, this assumption is likely to be violated if the sources must be operated sequentially to avoid interference. In this case the microphone may have moved between successive measurements. Violating the assumption of simultaneous measurements results in measurement errors that vary with the target's rate of motion. See "The Simultaneity Assumption" on page 73 for further discussion.

4.1.2 Optical Trackers with Fixed 2D Sensors

Each camera in an optical system with 2D sensors that are fixed in laboratory coordinates determines a ray in 3D. The ray can be described using the parametric form with parameter s_1 for camera 1 and s_2 for camera 2. The parameters vary from 0 at the center of projection of a camera to infinity. The parametric equations are:

$$\begin{aligned} A_1 &= C_1 + s_1 D_1 \\ A_2 &= C_2 + s_2 D_2 \end{aligned} \tag{4.4}$$

C_1 and C_2 are the centers of projection of the cameras. D_1 is the unit-length direction vector determined by the image of the target on the image plane of camera 1. Likewise for D_2. The baseline $B = C_2 - C_1$ is the vector between the centers of projection.

These two rays in three space almost certainly do not intersect. Of course the exact two lines must intersect because they result from images of the same point in space. But errors in calibration of the cameras and in determining the imaged coordinates in each camera will result in line equations that likely do not intersect.

We want to determine the point in 3D that is closest to both rays. We do this by finding the parameter values that minimize the distances between the lines. That is, we want to minimize

$$\|(C_2 + s_2 D_2) - (C_1 + s_1 D_1)\| \tag{4.5}$$

This equation is the length of the line joining the two rays. Since the shortest line joining the two rays must be perpendicular to each of the rays, it must be true that

$$\begin{aligned} [(C_2 + s_2 D_2) - (C_1 + s_1 D_1)] \bullet D_1 &= 0 \\ [(C_2 + s_2 D_2) - (C_1 + s_1 D_1)] \bullet D_2 &= 0 \end{aligned} \tag{4.6}$$

This system of two equations in two unknowns has the solution

$$\begin{aligned} s_1 &= \frac{(B \bullet D_1) - (D_2 \bullet D_1)(B \bullet D_2)}{1 - (D_1 \bullet D_2)^2} \\ s_2 &= \frac{(D_1 \bullet D_2)(B \bullet D_1) - (B \bullet D_2)}{1 - (D_1 \bullet D_2)^2} \end{aligned} \tag{4.7}$$

The point closest to the two rays is the midpoint of this shortest line segment

$$\tilde{P} = \frac{(C_1 + s_1 D_1) + (C_2 + s_2 D_2)}{2} \tag{4.8}$$

Examining equation (4.7) we can see the source of the difficulty described in Section 3.1.5 when the sensors are close together relative to the distance to the target. In this case the direction vectors D_1 and D_2 will be nearly parallel. Thus their dot product will be nearly one and the denominator of the equations will grow very small. This will amplify the effect of small errors.

4.1.3 Optical Trackers with Fixed 1D Sensors

The following approach may also be used with 2D sensors by considering that you are given four 1D measurements rather than two 2D measurements.

By a calibration process we determine the coefficients that map the 1D sensor coordinate to a plane in 3D. The plane for sensor *i* will produce an equation of the form

$$A_i x + B_i y + C_i z = D_i \tag{4.9}$$

Where $[x, y, z]^{\mathrm{T}}$ is the 3D coordinate of the tracked point. With three such linear equations we can use standard solution techniques to solve for the position.

$$M = \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{bmatrix}$$

$$M \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^{-1} \cdot \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

With more than three sensors we can use least-squares solution methods to determine the position with the minimum squared error.

$$M = \begin{bmatrix} A_1 & B_1 & C_1 \\ \dots & \dots & \dots \\ A_n & B_n & C_n \end{bmatrix}$$

$$M \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} D_1 \\ \dots \\ D_n \end{bmatrix}$$

$$M^T \cdot M \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^T \cdot \begin{bmatrix} D_1 \\ \dots \\ D_n \end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (M^T \cdot M)^{-1} \cdot M^T \cdot \begin{bmatrix} D_1 \\ \dots \\ D_n \end{bmatrix}$$

Analogously to the 2D case described earlier, if the target is far away relative to the distance between cameras, rows of the matrix M will be very similar resulting in near singularity and amplification of small errors.

4.1.4 Optical Trackers with Moving Sensors

Azuma and Ward (Azuma & Ward, 1991) document the Space-Resection approach to solving for the position and orientation of the sensor cluster. This method was used in the first generation “Ceiling Tracker” at UNC. The HiBall tracker uses a much simpler method that will be described later.

Their method is too complicated to describe here. They set up the system of non-linear equations that described the relationships among the known 3D coordinates of the LEDs, the known 2D image-plane coordinates for the sightings of the LEDs, and the unknown position and orientation of the camera cluster. They solved the non-linear system of equations using an iterative approach that required a good initial guess. During normal system operation the previously known pose was usually an excellent guess for the current pose and the iterative method converged rapidly. Initialization at system startup was accomplished by sequentially trying a small set of different orientations to see if any will converge to a likely solution. The convergence region of the algorithm was large enough that it could acquire the initial position within a few seconds when the tracker was held upright at about head height.

4.2 Stochastic Approaches

While there are many application-specific approaches to “computing” (estimating) the position and orientation or *pose* of an object (see Section 4.1), most of these methods do not inherently take into consideration the noisy nature of the sensor measurements. While the requirements for the pose 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 the material in the appendices. For a more extensive discussion of stochastic estimation see for example (Kailath et al., 2000; Lewis, 1986).

4.2.1 State-Space Models

State-space models are essentially a notational convenience for estimation and control problems, developed to make what would otherwise be a notationally-intractable analysis tractable. 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

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

which ensures (Kailath et al., 2000) that

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

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

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

which leads to the *state-space model*

$$\vec{x}_{i+1} = A\vec{x}_i + Gu_i$$
$$\vec{y}_i = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix} \vec{x}_i$$

or the more general form

$$\vec{x}_{i+1} = A\vec{x}_i + Gu_i \qquad (4.12)$$

$$\vec{y}_i = H_i \vec{x}_i. \qquad (4.13)$$

Equation (4.12) 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 (4.13) 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.

4.2.2 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*.[1] 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 (4.12):

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} . \tag{4.14}$$

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 (4.13):

$$z_k = Hx_k + v_k . \tag{4.15}$$

The terms w_k and v_k are random variables (see Section 2.2.1 on page 23) representing the process and measurement noise respectively.

Measurement and Process Noise

There are many sources of noise in sensor 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 pose estimates, and analytical measurement models typically incorporate some notion of random measurement noise or uncertainty v_k as shown above in equation (4.15).

1. Access to the system's control inputs is also presumed, but less relevant in the case of tracking and motion capture, so we will omit that aspect. See for example (Kailath et al., 2000) for more information.

In the case of tracking or motion capture of humans, there is the additional problem that the user's intended motion is essentially completely unknown. While we can make predictions over relatively short intervals using models based on recent motion as a guide (see Section 5.3), such predictions assume that the user's motion is predictable, which is not always the case. The result is that like sensor information, ongoing estimates of the user pose must be qualified as they are combined with measurements in an overall sequence of pose estimates. In addition, analytical motion or process models typically incorporate some notion of random motion or uncertainty w_k as shown above in equation (4.14).

4.2.3 Optimal Estimation—The Kalman Filter

Among the substantial number of mathematical tools that can be used for stochastic pose 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).

The 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 page 68) in our HiBall Tracking System (Welch et al., 1999, 2001) which is commercially available from 3rdTech (3rdTech, 2000). It has also been used for motion prediction (Azuma, 1995a; Azuma & Bishop, 1994), and it is used for multi-sensor (inertial-acoustic) fusion in the commercial Constellation™ wide-area tracking system by Intersense (Foxlin et al., 1998; Intersense, 2000). See also (Azarbayejani & Pentland, 1994; Emura & Tachi, 1994a, 1994b; Fuchs (Foxlin), 1993; Mazuryk & Gervautz, 1995; Van Pabst & Krekel, 1993).

We maintain a popular web site on the topic of the Kalman filter. The web address is

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

On this site you will find references to (and some copies of) introductory and advanced material on the Kalman filter. New for 2001—we will be bringing on-line a Java-based *Kalman Filter Learning Tool*. In addition, we are also teaching an Introduction to the Kalman Filter two-hour tutorial at SIGGRAPH 2001 (Course 8), for which there are separate course notes. Finally, in Appendix A of this course pack (page 81) we have included a copy of our own introductory technical report. Beyond this Appendix, a very "friendly" introduction to the general idea of the Kalman filter can be found in Chapter 1 of (Maybeck, 1979)—which is available from the above Kalman filter web site, while a more complete introductory discussion can be found in (Sorenson, 1970), which also contains some interesting historical narrative. More extensive references include (Brown & Hwang, 1996; Gelb, 1974; Grewal & Andrews, 2001; Jacobs, 1993; Lewis, 1986; Maybeck, 1979).

4.2.4 Hybrid or Multi-Sensor Fusion

Stochastic estimation tools such as the Kalman filter (see Appendix A on page 81) can be used to combine or *fuse* information from different mediums or sensors for *hybrid systems* (see Section 3.3 on page 56). 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 4.1 (also called a *complementary* or *error-state* Kalman filter) is often used to combine 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 adjustment or *tuning* of the Kalman filter parameters then determines 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 4.1 indicates the additional 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).

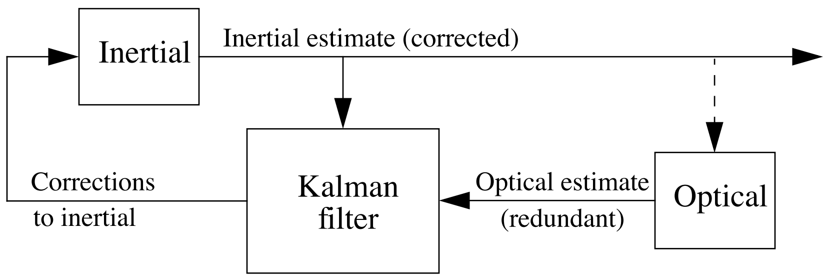


Figure 4.1: The Kalman filter used in an *indirect-feedback* configuration to optimally weight inertial and optical information.

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. See Appendix A on page 81 for more information about these models.

4.2.5 Single-Constraint-at-a-Time Tracking

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 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 & 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 solutions to the observer design problem in general (Section 4.2.2), or the Kalman filter in particular (Section 4.2.3), that dictates the ordering of measurement information. In 1996 we introduced a new Kalman filter-based approach to tracking, 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 & 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 et al., 1999, 2001), the commercial version of the same system (3rdTech, 2000), and the commercial systems manufactured by Intersense (Foxlin et al., 1998; Intersense, 2000).

One of the most interesting things about the SCAAT approach is that it can be almost universally applied in place of conventional approaches (Section 4.1). Essentially all you need is to be able to predict a sensor measurement given a current pose estimate. In other words, if you can formulate a measurement model as in equation (4.15) in Section 4.2.2, you can use the SCAAT approach. In fact, one of the unusual things about the approach is that you never actually compute the pose directly, you only compute the measurement you think the sensor should "see." For computer graphics people, the measurement model for optical sensors in particular is "simple" as it typically resembles the normal graphics viewing transformations.

Consider for a moment the UNC hybrid landmark-magnetic tracker presented at SIGGRAPH 96 (State, Hirota, Chen, Garrett, & Livingston, 1996). 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 predict the location of a landmark in the image, then identify and locate that single landmark in the actual image. It would process one landmark per image update in this fashion. 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 (State et al., 1996), *no special processing would be needed* for the cases where the number of visible landmarks falls below the number 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.

For more information see (Welch & Bishop, 1997) and (Welch, 1996), the latter of which is included at the end of this course pack in Appendix C.

5. Problems and Insights

In this chapter we consider some of the primary sources of error in estimates from tracking and motion capture systems. While we focus specifically on head tracking for interactive computer graphics, the basic principles are applicable to tracking of hands, and even to motion capture. We attempt to provide some insight into the source of the error, and even a means for addressing (not necessarily *solving*) one of the biggest problems: end-to-end delay in the entire tracking and graphics pipeline.

5.1 Classification of Error

There are of course many causes of visual error in interactive computer graphics systems. There are many people (aside from the authors) who would argue that various errors originating in the tracking system dominate all other sources. In his 1995 Ph.D. dissertation thoroughly analyzing the sources of error in an Augmented Reality system for computer-aided surgery, Rich Holloway stated

> Clearly, the head tracker is the major cause of registration error in AR systems. The errors come as a result of errors in aligning the tracker origin with respect to the World CS (which may be avoidable), measurement errors in both calibrated and multibranched trackers, and delay in propagating the information reported by the tracker through the system in a timely fashion.

Rich Holloway's dissertation offers a very thorough look at the sources of error in the entire AR pipeline, including the stages associated with tracking. Much of the dissertation is applicable to VR systems in general, and even motion capture. We highly encourage you to take a look if you are really interested in a rigorous mathematical analysis. Chapter 8 of the dissertation discusses some methods for combating the problems introduced by tracker error, in particular delay. The dissertation is available from http://www.cs.unc.edu/Publications/Dissertations.html.

Sources of Error

For a person designing, calibrating, or using a tracking or motion capture system, it is useful to have some insight into where errors come from. As (Deering, 1992) notes, "...the visual effect of many of the errors is frustratingly similar." This is especially true for tracking errors. We have seen people build VR applications with obvious head tracker transformation errors, and yet people had great difficulty figuring out what part of the long sequence of transforms was wrong, if it was a static calibration error, or a simple sign error.

Yet even when all of the transforms are of the correct form, the units of translation and orientation match, and all the signs are correct, there are still unavoidable errors in motion tracking, errors that confound even the most experienced of practitioners of interactive computer graphics. No matter what the approach (see Chapter 4), the process of pose estimation can be thought of as a sequence of events and operations. The sequence begins with the user motion, and typically ends with a pose estimate arriving at the host computer, ready to be consumed by the application. Clearly by the time a pose estimate arrives at the host computer it is already “late”—and you still have to render an image and wait for it to be displayed! Section 5.3 offers some hope for addressing the long delays and in some sense “catching up” with the user motion, but that doesn’t mean that we don’t want to minimize the delay, and to understand how all of the various errors affect the outcome.

The sources of error in tracking and motion capture can generally be divided into two primary classes. The first includes all errors related to making static measurements, either off line prior to running an application, or on line during normal operation. We call this *static measurement error*. The second includes all errors that arise from the inevitable sources of delay in the tracking pipeline. We call this *delay-induced error*.

5.1.1 Static Measurement Error

Static Field Distortion

For an *immobile* sensor (static motion), we can divide the measurement errors into two types: *repeatable* and *nonrepeatable*. Some trackers (for example, magnetic ones) have systematic, repeatable distortions of their measurement volume which cause them to give erroneous data; we will call this effect *static field distortion*. The fact that these measurement errors are repeatable means that they can be measured and corrected as long as they remain unchanged between this calibration procedure and run time.

Random Noise or Jitter

Here we consider the *non-repeatable* errors made by the tracker for an *immobile* sensor. As we discussed in general in Section 2.1.1, some amount of noise in the sensor inputs is inevitable with any measurement system, and this measurement noise typically leads to random noise or *jitter* in the pose estimates. By our definition, this type of error is not repeatable and therefore not correctable *a priori* via calibration. Moreover, the amount of jitter in the tracker’s outputs limits the degree to which the tracker can be calibrated. The amount of jitter is often proportional to the distance between the sensor(s) and the source(s), and may become relatively large near the edge of the tracker’s working volume.

5.1.2 Delay-Induced Error

Any measurement of a non-repeating, time-varying phenomenon is valid (at best) at the instant the sample occurs—or over the brief interval it occurs, and then becomes "stale" with the passage of time until the next measurement. The age of the data is thus one factor in its accuracy. Any delay between the time the measurement is made and the time that measurement is manifested by the system in a pose estimate contributes to the age and therefore the inaccuracy of that measurement. The older the tracker data is, the more likely that the displayed image will be misaligned with the real world.

We feel that concerns related to *dynamic error* (including *dynamic tracker error* and *delay-induced error* from above) deserve distinct discussion. This class of error is often less obvious when it occurs (you know something isn't correct, but you don't know why), and when you do recognize it, it is difficult to know where to look to minimize the effects.

First-Order Dynamic Error

Probably the most obvious effect here is the overall *dynamic error* caused by continued user motion after a tracker cycle (sample, estimate, produce) has started. If the user's head is rotating with an angular velocity of $\dot{\theta}$ and translating with a linear velocity of $\dot{x}$ then simple first-order models for the delay-induced orientation and translation error are given by

$$\varepsilon_{\text{dyn}, \theta} = \dot{\theta}\Delta t \tag{5.1}$$

$$\varepsilon_{\text{dyn}, x} = \dot{x}\Delta t \tag{5.2}$$

where Δt is the sum of the total motion delay Δt_m for the tracking system as described below in Section 5.2, as well as Δt_g, the delay through the remainder of the graphics pipeline—including rendering and image generation, video synchronization delay, frame synchronization delay, and internal display delay. The *video synchronization delay* is the amount of time spent waiting for a frame buffer to swap—on average $1/2$ the frame time. (Synchronization delay in general is described more below.) The *internal display delay* is any delay added by the display device beyond the normal frame delay. For example, some LCD and DLP devices buffer images internally in a non-intuitive manner. The delay must be measured on a per-device basis if it is important.

The Simultaneity Assumption

Many popular tracking systems collect sensor measurements sequentially, and then assume (mathematically) that they 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. Consider that typical arm and wrist motion can occur in as little as 1/2 second, with typical "fast" wrist tangential motion occurring at three meters per second (Atkeson & Hollerbach, 1985). For the a typical magnetic tracker with 20-80 ms of latency, such "fast" motion corresponds to approximately one to ten centimeters of translation *throughout* the sequence of sensor samples used for a single estimate. For systems that attempt sub-millimeter accuracies, even slow motion occurring during a sequence of sequential samples impacts the accuracy of the estimates. For example, in a multiple-sample system with $\Delta t_s = 30$ [ms] of total sample time, motion of only three centimeters per second corresponds to approximately one millimeter of target translation throughout the sequence of samples for one estimate. Figure 5.1 presents the results of a simulation from (Welch, 1996) (page 161) which includes a more extensive analysis of this error source. Figure 5.1 shows how estimates can be pulled away from the truth by an error amount of ε_{sa} as the simultaneity assumption is violated.

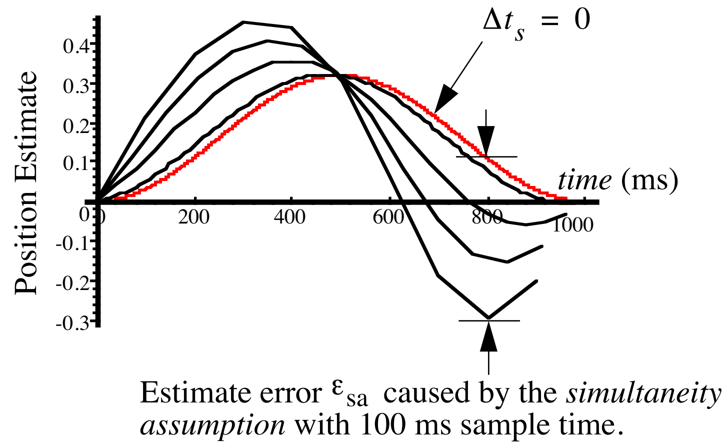


Figure 5.1: Simulated error resulting from the simultaneity assumption. The family of curves shows how simulated position estimates become skewed by the simultaneity assumption as a target undergoes motion with a one Hertz sinusoidal velocity. Note the increasing skew of the estimates with total sensor sample times of $\Delta t_s \in \{0, 10, 40, 70, 100\}$ ms. Details appear in (Welch, 1996).

Sensor Sample Rate

Per Shannon's sampling theorem (Jacobs, 1993) the measurement or *sampling* rate r_{ss} 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 (Fischer, Daniel, & Siva, 1990; Foxlin, 1993; Neilson, 1972), the *sampling* rate should ideally be greater than 40 Hz. Furthermore, the *estimation* rate r_e should be as high as possible so that slight (expected and acceptable) estimation error can be discriminated from the unusual error that might be observed during times of significant target dynamics.

Synchronization Delay

While other latencies (delays) certainly do exist in the typical VE system (Council, 1994; Mine, 1993; Wloka, 1995) tracker latency is unique in that it determines how much time elapses before the first possible opportunity to respond to user motion. When the user moves, we want to know as soon as possible. Within the tracking system pipeline of events (and throughout the rendering pipeline) there are both fixed latencies associated with well-defined tasks such as executing functions to compute the pose, and variable latencies associated with the synchronization between well-defined asynchronous tasks. The latter is often called *synchronization delay*, although sometimes also *phase delay* or *rendezvous delay.* See for example Figure 5.2.

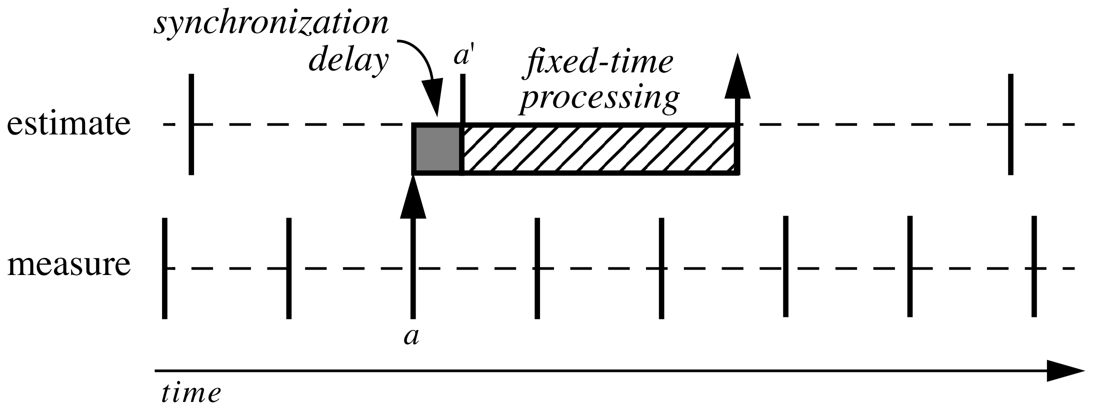


Figure 5.2: Synchronization delay. A measurement is taken at a but not used to estimate the pose until a'. The intervening time is called *synchronization delay.*

In the example of Figure 5.2, measurements and pose estimates occur at regular but *different* rates. Inevitably, any measurement will sit for some time before being used in to compute a pose estimate. At best, the measurement will be read immediately *after* it is made. At worst the measurement will be read just *before* it is replaced with a newer measurement. On average the delay would be $1/2$ the measurement rate.

5.2 Total Tracker Error

Figure 5.3 presents a more involved example, a sequence of inter-tracker events and the corresponding delays. Consider an instantaneous step-like user motion as depicted in Figure 5.3. The sequence of events begins at t_m, the instant the user begins to move. In this example the sensors are sampled at a regular rate $r_{ss} = 1/\tau_{ss}$, such as would typically be the case with video or a high-speed A/D conversion. On average there will be $\Delta t_{ss} = \tau_{ss}/2$ seconds of sample synchronization delay before any sample is used for pose estimation. Because the pose estimate computations are repeated asynchronously at the regular rate of $r_e = 1/\tau_e$ there will be an average of $\Delta t_e = \tau_e/2$ seconds of estimation synchronization delay, after which time the estimation will take τ_e seconds.

Assuming a client-server architecture such as (VRPN, 2001) the final estimate will be written to a server communications buffer where it is being read at a rate of $r_{srb} = 1/\tau_{srb}$, and will therefore wait an average of $\Delta t_{srb} = \tau_{srb}/2$ seconds before being read and transmitted over the network to the client. The network transmission itself will take τ_{net}, and the final client read-buffer synchronization delay will take $\Delta t_{crb} = \tau_{crb}/2$ seconds, where $\tau_{crb} = 1/r_{crb}$ (the client read-buffer rate). The total (average) motion delay in this example is then

$$\begin{aligned} \Delta t_m &= t_{m'} - t_m \\ &= \Delta t_{ss} + \Delta t_e + \tau_e + \Delta t_{srb} + \tau_{net} + \Delta t_{crb} \\ &= \frac{1}{2r_{ss}} + \frac{1}{2r_e} + \tau_e + \frac{1}{2r_{srb}} + \tau_{net} + \frac{1}{2r_{crb}} \end{aligned} \tag{5.3}$$

where r_{ss} is the sensor sample rate, r_e is the estimate rate, $\tau_e = 1/r_e$, r_{srb} is the server read-buffer rate, τ_{net} is the network transmission time, and r_{crb} is the client read-buffer rate.

Note that this bound does not include any latency inherently added by pose estimate computations that also implement some form of filtering.

Summing the *static measurement error* from Section 5.1.1, the error $\varepsilon_{\mathrm{sa}}$ caused by violation of the simultaneity assumption, and the *dynamic error* given by equations (5.1) and (5.2), we get a total error of

$$\begin{aligned} \varepsilon_\theta &\approx \varepsilon_{\mathrm{stat},\,\theta} + \varepsilon_{\mathrm{sa},\,\theta} + \dot{\theta}(\Delta t_m + \Delta t_g) \\ \varepsilon_x &\approx \varepsilon_{\mathrm{stat},\,x} + \varepsilon_{\mathrm{sa},\,x} + \dot{x}(\Delta t_m + \Delta t_g) \end{aligned} \tag{5.4}$$

where Δt_m is from equation (5.3), and Δt_g includes the remainder of the graphics pipeline delay as described in “First-Order Dynamic Error” above in Section 5.1.2. Clearly the final rotation and translation error is sensitive to both the user motion velocity, and the total delay of the tracker and graphics pipeline.

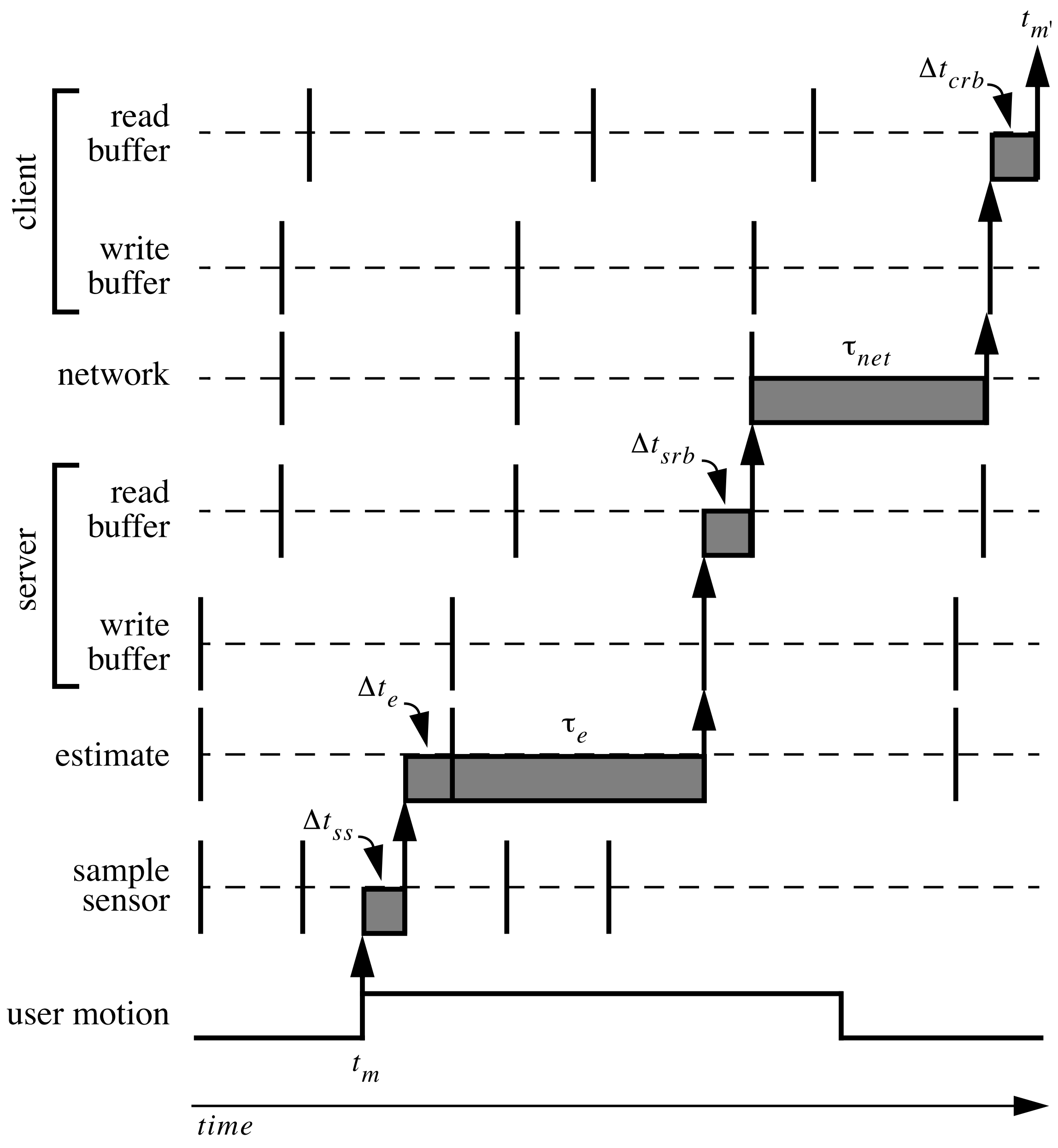


Figure 5.3: An example sequence of inter-tracker events and delays.

5.3 Motion Prediction

When trackers are used to implement VE or AR systems, end-to-end delays the total system will result in perceived *swimming* of the virtual world whenever the user's head moves. The delay causes the virtual objects to appear to follow the user's head motion with a velocity dependent error.

The sequence of events in a head-mounted display system goes something like this:

Time	Event
t_0	tracker measures user's pose
t_1	tracker reports the pose
t_2	application receives the reported pose
t_3	updated image is ready in the hidden buffer of a double-buffered display
t_4	buffer swap happens at vertical interval
t_5	image is scanned out to the display

Table 5.1: Time series of events in a head-mounted display system.

The interval from t_0 to t_5 is on the order of 30ms in the fastest systems and upwards to 200ms in the slowest. If the user is moving during this interval the image finally displayed at t_5 will not be appropriate for the user's new position. We are displaying images appropriate for where the user *was* rather than for where he *is*.

The most important step in combating this swimming is to reduce the end-to-end delay. This process can be taken only so far though. Each of the steps takes *some* time and this time is not likely to be reduced to negligible simply by accelerating the hardware.

After the avoidable delays have been eliminated we can mitigate the effect of the unavoidable delays by using motion prediction. Our goal is to extrapolate the user's past motion to predict where he will be looking at the time the new image is ready. As Azuma (Azuma, 1995a) points out, this is akin to driving a car by looking only the rear-view mirror. To keep the car on the road, the driver must predict where the road will go, based solely on the view of the past and knowledge of roads in general. The difficulty of this task depends on how fast the car is going and on the shape of the road. If the road is straight and remains so, then the task is easy. If the road twists and turns unpredictably, the task will be impossible.

Motion predictors attempt to extract information from past measurements to predict future measurements. Most methods, at their core, attempt to estimate the local derivatives so that a Taylor series can be evaluated to estimate the future value. The differences among methods are mostly in the type and amount of smoothing applied to the data in estimating the derivatives.

The simplest approach simply extends a line through the previous two measurements to the time of the prediction. This approach will be very sensitive to noise in the measurements. More sophisticated approaches will take weighted combinations of several previous measurements. This will reduce sensitivity to noise but will incur a delay in responding to rapid changes. All methods based solely on past measurements of position and orientation will face a trade off between noise and responsiveness.

Performance of the predictor can be improved considerably if direct measurements of the derivatives of motion are available from inertial sensors. As described earlier, linear accelerometers and rate gyros provide estimates of the derivatives of motion with high bandwidth and good accuracy. Direct measurements are superior to differentiating the position and orientation estimates because they are less noisy and are not delayed.

Azuma (Azuma & Bishop, 1994) demonstrated prediction using inertial sensors that reduced swimming in an augmented reality system by a factor of 5 to 10 with end-to-end delay of 80 [ms]. Further in (Azuma & Bishop, 1995) he shows that error in predictions based on derivatives and simple models of motion are related to the square of the product of the prediction interval and the bandwidth of the motion sequence. Doubling the prediction interval for the same sort in input will quadruple the error.

A. An Introduction to the Kalman Filter

This appendix is a copy of UNC technical report TR 95-041, written by Welch and Bishop in 1995. It is included to provide a ready and accessible introduction to both the discrete Kalman filter and the extended Kalman filter. This report and other useful material can be found at the authors' Kalman filter web site, http://www.cs.unc.edu/~welch/kalman/.

A.1 The Discrete Kalman Filter

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [Kalman60]. Since that time, due in large part to advances in digital computing, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. A very "friendly" introduction to the general idea of the Kalman filter can be found in Chapter 1 of [Maybeck79], while a more complete introductory discussion can be found in [Sorenson70], which also contains some interesting historical narrative. More extensive references include [Gelb74; Grewal93; Maybeck79; Lewis86; Brown92; Jacobs93].

A.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

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

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

$$z_k = Hx_k + v_k. \tag{A.2}$$

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

$$p(w) \sim N(0, Q), \tag{A.3}$$

$$p(v) \sim N(0, R). \tag{A.4}$$

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 (A.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 (A.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.

A.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

$$P_k^- = E[e_k^- e_k^{-T}], \tag{A.5}$$

and the *a posteriori* estimate error covariance is

$$P_k = E[e_k e_k^T]. \tag{A.6}$$

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 (A.7). Some justification for equation (A.7) is given in "The Probabilistic Origins of the Filter" found below.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{A.7}$$

The difference $(z_k - H\hat{x}_k^-)$ in equation (A.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 (A.7) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance equation (A.6). This minimization can be accomplished by first substituting equation (A.7) into the above definition for e_k, substituting that into equation (A.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 [Maybeck79; Brown92; Jacobs93]. One form of the resulting K that minimizes equation (A.6) is given by[1]

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \frac{P_k^- H^T}{H P_k^- H^T + R} \end{aligned} \quad . \qquad \text{(A.8)}$$

Looking at equation (A.8) we see that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily. Specifically,

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

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.

A.1.3 The Probabilistic Origins of the Filter

The justification for equation (A.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,

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

1. All of the Kalman filter equations can be algebraically manipulated into to several forms. Equation equation (A.8) represents the Kalman gain in one popular form.

The *a posteriori* state estimate equation (A.7) reflects the mean (the first moment) of the state distribution— it is normally distributed if the conditions of equation (A.3) and equation (A.4) are met. The *a posteriori* estimate error covariance equation (A.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 [Maybeck79; Brown92; Jacobs93].

A.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 A.1.

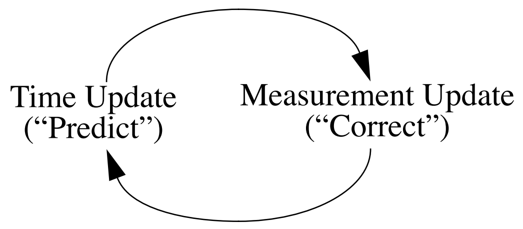


Figure A.1: The ongoing discrete Kalman filter cycle. The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time and measurement updates are presented below in table A.1 and table A.2.

Table A.1: Discrete Kalman filter time update equations.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad \text{(A.9)}$$

$$P_k^- = AP_{k-1}A^T + Q \qquad \text{(A.10)}$$

Again notice how the time update equations in table A.1 project the state and covariance estimates forward from time step $k-1$ to step k. A and B are from equation (A.1), while Q is from equation (A.3). Initial conditions for the filter are discussed in the earlier references.

Table A.2: Discrete Kalman filter measurement update equations.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \qquad \text{(A.11)}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \qquad \text{(A.12)}$$

$$P_k = (I - K_k H)P_k^- \qquad \text{(A.13)}$$

The first task during the measurement update is to compute the Kalman gain, K_k. Notice that the equation given here as equation (A.11) is the same as equation (A.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 (A.12). Again equation (A.12) is simply equation (A.7) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via equation (A.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 [Brown92] 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 A.2 below offers a complete picture of the operation of the filter, combining the high-level diagram of Figure A.1 with the equations from table A.1 and table A.2.

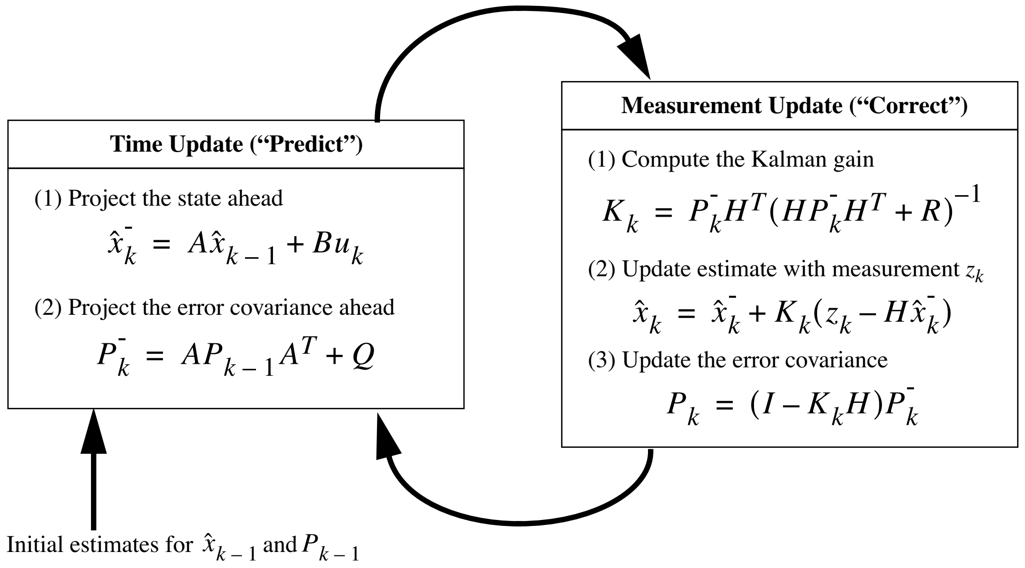

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure A.2: A complete picture of the operation of the Kalman filter, combining the high-level diagram of Figure A.1 with the equations from table A.1 and table A.2.

A.2 The Extended Kalman Filter (EKF)

A.2.1 The Process to be Estimated

As described above in Section A.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 A.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

$$x_k = f(x_{k-1}, u_k, w_{k-1}), \tag{A.14}$$

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

$$z_k = h(x_k, v_k), \tag{A.15}$$

where the random variables w_k and v_k again represent the process and measurement noise as in equation (A.3) and equation (A.4). In this case the *non-linear* function f in the difference equation equation (A.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 (A.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

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0) \tag{A.16}$$

and

$$\tilde{z}_k = h(\tilde{x}_k, 0), \tag{A.17}$$

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 [Julier96].

A.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 (A.16) and equation (A.17),

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}, \tag{A.18}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k. \tag{A.19}$$

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 (A.16) and equation (A.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 (A.3) and equation (A.4).
- *A* is the Jacobian matrix of partial derivatives of f with respect to *x*, that is

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0),$$

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

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_k, 0),$$

- *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,

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k, \tag{A.20}$$

and the measurement residual,

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k. \tag{A.21}$$

Remember that in practice one does not have access to x_k in equation (A.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 (A.21), it is the actual measurement that one is using to estimate x_k. Using equation (A.20) and equation (A.21) we can write governing equations for an *error process* as

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k , \tag{A.22}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k , \tag{A.23}$$

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 (A.3) and (A.4) respectively.

Notice that the equations equation (A.22) and equation (A.23) are linear, and that they closely resemble the difference and measurement equations equation (A.1) and equation (A.2) from the discrete Kalman filter. This motivates us to use the actual measurement residual $\tilde{e}_{z_k}$ in equation (A.21) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{e}_{x_k}$ given by equation (A.22). This estimate, call it $\hat{e}_k$, could then be used along with equation (A.20) to obtain the *a posteriori* state estimates for the original non-linear process as

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k . \tag{A.24}$$

The random variables of equation (A.22) and equation (A.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

$$\hat{e}_k = K_k\tilde{e}_{z_k} . \tag{A.25}$$

By substituting equation (A.25) back into equation (A.24) and making use of equation (A.21) we see that we do not actually need the second (hypothetical) Kalman filter:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k\tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k(z_k - \tilde{z}_k) \end{aligned} \tag{A.26}$$

Equation equation (A.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 (A.16) and equation (A.17), and the Kalman gain K_k coming from equation (A.11) with the appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below in table A.3 and table A.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 A.3: EKF time update equations.

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{A.27}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{A.28}$$

As with the basic discrete Kalman filter, the time update equations in table A.3 project the state and covariance estimates from the previous time step $k-1$ to the current time step k. Again f in equation (A.27) comes from equation (A.16), A_k and W_k are the process Jacobians at step k, and Q_k is the process noise covariance equation (A.3) at step k.

Table A.4: EKF measurement update equations.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{A.29}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{A.30}$$

$$P_k = (I - K_k H_k) P_k^- \tag{A.31}$$

As with the basic discrete Kalman filter, the measurement update equations in table A.4 correct the state and covariance estimates with the measurement z_k. Again h in equation (A.30) comes from equation (A.17), H_k and V are the measurement Jacobians at step k, and R_k is the measurement noise covariance equation (A.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 A.1. Figure A.3 below offers a complete picture of the operation of the EKF, combining the high-level diagram of Figure A.1 with the equations from table A.3 and table A.4.

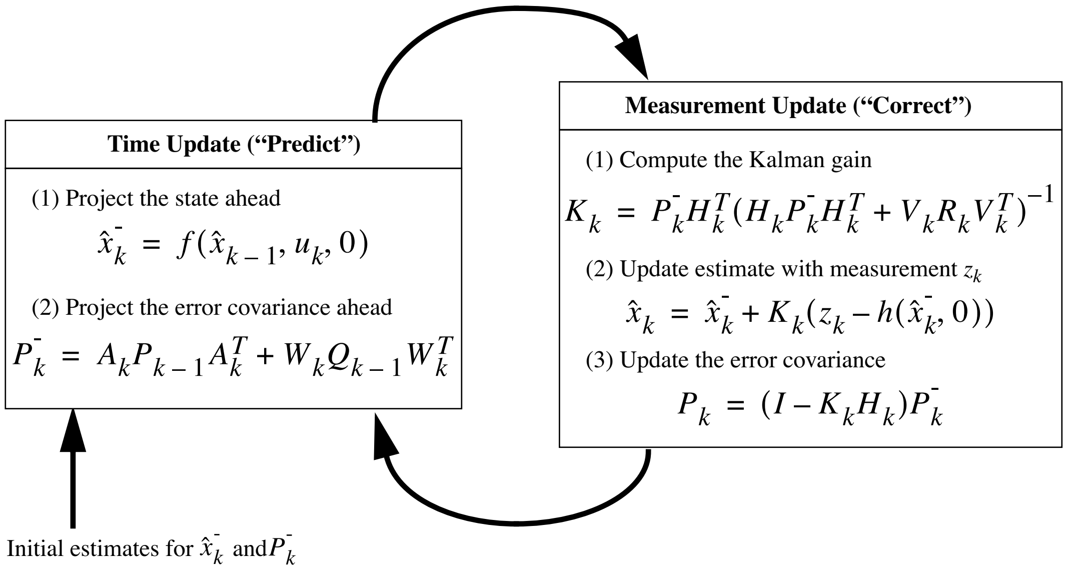

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0)$
(2) Project the error covariance ahead
$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$
(3) Update the error covariance
$P_k = (I - K_k H_k) P_k^-$
Initial estimates for $\hat{x}_k^-$ and P_k^-

Figure A.3: A complete picture of the operation of the *extended* Kalman filter, combining the high-level diagram of Figure A.1 with the equations from table A.3 and table A.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*.

A.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.

A.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.)

A.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

$$\begin{aligned} K_k &= P_k^-(P_k^- + R)^{-1} \\ &= \frac{P_k^-}{P_k^- + R} \end{aligned}, \tag{A.32}$$

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

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

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 $P_0 = 1$.

A.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 A.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.

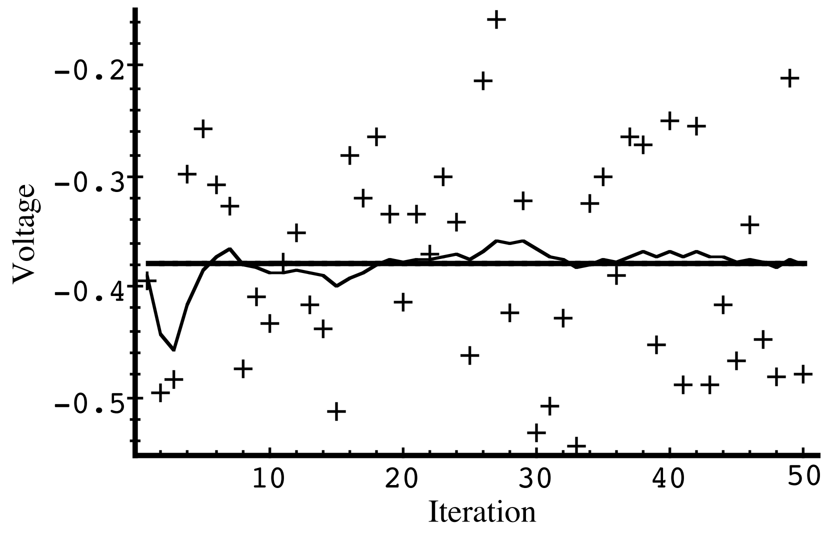


Figure A.4: The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

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 A.5 we have plotted the value of P_k versus the iteration. By the 50th iteration, it has settled from the initial (rough) choice of 1 to approximately 0.0002 (Volts^2).

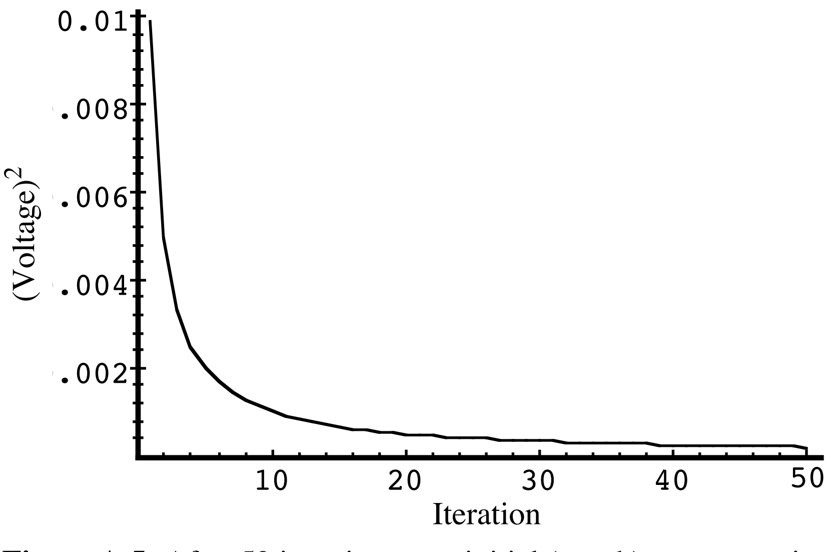


Figure A.5: After 50 iterations, our initial (rough) error covariance P_k choice of 1 has settled to about 0.0002 (Volts^2).

In Figure A.6 and Figure A.7 below we can see what happens when R is increased or decreased by a factor of 100 respectively. In Figure A.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.

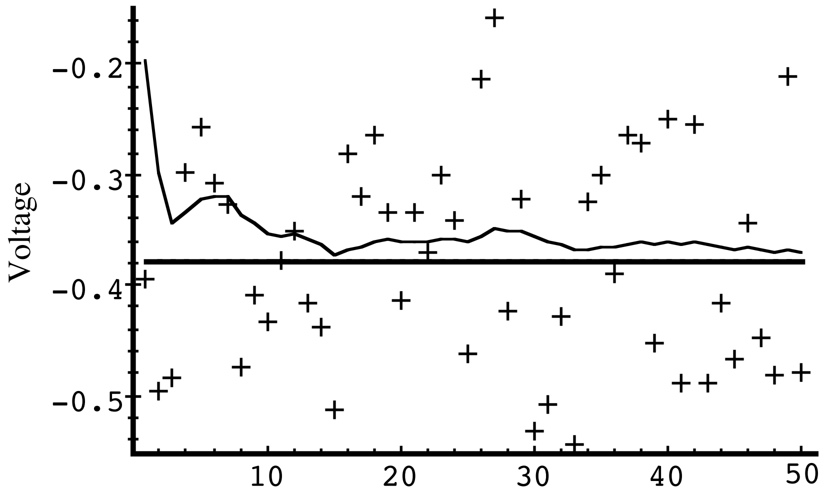


Figure A.6: Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

In Figure A.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.

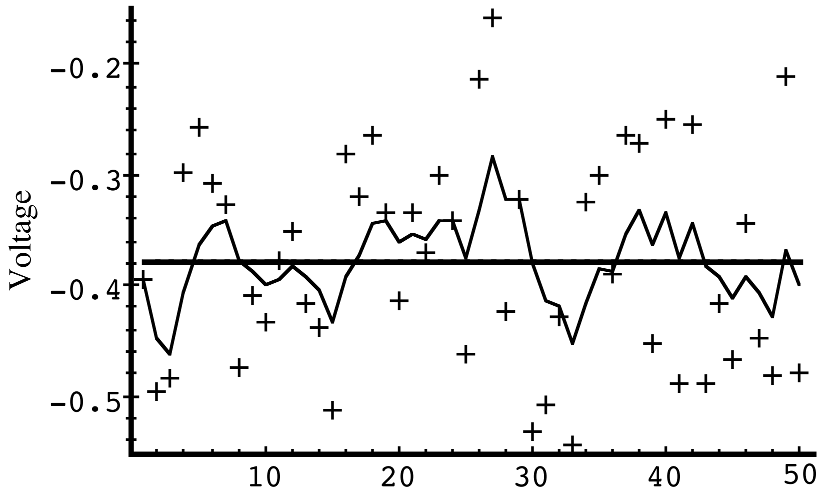


Figure A.7: Third simulation: $R = 0.0001$. The filter responds to measurements quickly, increasing the estimate variance.

While the estimation of a constant is relatively straight-forward, it clearly demonstrates the workings of the Kalman filter. In Figure A.6 in particular the Kalman "filtering" is evident as the estimate appears considerably smoother than the noisy measurements.

References

Brown92	Brown, R. G. and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*, John Wiley & Sons, Inc.
Gelb74	Gelb, A. 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.
Grewal93	Grewal, Mohinder S., and Angus P. Andrews (1993). Kalman Filtering Theory and Practice. Upper Saddle River, NJ USA, Prentice Hall.
Jacobs93	Jacobs, O. L. R. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.
Julier96	Julier, Simon and Jeffrey Uhlman. "A General Method of Approximating Nonlinear Transformations of Probability Distributions," Robotics Research Group, Department of Engineering Science, University of Oxford [cited 14 November 1995]. Available from http://www.robots.ox.ac.uk/~siju/work/publications/Unscented.zip.
	Also see: "A New Approach for Filtering Nonlinear Systems" by S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, Proceedings of the 1995 American Control Conference, Seattle, Washington, Pages:1628-1632. Available from http://www.robots.ox.ac.uk/~siju/work/publications/ACC95_pr.zip.
	Also see Simon Julier's home page at http://www.robots.ox.ac.uk/~siju/.
Kalman60	Kalman, R. E. 1960. "A New Approach to Linear Filtering and Prediction Problems," Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).
Lewis86	Lewis, Richard. 1986. *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.
Maybeck79	Maybeck, Peter S. 1979. *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.
Sorenson70	Sorenson, H. W. 1970. "Least-Squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63-68, July 1970.

B. Tracking Bibliography

This appendix includes a listing of most of the tracking-related publications that we are aware of and refer to.This list is also available in electronic bibliography format at

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

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

Agar, W. O., & Blythe, J. H. (1968). An optical method of measuring transverse surface velocity. *Journal of Scientific Instruments (Journal of Physics E) 1968 Series 2, 1*, 25-28.

Aidala, V. J. (1979). Kalman filter behavior in bearings-only tracking applications. *IEEE Transactions on Aerospace and Electronic Systems, AES-15*(1), 29-39.

Aidala, V. J., & Hammel, S. E. (1983). Utilization of modified polar coordinates for bearings-only tracking. *IEEE Trans. Automat. Contr., AC-28*, 283-294.

Akatsuka, Y., & Bekey, G. A. (1998). Compensation for end to end delays in a VR system, *Proceedings of IEEE VRAIS'98* (pp. 156-159). Atlanta, GA: IEEE.

Antonsson, E. K., & Mann, R. W. (1989). Automatic 6-D.O.F. kinematic trajectory acquisition and analysis. *Journal of Dynamic Systems, Measurement, and Control, 111*, 31-39.

Ascension. (2000). *Ascension Technology Corporation*, [HTML]. Ascension Technology Corporation. Available: http://www.ascension-tech.com/ [2000, September 15].

Atkeson, C. G., & Hollerbach, J. M. (1985). Kinematic features of unrestrained vertical arm movements. *Journal of Neuroscience, 5*, 2318-2330.

Ator, J. T. (1963). Image-velocity sensing with parallel-slit recticles. *Journal of the Optical Society of America, 53*(12), 1416-1422.

Ator, J. T. (1966). Image velocity sensing by optical correlation. *Applied Optics, 5*(8), 1325-1331.

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

Azarbayejani, A., & Pentland, A. (1995a). *Camera self-calibration from one point correspondence* (Perceptual Computing Technical Report 341): MIT Media Laboratory.

Azarbayejani, A., & Pentland, A. (1995b). Recursive Estimation of Motion, Structure, and Focal Length. *IEEE Trans. Pattern Analysis and Machine Intelligence, 17*(6), 562-575.

Azarbayejani, A., & Pentland, A. (1996). *Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features* (Technical report 363). Cambridge, MA: Massachusetts Institute of Technology (MIT).

Azuma, R. T. (1993, July). Tracking Requirements for Augmented Reality. *Communications of the ACM, 36,* 50-51.

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

Azuma, R. T. (1995b). *Predictive Tracking for Augmented Reality* (TR95-007). Chapel Hill, NC: University of North Carolina at Chapel Hill, Department of Computer Science.

Azuma, R. T. (1995c). *A survey of augmented reality*.Unpublished manuscript, Malibu, CA.

Azuma, R. T. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 355-385.

Azuma, R. T. (1999). The Challenge of Making Augmented Reality Work Outdoors. In Y. Ohta & H. Tamura (Eds.), *Mixed Reality: Merging Real and Virtual Worlds* (pp. 379-390). Yokohama, Japan: Springer-Verlag.

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., & Bishop, G. (1995). A Frequency-Domain Analysis of Head-Motion Prediction, *Computer Graphics* (SIGGRAPH 94 Conference Proceedings ed., pp. 401-408). Los Angeles, CA: ACM Press, Addison-Wesley.

Azuma, R. T., Hoff, B. R., Neely, H. E., III, Sarfaty, R., Daily, M. J., Bishop, G., Chi, V., Welch, G., Neumann, U., You, S., Nichols, R., & Cannon, J. (1998). Making Augmented Reality Work Outdoors Requires Hybrid Tracking, *First International Workshop on Augmented Reality* (pp. 219-224). San Francisco, CA, USA.

Azuma, R. T., Hoff, B. R., & Neely, H. E. I. (1999). A Motion-Stabilized Outdoor Augmented Reality System, *IEEE Virtual Reality* (pp. 252-259). Houston, TX USA.

Azuma, R. T., Lee, J. W., Jiang, B., Park, J., You, S., & Neumann, U. (1999). Tracking in unprepared environments for augmented reality systems. *Computers & Graphics, 23*(6), 787-793.

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.

Bachmann, E., Robert. (2000). *Inertial and Magnetic Tracking of Limb Segment Orientation for Inserting Humans into Synthetic Environments*. Unpublished Ph.D. Thesis, The Naval Postgraduate School, Monterey, CA.

Bachmann, E., Robert, Duman, I., McGhee, R. B., & Zyda, M. J. (1999). *Sourceless Sensing of Limb Segment Angles for Inserting Humans Into Networked Virtual Environment*: The Naval Postgraduate School.

Bailey, T., Nebot, E., M., Rosenblatt, J. K., & Durrant-Whyte, H., F. (1999). Robust distinctive place recognition for topological maps, *International Conference on Field and Service Robotics (FSR 99)*. Pittsburgh, PA USA.

Bajura, M., & Neumann, U. (1995). Closed-Loop Tracking for Augmented-Reality Systems. *IEEE Computer Graphics & Applications, 15*(5), 52-60.

Bajura, M., & Neumann, U. (2001). *Dynamic Compensation of Alignment Error in Augmented-Reality Systems*, [HTML]. Available: http://www.usc.edu/dept/CGIT/papers/VRpose_94_022.pdf [2001, March 24, 2001].

Bancroft, S. (1984). An algebraic solution of the GPS equations. *IEEE Transactions on Aerospace and Electronic Systems, AES-21*(7), 56-59.

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

Baron, S., Lancraft, R., & Caglayan, A. (1984). *An optimal control model approach to the design of compensators for simulator delay* (NASA Contractor Report 3064). Cambridge, MA: Bolt Beranek and Newman Inc.

Behringer, R. (1999). Registration for Outdoor Augmented Reality Applications Using Computer Vision Techniques and Hybrid Sensors, *IEEE Virtual Reality* (pp. 244-251). Houston, TX, USA.

Bell, B. M., & Cathey, F. W. (1993). The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control, 38*(2), 294-297.

Best, R. E. (1999). Phase-Locked Loops: Design, Simulation, and Applications.

Bhatnagar, D. K. (1993). *Position trackers for Head Mounted Display systems: A survey* (Technical Report TR93-010). Chapel Hill, NC USA: University of North Carolina at Chapel Hill.

Bible, S. R., Zyda, M., & Brutzman, D. Using spread-spectrum ranging techniques for position tracking in a virtual environment (pp. 15).

Bishop, G. (1984). *The Self-Tracker: A Smart Optical Sensor on Silicon*. Unpublished Ph.D. Dissertation, University of North Carlina at Chapel Hill, Chapel Hill, NC USA.

Bishop, G., & Fuchs, H. *Self-Tracker: a VLSI-based three-dimensional input system* (paper 83-002). Chapel Hill, NC: University of North Carolina at Chapel Hill Department of Computer Science.

Bishop, G., & Fuchs, H. (1984). The Self-Tracker: A Smart Optical Sensor on Silicon, *Advanced Research in VLSI* (pp. 65-73). Massachusetts Institute of Technology: Artech House.

BL. (2000). *CODA mpx30 Motion Capture System*, [html]. B & L Engineering. Available: http://www.charndyn.com/ and http://www.bleng.com/coda.htm [2000, April 27].

Bolles, R. C., Kremers, J. H., & Cain, R. A. (1981). *A simple sensor to gather three-dimensional data* (Technical Note 249

SRI Project 1538). Menlo Park, CA: SRI International, Industrial Automation Department, Computer Science and Techology Division.

Bordtad, A. J. (1985). Bearings-only target motion analysis estimation characteristics. *Control and Computers, 13*(3), 95-101.

Borghese, N. A., & Ferrigno, G. (1990). An algorithm for 3-D automatic movement detection by means of standard TV cameras. *IEEE Transactions on Biomedical Engineering, 37*(12), 1221-1225.

Bouget, J.-Y. (1997). 3D transformations & camera calibration (pp. 7 pp).

Britannica, E. (1994). *Encyclopedia Britannica*, [HTML]. Encyclopedia Britannica. Available: http://www.britannica.com/ [2001, April 27].

Broida, T. J., & Chellappa, R. (1986). Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence, 8*(1), 90-99.

Brown, R. G., & Hwang, P. Y. C. (1992). *Introduction to Random Signals and Applied Kalman Filtering* (Second ed.): Wiley & Sons, 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.

Brown, R. G., & Hwang, P. Y. C. (1997). Introduction to Random Signals and Applied {K}alman Filtering: with {MATLAB} Exercises and Solutions.

Brugger, W., & Milner, M. (1978). Computer-aided tracking of body motions using a c.c.d.-image sensor. *Medical & Biological Engineering & Computing, 16*, 207-210.

Bruss, A. R., & Horn, B. K. P. (1981). *Passive navigation* (A.I. Memo 662). Cambridge, MA: Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Bryson, S. (1992). SPIE Proceedings Volume 1669 Stereoscopic Displays and Applications III. In J. O. Merritt & S. S. Fisher (Eds.), *SPIE* (Vol. 1669, pp. 244-255). San Jose, CA.

Bui, H. H., Vankatesh, S., & West, G. (2000). A probabilistic framework for tracking in wide-area environments, *Proceedings of the International Conference on Pattern Recognition (ICPR'00)* (Vol. 4, pp. 702-706). Barcelona, Spain.

Burdea, G., & Coiffet, P. (1994). *Virtual Reality Technology* (First ed.): John Wiley & Sons, Inc.

Burton, R. P. (1973). *Real-Time Measurement of Multiple Three-Dimensional Positions*. Unpublished Ph.D., University of Utah, Salt Lake City, UT USA.

Burton, R. P., & Sutherland, I. E. (1974). TWINKLEBOX: A Three-Dimensional Computer-Input Device, *AFIPS Conference Proceedings, 1974 National Computer Conference* (Vol. 43, pp. 513-520). Chicago, IL USA: AFIPS Press, Montvale, New Jersey.

Card, S. K., Mackinlay, J. D., & Robertson, G. G. (1991). A Morphological Analysis of the Design Space of Input Devices. *9*(2), 99-122.

Cavallaro, R. (1997). The FoxTrax hockey puck tracking system, *IEEE CG&A* (pp. 6-12).

Chi, V. L. (1995). *Noise Model and Performance Analysis Of Outward-looking Optical Trackers Using Lateral Effect Photo Diodes* (TR95-012). Chapel Hill, NC USA: University of North Carlina at Chapel Hill.

Chou, J. C. K. (1992). Quaternion kinematic and dynamic equations. *IEEE Transactions on Robotics and Automation, 8*(1), 53-64.

Corporation, A. (2001, 2000). *Gypsy Motion Capture System*, [HTML]. Analogus Corporation. Available: http://www.metamotion.com/gypsy-motion-capture-system/gypsy-motion-capture-system.htm [2001, March 7].

Council, N. R. (1994). *Virtual Reality, Scientific and Technological Challenges*. Washington, DC: National Academy Press.

Crane, D. F. (1980). The effects of time delay in man-machine control systems: Implementations for design of flight simulator-display-delay compensation, *IMAGE III* (pp. 331-343). Williams AFB, AZ: Air Force Human Resources Laboratory.

Crowley, J. L., & Demazeau, Y. (1993). Principles and Techniques for Sensor Data Fusion. *Signal Processing (EURASIP), 32*, 5-27.

Darrell, T., Azarbayejani, A., & Pentland, A. P. (1994). *Segmentation of rigidly moving objects using multiple Kalman filters* (Technical report 281). Cambridge, MA: Massachusetts Institute of Technology (MIT).

David, P., Balakirsky, S., & Hillis, D. (1990). A real-time automatic target acquisition system, *Symposium of the Association for Unmanned Vehicle Systems* (pp. 13 pp). Dayton, OH.

De Geeter, J., Van Brussel, H., De Schutter, J., & Decreton, M. (1996). Recognising and locating objects with local sensors, *IEEE Conference on Robotics and Automation* (pp. 6 pp (In conf. proceedings: 3478-3483)). Minneapolis, Minnesota.

Deering, M. F. (1992). High resolution virtual reality. In E. E. Catmull (Ed.), *Computer Graphics (Proceedings of SIGGRAPH 92)* (Vol. 26, pp. 195-202). Chicago, Illinois.

Deyst, J., J., & Price, C. F. (1968). Conditions for Asymptotic Stability of the Discrete Minimum-Variance Linear Estimator. *IEEE Transactions on Automatic Control*.

Division, S. D. I. (2001). *Gyrochip theory of operation*, [HTML]. BEI Electronics Company, Systron Donner Inertail Division [2001, April 27].

Donner, S. (2001). *Systron Donner Home Page*, [HTML]. Available: http://www.systron.com/ [2001, April 27].

Dowski, E., R. (1995). An Information Theory Approach to Incoherent Information Processing Systems. *Signal Recovery and Synthesis V, OSA Technical Digest Series*, 106-108.

Drane, C., R. (1992). *Positioning Systems : A Unified Approach* (Vol. 181): Springer Verlag.

Durlach, N., & Mavor, A. S. (1994). Position tracking and mapping, *Virtual reality: scientific and technological challenges* (pp. 188-204). Washington, D.C.: National Academy Press.

Ellis, S. R., Adelstein, B. D., Baumeler, S., Jense, G. J., & Jacoby, R. H. (1999). Sensor spatial distortion, visual latency, and update rate effects on 3D tracking in virtual environments, *Proceedings of the IEEE Virtual Reality*. Houston, Texas: Institute of Electrical and Electronics Engineers.

Ellis, S. R., & Menges, B. M. (1997). Judgements of the distance to nearby virtual objects: interaction of viewing conditions and accommodative demand. *Presence: Teleoperators and Virtual Environments, 6*(4), 452-460.

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.

Emura, S., & Tachi, S. (1994). Sensor Fusion based Measurement of Human Head Motion, *3rd IEEE International Workshop on Robot and Human Communication (RO-MAN 94 NAGOYA)* (pp. 124-129). Nagoya University, Nagoya, Japan.

Etienne-Cummings, R., Spiegel, V. d., & Mueller, P. (1997). A focal plane visual motion measurement sensor. *IEEE Transactions on Circuits and Systems, 44*(1), 55-66.

Falconer, D. G. (1979). *Target tracking with the Hough and Fourier-Hough transform* (Technical Note 202). Menlo Park, CA: SRI International.

Faugeras, O. (1999). *Three-dimensional computer vision: a geometric viewpoint* (3rd ed. Vol. 1). Cambridge, Massachusetts: The MIT Press.

Feder, H. J. S., Leonard, J. J., & Smith, C. M. (1999). Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research, 18*(7), 650-668.

Feiner, S., MacIntyre, B., & Höllerer, T. (1997). A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment, *Proceedings of First International Symposium on Wearable Computers* (pp. 74-81). Cambridge, MA.

Feiner, S., MacIntyre, B., Höllerer, T., & Webster, A. (1997). A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring Urban Environments. *Personal Technologies, 1*(4), 208-217.

Ferrin, F. J. (1991). Survey of Helmet Tracking Technologies, *Large-Screen-Projection, Avionic, and Helmet-Mounted Displays* (Vol. 1456, pp. 86--94).

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

Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM, 24*(6), 381-395.

Fleming, R., & Kushner, C. (1995). Lowe-Power, Miniature, Distributed Position Location and Communication Devices Using Ultra-Wideband, Nonsinusoidal Communication Technology.

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1997). *Computer Graphics: Principles and Practice* (2nd ed. Vol. 1). Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Fox, D., Burgard, W., & Thrun, S. (1999). Probabilistic Methods for Mobile Robot Mapping, *IJCAI-99 Workshop on Adaptive Spatial Representations of Dynamic Environments* (pp. 10).

Foxlin, E. (1993). *Inertial Head-Tracking*. Unpublished Master of Science, Massachusetts Institute of Technology (MIT), Cambridge, MA.

Foxlin, E. (1996). Intertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter, *VRAIS 96* (pp. 185-194). Los Alamitos, CA USA: IEEE.

Foxlin, E., & Durlach, N. (1994). An Inertial Head-Orientation Tracker with Automatic Drift Compensation for Use With HMD's. In G. Singh & S. Feiner & D. Thalmann (Eds.), *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 159-173). Singapore: ACM SIGGRAPH, Addison-Wesley.

Foxlin, E., Harrington, M., & Altshuler, Y. (1998). Miniature 6-DOF Inertial System for Tracking HMDs, *SPIE Helmet and Head-Mounted Displays III* (Vol. 3362). Orlando, FL USA: SPIE.

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.

Frey, W., Zyda, M., McGhee, R., & Cockayne, W. (1996). *Off-the-Shelf, Real-Time, Human Body Motion Capture for Synthetic Environments* (Technical Report NPSCS-96-003). Monterey, CA, USA: The Naval Postgraduate School.

Friedman, M., Starner, T., & Pentland, A. (1992). Synchronization in Virtual Realities. *Presence: Teleoperators and Virtual Environments, 1*, 139-144.

Friedmann, M., Starner, T., & Pentland, A. (1992). Device Synchronization Using an Optimal Linear Filter, *1992 Symposium on Interactive 3D Graphics*. Cambridge, MA USA.

Frye, W. E. (1957 ?). Fundamentals of inertial guidance and navigation. *Journal of the Astronautical Sciences*, 1-10.

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

Fuchs, H., Duran, J., & Johnson, B. (1977). A system for automatic acquisition of three-dimensional data, *AFIPS Conf. Proc.* (Vol. 46, pp. 49-53).

Fuhrmann, A., Loffelmann, H., & Schmalstieg, D. (1997). Collaborative augmented reality: exploring dynamical systems, *Proceedings of IEEE Visualization '97* (pp. 459-462). Phoenix, AZ.

Ganapathy, S. Decomposition of transformation matrices for robot vision (pp. 21).

Ganapathy, S. (1984). *Real-time motion tracking using a single camera* (Technical memorandum 11358-841105-21-TM, Charge case 311306-0399, File case 39394): AT&T Bell Laboratories.

Ganapathy, S. (1994). *Camera Location Determination Problem* (Technical Memorandum 11358-841102-20-TM): AT&T Bell Laboratories Technical Memorandum.

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

Gillis, J. T. (1991). Examination of 3-D angular motion using gyroscopes and linear accelerometers. *IEEE Transactions on Aerospace and Electronic Systems, 27*(6), 910-920.

Golding, A. R., & Lesh, N. (1999). *Indoor navigation using a diverse set of cheap, wearable sensors* (Technical Report TR99-32). Cambridge, MA, USA: Mitsubishi Electric Information Technology Center America.

Goncalves, L., Di Bernardo, E., Ursella, E., & Perona, P. (1995). Monocular tracking of the human arm in 3D, *ICCV'95* (pp. 7 pp (In conference proceedings: 764-770)).

Goshtasby, A. A. (2001). *Active Vision*. Available: http://www.cs.wright.edu/people/faculty/agoshtas/3.activevis/ActiveVis.html [2001, April 27].

Gottschalk, S., & Hughes, J. F. (1993). Autocalibration for Virtual Environments Tracking Hardware. In J. T. Kajiya (Ed.), *Computer Graphics* (SIGGRAPH 93 Conference Proceedings ed., pp. 65-72). Anaheim, CA USA: ACM Press, Addison Wesley.

Grewal, M., S., & Andrews, A., P. (1993). *Kalman Filtering Theory and Practice*. Upper Saddle River, NJ USA: Prentice Hall.

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

Grewal, M. S., Weill, L., R., & Andrews, A. P. (2001). *Global Positioning Systems, Inertial Navigation, and Integration*. New York, NY USA: John Wiley & Sons, Inc.

Guivant, J., Nebot, E., M., & Durrant-Whyte, H., F. (2000). Simultaneous localization and map bulding using natural features in outdoor environments, *IAS-6 Intelligent Autonomous Systems*. Italy.

Ham, F. M., & Brown, R. G. (1983). Observability, eigenvalues, and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems, AES-19*(2), 269-273.

Hamilton, W. R. (1853). *Lectures on Quaternions*. Dublin: Hodges and Smith.

Held, R., & Durlach, N. (1991). Telepresence, time delay, and adaptation. In S. R. Ellis (Ed.), *Pictorial Communication in Virtual and Real Environments* (pp. 28-21 - 28-16): Taylor and Francis.

Herring, T. A. (1996). The global positioning system. *Scientific American*, 44-50.

Hill, P. D., & Walsh, T. R. (1992). The advanced modular tracker: a real-time video tracker. Danvers, MA: Datacube, Inc.

Hoff, W. A., & Nguyen, K. (1996). Computer vision-based registration techniques for augmented reality, *Proceedings of Intelligent Robots and Computer Vision XV, SPIE* (Vol. 2904, pp. 538-548). Boston, MA.

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., & Schwehm, M. (1999a). *Next century challenges: Nexus--an open global infrastructure for spatial-aware applications*. Paper presented at the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking.

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., & Schwehm, M. (1999b). Next century challenges: Nexus--an open global infrastructure for spatial-aware applications, *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (pp. 249–255).

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., & Schwehm, M. (1999c). Nexus - An open global infrastructure for spatial-aware applications (pp. 13 pp).

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., & Hallaway, D. (1999). Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics, 23*(6), 779-785.

Holloway, R. (1997). Registration error analysis for augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 413-432.

Holloway, R. L. (1995). *Registration Errors in Augmented Reality Systems*. Unpublished Ph.D. Dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

Huang, T. S., & Netravali, A. N. (1994). Motion and structure from feature correspondences: a review, *Proceedings of the IEEE* (Vol. 82, pp. 252-267).

Ickes, B. P. (1970). A new method for performing digital control systems attitude computations using quaternions. *AIAA Journal, 5*(1), 13-17.

IGT. (2000). *FlashPoint 5000*, [HTML]. Image Guided Technologies. Available: http://www.imageguided.com/ [2000, September 15].

Iltanen, M., Kosola, H., Palovuori, K., & Vanhala, J. (1998). Optical Positioning and Tracking System for a Head Mounted Display Based on Spread Spectrum Technology, *Proceedings of the 2nd International Conference on Machine Automation (ICMA)* (pp. 597--608).

Inigo, R. M., & McVey, E. S. (1981). CCD Implementation of a three-dimensional videotracking algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3*(2), 230-240.

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

Irani, M., Rousso, B., & Peleg, S. (1997). Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(3), 6 pp.

Jacobs, M. C., Livingston, M. A., & State, A. (1997). Managing latency in complex augmented reality systems, *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (pp. 49-54). Providence, RI.

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

Janin, A., Zikan, K., Mizell, D., Banner, M., & Sowizral, H. (1994). A videometric headtracker for augmented reality applications, *Telemanipulator and Telepresence Technologies* (Vol. 2351, pp. 308-315). Bellingham, WA.

Jebara, T., Eyster, C., Weaver, J., Starner, T., & Pentland, A. (1997). An optical tracker for augmented reality and wearable computers, *Proceedings of the First International Symposium on Wearable Computers* (pp. 146-150). Cambridge, MA.

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.

Julier, S., J., Uhlmann, J., K., & Durrant-Whyte, H., F. (1995). A New Approach for Filtering Nonlinear Systems, *1995 American Control Conference* (pp. 628-1632).

Kadaba, M. P., & Stine, R. (2000). *Real-Time Movement Analysis Techniques and Concepts for the New Millennium in Sports Medicine*, [HTML]. Motion Analysis Corporation, Santa Rosa, CA USA. Available: http://www.motionanalysis.com/applications/movement/rtanalysis.html [2000, September 15].

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

Kalawsky, R. S. (1993). *The Science of Virtual Reality and Virtual Environments* (First ed.): Addison-Wesley Publishing Company.

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.

Kanade, T. Very fast 3-D sensing hardware (pp. 15 pp).

Kaplan, E. D. E. (1996). *Understanding GPS Principles and Applications*: Artech House.

Kim, D., Richardst, S., W., & Caudellt, T., P. (1997). An Optical Tracker for Augmented Reality and Wearable Computers, *1997 Virtual Reality Annual International Symposium (VRAIS '97)* (pp. 146-150). Albuquerque, NM: IEEE.

Kite, D. H., & Magee, M. (1990). Determining the 3D position and orientation of a robot camera using 2D monocular vision. *Pattern Recognition, 23*(8), 819-831.

Klinker, G. J., Ahlers, K. H., Breen, D. E., Chevalier, P.-Y., Crampton, C., Greer, D. S., Koller, D., Kramer, A., Rose, E., Tuceryan, M., & Whitaker, R. T. (1997). Confluence of computer vision and interactive graphics for augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 433-451.

Kolasinski, E. M. (1995). *Simulator sickness in virtual environments* (Technical report 1027). Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.

Koller, D., Klinker, G., Rose, E., Breen, D., Whitaker, R., & Tuceryan, M. Real-time vision-based camera tracking for augmented reality applications (pp. 8 pp).

Koller, D., Klinker, G., Rose, E., Breen, D., Whitaker, R., & Tuceryan, M. (1997a). Automated camera calibration and 3D egomotion estimation for augmented reality applications, *Proceedings of the CAIP '97, 7th International Conference Computer Analysis of Images and Patterns* (pp. 199-206).

Koller, D., Klinker, G., Rose, E., Breen, D., Whitaker, R., & Tuceryan, M. (1997b). Real-time vision-based camera tracking for augmented reality applications, *Proceedings of the ACM Symposium on Virtual Reality, Software and Technology (VRST-97).*

Krause, L. O. (1987). A direct solution to GPS-type navigation equations. *IEEE Transactions on Aerospace and Electronic Systems, AES-23*(2), 225-232.

Krauss, T. (1997). Beware of shortcomings when applying classical spectral-analysis techniques. *Personal Engineering*, 36-42.

Krouglicof, N., McKinnon, G. M., & Svoboda, J. (1987). Optical position and orientation measurement techniques. United States: CAE Electronics, Ltd., Montreal, Canada.

Krouglicof, N., Svoboda, J. V., & McKinnon, G. M. (1986). Noncontact position and orientation measurement techniques for real-time systems, *Proceedings of ASME International Computers in Engineering Conference* (pp. 177-183).

Kubach, U., Leonhardi, A., Rothermel, K., & Schwehm, M. (1999). *Analysis of distribution schemes for the management of location information*. Stuttgart: Institut fur Parallele und Verteilte, Hochstleistungrechner (IPVR), Universitat Stuttgart.

Kuipers, J. B. (1980). SPASYN - An electromagnetic relative position and orientation tracking system. *IEEE Transactions on Instrumentation and Measurement, IM-29*(4), 462-466.

Kuipers, J. B. (1998). *Quaternions and Rotation Sequences*. Princeton: Princeton University Press.

Kumar, R., & Hanson, A. R. (1994). Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image Understanding, 60*(3), 313-342.

Kutulakos, K. N., & Vallino, J. R. (1998). Calibration-free augmented reality. *IEEE Transactions on Visualization and Computer Graphics, 4*(1), 1-20.

Laughlin, D. R., Ardaman, A. A., & Sebesta, H. R. (1992). Inertial angular rate sensors: theory and applications. *Sensors*(October), 20-24.

Lee, J. W., & Neuman, U. (2000). Motion Estimation with Incomplete Information using Omnidirectional Vision, *ICIP2000*.

Lee, J. W., You, S., & Neuman, U. (2000). Large Motion Estimation for Omnidirectional Vision, *IEEE Workshop on Omnidirectional Vision 2000*.

Leonhardi, A., & Kubach, U. (1999). An Architecture for a Distributed Universal Location Service, *Proceedings of the European Wireless '99 Conference* (pp. 351-355). Munich, Germany: ITG Fachbericht, VDE Verlag.

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

Lindgren, A. G., & Gong, K. F. (1978). Position and velocity estimation via bearing observations. *IEEE Transactions on Aerospace and Electronic Systems, AES-14*(4), 564-577.

Link, B. (1993). Field-qualified silicon accelerometers: from 1 milli g to 200,000 g. *Sensors*, 28-33.

List, U. H. (1983). *Nonlinear prediction of head movements for helmet-mounted displays* (Final technical paper): Operations Training Division, Williams Air Force Base, Arizona.

Livingston, M. A., & State, A. (1997). Magnetic tracker calibration for improved augmented reality registration. *Presence: Teleoperators and Virtual Environments, 6*(5), 532-546.

Lucas, B. D., & Kanade, T. (1985). Optical navigation by the method of differences. In A. K. Joshi (Ed.), *Proceedings of the 9th International Joint Conference on Artificial Intelligence* (pp. 981-984): Morgan Kaufmann.

MAC. (2000). *HiRes 3D Motion Capture System*, [html]. Motion Analysis Corporation. Available: http://www.motionanalysis.com/applications/movement/gait/3d.html [2000, September 15].

Macellari, V. (1983). CoSTEL: a computer peripheral remote sensing device for 3-dimensional monitoring of human motion. *Medical & Biological Engineering & Computing, 21*, 311-318.

Madhavan, R., Dissanayake, M., & Durrant-Whyte, H., F. (1998). Map-building and map-based localization in an underground-mine by statistical pattern matching. In S. V. A.K. Jain, B.C. Lovell (Ed.), *International Conference on Pattern Recognition (ICPR)* (Vol. 2, pp. 1744-1746). Brisbane, Australia: IEEE Computer Society Press Piscataway, NJ USA.

Mahmoud, R., Loffeld, O., & Hartmann, K. (1994). Multisensor Data Fusion for Automated Guided Vehicles, *Proceedings of SPIE - The International Society for Optical Engineering* (Vol. 2247, pp. 85-89).

Mann, R. W., Rowell, D., Dalrymple, G., Conati, F., Tetewsky, A., Ottenheimer, D., & Antonsson, E. (1983). Precise, rapid, automatic 3-D position and orientation tracking of multiple moving bodies. In H. Matsui & K. Kobayashi (Eds.), *Proceedings of the VIII international congress of biomechanics* (pp. 1104-1112). Champaign, IL: Human Kinetics.

Mark, W. R., McMillan, L., & Bishop, G. (1997). Post-rendering 3D warping, *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (pp. 7-16). Providence, RI.

Matthies, L., & Shafer, S. A. (1987). Error Modeling in Stereo Navigation. *IEEE Journal of Robotics and Automation, RA-3*(3), 239-248.

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).

McFarland, R. E. (1986). *CGI delay compensation* (NASA Technical Memorandum NASA TM 86703): NASA Scientific and Technical Information Branch.

McFarland, R. E. (1988). *Transport delay compensation for computer-generated imagery systems* (NASA Technical Memorandum NASA TM 100084). Moffett Field, CA: NASA Ames Research Center.

Meditch, S. (1969). *Stochastic Optimal Linear Estimation and Control*. New York: McGraw-Hill.

Meyer, K., Applewhite, H., & Biocca, F. (1991). A survey of position trackers (pp. 61).

Meyer, K., Applewhite, H. L., & Biocca, F. A. (1992). A Survey of Position Trackers. *Presence, a publication of the Center for Research in Journalism and Mass Communication, 1*(2), 173-200.

Mine, M. R. (1993). *Characterization of end-to-end delays in head-mounted display systems* (Technical report TR93-001). Chapel Hill, NC: Department of Computer Science, University of North Carolina at Chapel Hill.

Molineros, J., Raghavan, V., & Sharma, R. (1998). Computer vision based augmented reality for guiding and evaluating assembly sequences (poster), *Proceedings of IEEE VRAIS '98* (pp. 214). Atlanta, GA.

Mulder, A. (1994a). *Human Movement Tracking Technology* (Technical Report TR 94-1): School of Kinesiology, Simon Fraser University.

Mulder, A. (1994b, May 8, 1998). *Human Movement Tracking Technology: Resources*, [HTML]. School of Kinesiology, Simon Fraser University. Available: http://www.cs.sfu.ca/people/ResearchStaff/amulder/personal/vmi/HMTT.add.html [2000, September 15].

Mulder, A. (1998, May 8, 1998). *Human Movement Tracking Technology*, [HTML]. School of Kinesiology, Simon Fraser University. Available: http://www.cs.sfu.ca/people/ResearchStaff/amulder/personal/vmi/HMTT.pub.html [2000, September 15].

Nardone, S. C., & Aidala, V. J. (1981). Observability criteria for bearings-only target motion analysis. *IEEE Transactions on Aerospace and Electronic Systems, AES-17*(2), 162-166.

Nash, J. (1997, October 1997). Wiring the jet set. *Wired, 5,* 128-135.

Nayar, S. K., Watanabe, M., & Noguchi, M. (1994). *Real-time focus range sensor* (paper CUCS-028-94). New York, NY: Department of Computer Science, Columbia University.

NDI. (2001). *OPTOTRAK*, [HTML]. Northern Digital Inc. Available: http://www.ndigital.com/optotrak.html [2001, April].

Nebot, E., M. (1999). Sensors Used for Autonomous Navigation. In S. G. Tzafestas (Ed.), *ADVANCES IN INTELLIGENT AUTONOMOUS SYSTEMS* (pp. 135-156): Kluwer Academic Publisher (Dordrecht / Boston / London).

Nebot, E., M., Durrant-Whyte, H., F, & Scheding, S. (1988). Frequency domain modelling of aided GPS for vehicle navigation systems. *Robotics and Autonomous Systems, 25*(1), 73-82.

Neilson, P. D. (1972). Speed of Response or Bandwidth of Voluntary System Controlling Elbow Position in Intact Man. *Medical and Biological Engineering, 10*, 450-459.

Nettleton, E. W., Gibbens, P. W., & Durrant-Whyte, H., F. (2000). Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem, *AeroSense 2000*. Orlando, FL USA.

Neumann, U., & Cho, Y. (1996). A self-tracking augmented reality system, *ACM International Symposium on Virtual Reality and Applications* (pp. 109-115). Hong Kong.

Neumann, U., & Majoros, A. (1998). Cognitive, performance, and system issues for augmented reality applications in manufacturing and maintenance, *Proceedings of IEEE VRAIS '98* (pp. 4-11). Atlanta, GA.

Neumann, U., & Park, J. (1998). Extendible object-centric tracking for augmented reality, *Proceedings of IEEE VRAIS '98* (pp. 148-155). Atlanta, GA.

Neumann, U., & You, S. (1998). Integration of region tracking and optical flow for image estimation.

Neumann, U., & You, S. (1999). Natural Feature Tracking for Augmented Reality. *IEEE Transactions on Multimedia, 1*(1), 53-64.

Newman, J. (1999). An example of the extended Kalman filter (pp. 7 pp).

Oceanographers, W. (2001). *Women Exploring the Oceans*, [HTML]. Available: http://www.womenoceanographers.org [2001, April 27].

Ohshima, T., Satoh, K., Yamamoto, H., & Tamura, H. (1998). AR2 hockey: a case study of collaborative augmented reality, *Proceedings of IEEE VRAIS '98* (pp. 268-275). Atlanta, GA: IEEE.

Pasman, W., van der Schaaf, A., Lagendijk, R., L., & Jansen, F., W. (1999). Accurate overlaying for mobile augmented reality. *Computers & Graphics, 23,* 875-881.

Petridis, V. (1981). A method for bearings-only velocity and position estimation. *IEEE Transactions on Automatic Control, AC-26*(2), 488-493.

Phillips, D. (2000, April). On the Right Track—A unique optical tracking system gives users greater freedom to explore virtual worlds. *Computer Graphics World,* 16-18.

Polhemus. (2000). *Polhemus*, [HTML]. Polhemus. Available: http://www.polhemus.com/home.htm [2000, September 15].

Raab, F. H., Blood, E. B., Steiner, T. O., & Jones, H. R. (1979). Magnetic Position and Orientation Tracking System. *IEEE Transactions on Aerospace and Electronic Systems, AES-15*, 709-718.

Rae-Dupree, J. (1997, July 26). Experts Look at Where Computing is Headed; Leaving the Desktop Behind. *San Jose Mercury News,* pp. 1.

Rauch, H. E. (1963). Solutions to the Linear Smoothing Problem. *IEEE Transactions on Auto. Control, AC-8*, 371-372.

Rauch, H. E., Tung, F., & Striebel, C. T. (1965). Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal, 3*, 1445-1450.

Rehg, J. M., & Kanade, T. (1993). *DigitEyes: vision-based human hand tracking* (CMU-CS-93-220). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Rehg, J. M., & Kanade, T. (1994). Visual tracking of high dof articulated structures: An application to human hand tracking. In J.-O. Eklundh (Ed.), *Proceedings of European Conference on Computer Vision* (Vol. 2, pp. 35-46): Springer-Verlag.

Rekimoto, J. (1997). NaviCam: a magnifying glass approach to augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 399-412.

Rekimoto, J., & Nagao, K. (1995). The World through the Computer: Computer Augmented Interaction with Real World Environments, *1995 Symposium on User Interface Software and Technology (UIST 95)*. Pittsburgh, PA: Association of Computing Machinery.

Reunert, M. K. (1993). Fiber-optic gyroscopes: principles and applications. *Sensors*(August), 37-38.

Richards, J. (1998). The Measurement of Human Motion: A Comparison of Commercially Available Systems, *3D Conference of Human Movement*. University of Tennessee, Chattanooga, TN USA.

Richards, J. (1999). The Measurement of Human Motion: A Comparison of Commercially Available Systems. *Human Movement Sciences, 18*.

Riner, B., & Browder, B. (1992). Design guidelines for a carrier-based training system, *IMAGE VI Conference* (pp. 65-73). Scottsdale, Arizona.

Roberts, K. S., & Ganapathy, S. (1987). *Stereo Triangulation Techniques* (Technical Memorandum 11352-861121-07TM). Holmdel, NJ: AT&T Bell Laboratories.

Roberts, L. G. (1966). The Lincoln Wand, *Proceedings of the 1966 Fall Joint Computer Conference, AFIPS Conference Proceedings* (Vol. 29, pp. 223-227).

Robinett, W., & Holloway, R. (1994). *The virtual display transformation for virtual reality*. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, Department of Computer Science.

Rose, E. J., Bose, S. C., Kouba, J. T., & Sobek, D. A. (1985). *A cost/performance analysis of hybrid inertial/externally referenced positioning/orientation systems* (ETL-R-086). Fort Belvoir, VA: U.S. Army Engineer Topographic Laboratories.

Salz, J., & Netravali, A. N. (1983). *Algorithms for estimation of 3-D motion* (Technical Memorandum TM 11345-831110-17, 1138-831110-01): Bell Laboratories.

Sawhney, H. S. (1994). Simplifying motion and structure analysis using planar parallax and image warping, *International Conference on Pattern Recognition, 1994* (pp. 14 pp). Jerusalem.

Sawhney, H. S., Ayer, S., & Gorkani, M. (1994). Model-based 2D & 3D dominant motion estimation for mosaicing and video representation (pp. 31 pp).

Sawhney, H. S., Ayer, S., & Gorkani, M. (1995). Model-based 2D & 3D dominant motion estimation for mosaicing and video representation, *Proceedings of the IEEE International Conference on Computer Vision* (pp. 583-590). Cambridge, MA.

Scheding, S., Dissanayake, M., Nebot, E., M., & Durrant-Whyte, H., F. (1999). An experiment in autonomous navigation of an underground mining vehicle. *IEEE Transactions on Robotics & Automation, 15*(1), 85-95.

Schodl, A., Haro, A., & Essa, I. A. (1998). *Head tracking using a textured polygonal model* (Technical report GIT-GVU-98-24). Atlanta, GA: Georgia Institute of Technology, College of Computing, Graphics, Visualization and Usability Center.

Schut, G. B. (1960). On exact linear equations for the computation of the rotational elements of absolute orientation. *Photogrammetria, 16*(1), 34-37.

Selspot. (1987). *Selspot MULTILab system description* (1987-02-05).

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, 27*, 379–423.

Sharma, R., & Molineros, J. (1997). Computer vision-based augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(3), 292-317.

Shaw, C., & Liang, J. (1992). An experiment to characterize head motion in VR and RR using MR, *Proceedings of the 1992 Western Computer Graphics Symposium* (pp. 99-101).

Shoemake, K. (1985). Animating Rotation with Quaternion Curves, *Computer Graphics* (SIGGRAPH 85 Conference Proceedings ed., pp. 245-254). San Francisco: ACM Press.

Smith, B. R. (1984). Digital head tracking and position prediction for helmet mounted visual display systems, *AIAA 22nd Aerospace Sciences Meeting*. New York, NY: AIAA.

So, R. H. Y., & Griffin, M. J. (1992a). Compensating lags in head-coupled displays using head position prediction and image deflection. *Journal of Aircraft, 29*(6), 1064-1068.

So, R. H. Y., & Griffin, M. J. (1992b). *Selspot Technical Specifications, Selcom Laser Measurements*.

Soatto, S., Perona, P., Frezza, R., & Picci, G. (1996). Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control, 41*(3), 393-414.

Sorensen, B. R., Donath, M., Yang, G.-B., & Starr, R. C. (1989). The Minnesota Scanner: a prototype sensor for three-dimentional tracking of moving body segments. *IEEE Transactions on Robotics and Automation, 5*(4), 499-509.

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

Sowizral, H., & Barnes, D. (1993). Tracking Position and Orientation in a Large Volume, *Proceedings of IEEE VRAIS 93* (pp. 132-139): IEEE Computer Society Press.

SPIRIT. (2000). *SPIRIT Project: Making networks location-aware*. Available: http://www.uk.research.att.com/spirit/ [2000, January 4].

Spohrer, J. (1999a). *3. Key subproblem: determining location*, [HTML]. Learning Communities Group, ATG, (c)Apple Computer, Inc. Available: http://worldboard.org/pub/spohrer/wbconcept/S03.html [1999, December 24, 1999].

Spohrer, J. (1999b). Information in Places. *IBM Systems Journal, Pervasive Computing, 38*(4).

Spohrer, J. (1999c, June 16). *WorldBoard: What Comes After the WWW?*, [HTML]. Learning Communities Group, ATG, (c)Apple Computer, Inc. Available: http://worldboard.org/pub/spohrer/wbconcept/default.html [1999, December 24, 1999].

State, A., Hirota, G., Chen, D. T., Garrett, B., & Livingston, M. A. (1996). Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. In H. Rushmeier (Ed.), *SIGGRAPH 96 Conference Proceedings* (pp. 429-438): Addison Wesley.

Strickland, D., Patel, A., Stovall, C., Palmer, J., & McAllister, D. Self tracking of human motion for virtual reality systems (pp. 10 pp).

Strickland, D., Patel, A., Stovall, C., Palmer, J., & McAllister, D. (1994). Self tracking of human motion for virtual reality systems, *Proceedings of the Stereoscopic Displays and Virtual Reality Systems* (Vol. 2177, pp. 278-287). Bellingham, WA: SPIE.

Sutherland, I. E. (1968). A head-mounted three dimensional display, *Proceedings of the 1968 Fall Joint Computer Conference, AFIPS Conference Proceedings* (Vol. 33, part 1, pp. 757-764). Washington, D.C.: Thompson Books.

Sutherland, I. E. (1974). Three-dimensional data input by tablet, *Proceedings of the IEEE* (Vol. 62 (4), pp. 453-461).

Tarabanis, K. A., Allen, P. K., & Tsai, R. Y. (1995). A survey of sensor planning in computer vision. *IEEE Trans. Robotics and Automation, 11*, 86-104.

Thompson, E. H. (1959). An exact linear solution of the problem of absolute orientation. *Photogrammetria, 15*(4), 163-179.

Thrun, S., Fox, D., & Burgard, W. (1998). A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning, 31*, 29-53.

Trucco, E., & Verri, A. (1998). *Introductory techniques for 3-D computer vision* (1st ed. Vol. 1). Upper Saddle River, New jersey: Prentice Hall.

UNC Tracker Project. (2000, July 10). *Wide-Area Tracking; Navigation Technology for Head-Mounted Displays*, [HTML]. Available: http://www.cs.unc.edu/~tracker [2000, July 18].

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.

Varona, J. G., Roca, F. X., & Villanueva, J. J. (2000). iTrack: Image-based probabilistic tracking of people, *Proceedings of the International Conference on Pattern Recognition (ICPR'00)* (Vol. 3, pp. 1122-1125). Barcelona, Spain.

Verplaetse, C., James. (1996). Inertial proprioceptive devices: Self-motion-sensing toys and tools. *IBM Systems Journal, 35*(3 & 4).

Verplaetse, C. J. (1997). *Inertial-Optical Motion-Estimating Camera for Electronic Cinematography*. Unpublished Masters of Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Vicci, L. (2001). *Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation* (Technical Report TR01-014). Chapel Hill, NC: University of North Carolina at Chapel Hill.

Vieville, T., Clergue, E., & Facao, P. E. D. S. (1993). Computation of ego-motion and structure from visual and inertial sensors using the vertical cue, *IEEE Proceedings of the Third International Conference on Computer Vision* (pp. 591-598). Berlin, Germany.

VRPN. (2001). *Virtual Reality Peripheral Network*, [HTML]. 3rdTech. Available: http://www.cs.unc.edu/Research/vrpn/ [2001, April 25].

Wallmark, J. T. (1957). A new semiconductor photocell using lateral photo-effect. *Proceedings IRE, 45*, 474-483.

Wang, J.-f. *A real-time optical tracker using off-the-shelf components*. Unpublished Dissertation proposal, University of North Carolina at Chapel Hill, Chapel Hill, NC.

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.

Wang, J.-F., Azuma, R. T., Bishop, G., Chi, V., Eyles, J., & Fuchs, H. (1990). Tracking a Head-Mounted Display in a Room-Sized Environment with Head-Mounted Cameras, *SPIE 1990 Technical Symposium on Optical Engineering and Photonics in Aerospace Sensing* (Vol. 1290, pp. 47-57). Orlando, FL: SPIE.

Wang, J.-f., Chi, V., & Fuchs, H. (1990). A Real-time Optical 3D Tracker for Head-mounted Display Systems, *Symposium on Interactive 3D Graphics* (I3D 90 Symposium Proceedings ed., Vol. 24 (2), pp. 205-215). Snowbird, UT: ACM Press, Addison Wesley.

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.

Ware, C., & Balakrishnan, R. (1994). Target acquisition in fish tank VR: the effects of lag and frame rate, *Proceedings of Graphics Interface '94* (pp. 1-7).

Warnekar, C. S., & Schalkoff, R. J. (1982). A predictor-corrector approach to tracking 3-D objects using perspective-projected images, *Proceedings of the IEEE Southeastcon '82* (pp. 371-384).

Watanabe, K., Kobayashi, K., & Munekata, F. (1994). Multiple sensor fusion for navigation systems, *1994 Vehicle Navigation & Information Systems Conference Proceedings* (pp. 575-578): IEEE.

Weber, H. (1997, April 9). *Predictive Head Tracking Using a Body-centric Coordinate System*. University of North Carolina at Chapel Hill. Available: http://www.cs.unc.edu/~weberh/research/predtrac.html and Weber1997_pred_tracking.pdf [2001, January 21].

Wefald, K. M., & McClary, C. (1984). Autocalibration of a laser gyro strapdown inertial reference/navigation system. *IEEE*, 66-74.

Welch, G. (1995). *Hybrid Self-Tracker: An Inertial/Optical Hybrid Three-Dimensional Tracking System* (TR95-048). Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, Department of Computer Science.

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. (1995). *An Introduction to the Kalman Filter* (TR95-041). Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, Department of Computer Science.

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. (2001, December 8, 2000). *The Kalman Filter*, [HTML]. University fo North Carolina at Chapel Hill. Available: http://www.cs.unc.edu/~welch/kalman/ [2001, January 1].

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).

White, P. R., & Garrett, R. E. (1976). A generalized interactive three dimensional input system, *CAD 76, Second International Conference and Exhibition on Computers in Engineering and Building Design* (pp. 275-282): IPC Science and Technology Press.

Wildes, R. P. (1991). Direct recovery of three-dimensional scene geometry from binocular stereo disparity. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(4), 761-774.

Williams, S. B., Newman, P., Dissanayake, M., & Durrant-Whyte, H., F. (2000). Autonomous underwater simultaneous localisation and map building, *2000 IEEE International Conference on Robotics and Automation*. San Francisco, CA USA: Institute of Electrical and Electronic Engineers, Inc.

Wloka, M. M. (1995). Lag in Multiprocessor Virtual Reality. *PRESENCE: Teleoperators and Virtual Environments, 4*(1), 50-63.

Woltring, H. J. (1974). New Possibilities for Human Motion Studies by Real-Time Light Spot Position Measurement. *Biotelemetry, 1*, 132-146.

Woltring, H. J. (1976). Calibration and measurement in 3-dimensional monitoring of human motion by optoelectronic means. II. Experimental results and discussion. *Biotelemetry, 3*, 65-97.

Wood, D. N., Azuma, D. I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D. H., & Stuetzle, W. (2000). Surface Light Fields for 3D Photography. In K. Akeley (Ed.), *Proceedings of SIGGRAPH 2000* (Vol. http://www.cs.washington.edu/homes/daniel/siggraph2000-slf.pdf, pp. 287-296): ACM Press / ACM SIGGRAPH / Addison Wesley Longman.

You, S., Neumann, U., & Azuma, R. T. (1999a, March 13-17). *Hybrid Inertial and Vision Tracking for Augmented Reality Registration*. Paper presented at the IEEE Virtual Reality, Houston, TX USA.

You, S., Neumann, U., & Azuma, R. T. (1999b). Hybrid Inertial and Vision Tracking for Augmented Reality Registration, *IEEE Virtual Reality* (pp. 260-267). Houston, TX USA.

You, S., Neumann, U., & Azuma, R. T. (1999c, Nov/Dec). Orientation Tracking for Outdoor Augmented Reality Registration. *IEEE Computer Graphics and Applications, 19,* 36-42.

Youngblut, C., Johnson, R. E., Nash, S. H., Wienclaw, R. A., & Will, C. A. (1996). Full body motion interfaces, *Review of Virtual Environment Interface Technology* (pp. 181-208). Alexandria, VA: Institute for Defense Analyses.

Yun, W., & Howe, R. T. (1992). Recent developments in silicon microaccelerometers. *Sensors*(October), 31-41.

Zikan, K., Curtis, W., D., Sowrizal, H., A., & Janin, A., L. (1994). A note on dynamics of human head motions and on predictive filtering of head-set orientations, *Telemanipulator and Telepresence Technologies*.

C. Related Papers

This appendix includes a sample of some relevant papers that we have permission to reproduce. See also the “Tracking Bibliography” on page 97, and the electronic bibliography version at

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

The Visual Display Transformation for Virtual Reality

TR94-031
September, 1994

Warren Robinett
Richard Holloway

Head-Mounted Display Project
Department of Computer Science
CB #3175, Sitterson Hall
UNC-Chapel Hill
Chapel Hill, NC 27599-3175

This research was supported by the following grants: ARPA DABT 63-93-C-0048, NSF Cooperative Agreement ASC-8920219, and ARPA "Science and Technology Center for Computer Graphics and Scientific Visualization", ONR N00014-86-K-0680, and NIH 5-R24-RR-02170.

The Visual Display Transformation for Virtual Reality

Warren Robinett[*]
Richard Holloway[†]

Abstract

The visual display transformation for virtual reality (VR) systems is typically much more complex than the standard viewing transformation discussed in the literature for conventional computer graphics. The process can be represented as a series of transformations, some of which contain parameters that must match the physical configuration of the system hardware and the user's body. Because of the number and complexity of the transformations, a systematic approach and a thorough understanding of the mathematical models involved is essential.

This paper presents a complete model for the visual display transformation for a VR system; that is, the series of transformations used to map points from object coordinates to screen coordinates. Virtual objects are typically defined in an object-centered coordinate system (CS), but must be displayed using the screen-centered CSs of the two screens of a head-mounted display (HMD). This particular algorithm for the VR display computation allows multiple users to independently change position, orientation, and scale within the virtual world, allows users to pick up and move virtual objects, uses the measurements from a head tracker to immerse the user in the virtual world, provides an adjustable eye separation for generating two stereoscopic images, uses the off-center perspective projection required by many HMDs, and compensates for the optical distortion introduced by the lenses in an HMD. The implementation of this framework as the core of the UNC VR software is described, and the values of the UNC display parameters are given. We also introduce the vector-quaternion-scalar (VQS) representation for transformations between 3D coordinate systems, which is specifically tailored to the needs of a VR system.

The transformations and CSs presented comprise a complete framework for generating the computer-graphic imagery required in a typical VR system. The model presented here is deliberately abstract in order to be general-purpose; thus, issues of system design and visual perception are not addressed. While the mathematical techniques involved are already well known, there are enough parameters and pitfalls that a detailed description of the entire process should be a useful tool for someone interested in implementing a VR system.

1. Introduction

A typical virtual reality (VR) system uses computer-graphic imagery displayed to a user through a *head-mounted display (HMD)* to create a perception in the user of a surrounding three-dimensional virtual world. It does this by tracking the position and orientation of the user's head and rapidly

[*] Virtual Reality Games, Inc., 719 E. Rosemary St., Chapel Hill NC 27514. E-mail: robinettw@aol.com

[†] Department of Computer Science, CB 3175, University of North Carolina, Chapel Hill, NC, 27599-3175. Email: holloway@cs.unc.edu

generating stereoscopic images in coordination with the user's voluntary head movements as the user looks around and moves around in the virtual world.

The hardware for a typical VR system consists of an HMD for visual input, a *tracker* for determining position and orientation of the user's head and hand, a graphics computer for generating the correct images based on the tracker data, and a hand-held *input device* for initiating actions in the virtual world. The visual environment surrounding the user is called the *virtual world*. The world contains *objects*, which are collections of graphics primitives such as polygons. Each object has its own position and orientation within the world, and may also have other attributes. The human being wearing the HMD is called the *user*, and also has a location and orientation within the virtual world.

A good graphics programmer who is given an HMD, a tracker, an input device, and a computer with a graphics library can usually, after some trial and error, produce code to generate a stereoscopic image of a virtual object that, as the user moves to observe it from different viewpoints, appears to hang stably in space. It often takes several months to get to this point. Quite likely, the display code will contain some "magic numbers" which were tweaked by trial and error until the graphics seen through the display looked approximately right. Further work by the programmer will enable the user to use a tracked manual input device to pick up virtual objects and to fly through the virtual world. It takes more work to write code to let the user scale the virtual world up and down, and have virtual objects that stay fixed in room or head or hand space. Making sure that the constants and algorithms in the display code both match the physical geometry of the HMD and produce correctly sized and oriented graphics is very difficult and slow work.

In short, writing the display code for a VR system and managing all of the transformations (or transforms, for short) and coordinate systems can be a daunting task. There are many more coordinate systems and transforms to keep track of than in conventional computer graphics. For this reason, a systematic approach is essential. Our intent here is to explain all of the coordinate systems and transformations necessary for the visual display computation of a typical VR system. We will illustrate the concepts with a complete description of the UNC VR display software, including the values for the various display parameters. In doing so, we will introduce the vector-quaternion-scalar (VQS) representation for 3D transformations and will argue that this data structure is well suited for VR software.

2. Related Work

Sutherland built the first computer-graphics-driven HMD in 1968 (Sutherland, 1968). One version of it was stereoscopic, with both a mechanical and a software adjustment for interpupillary distance. It incorporated a head tracker, and could create the illusion of a surrounding 3D computer graphic environment. The graphics used were very simple monochrome 3D wire-frame images.

The VCASS program at Wright-Patterson Air Force Base built many HMD prototypes as experimental pilot helmets (Buchroeder, Seeley, & Vukobradatovitch, 1981).

The Virtual Environment Workstation project at NASA Ames Research Center put together an HMD system in the mid-80's (Fisher, McGreevy, Humphries, & Robinett, 1986). Some of the early work on the display transform presented in this paper was done there.

Several see-through HMDs were built at the University of North Carolina, along with supporting graphics hardware, starting in 1986 (Holloway, 1987). The development of the display algorithm reported in this paper was begun at UNC in 1989.

CAE Electronics of Quebec developed a fiber-optic head-mounted display intended for flight simulators (CAE, 1986).

VPL Research of Redwood City, California, began selling a commercial 2-user HMD system, called "Reality Built for 2," in 1989 (Blanchard, Burgess, Harvill, Lanier, Lasko, Oberman, & Teitel, 1990).

A prototype see-through HMD targeted for manufacturing applications was built at Boeing in 1992 (Caudell & Mizell, 1992). Its display algorithm and the measurement of the parameters of this algorithm is discussed in (Janin, Mizell & Caudell, 1993).

Many other labs have set up HMD systems in the last few years. Nearly all of these systems have a stereoscopic HMD whose position and orientation is measured by a tracker, with the stereoscopic images generated by a computer of some sort, usually specialized for real-time graphics. Display software was written to make these HMD systems function, but except for the Boeing HMD, we are not aware of any detailed, general description of the display transformation for HMD systems. While geometric transformations have also been treated at length in both the computer graphics and robotics fields (Foley, van Dam, Feiner, & Hughes, 90), (Craig, 86), (Paul, 81), these treatments are not geared toward the subtleties of stereoscopic viewing in a head-mounted display. Therefore, we hope this paper will be useful for those who want to implement the display code for a VR system.

3. Definitions

We will use the symbol T_{A_B} to denote a transformation from coordinate system B to coordinate system A. This notation is similar to the notation $T_{A\leftarrow B}$ used in (Foley, van Dam, Feiner, & Hughes, 90). We use the term "A_B transform" interchangeably with the symbol T_{A_B}. Points will be represented as column vectors. Thus,

$$\mathbf{p}_A = T_{A\_B} \cdot \mathbf{p}_B \tag{3.1}$$

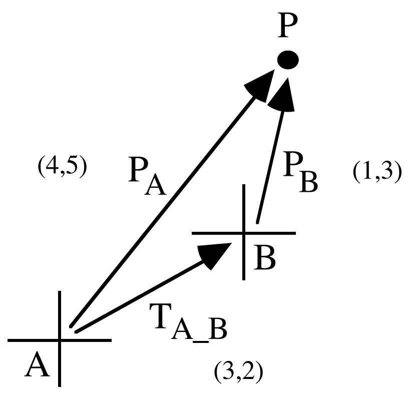
denotes the transformation of the point P_B in coordinate system B by T_{A_B} to coordinate system A. The composition of two transforms is given by:

$$T_{A\_C} = T_{A\_B} \cdot T_{B\_C} \tag{3.2}$$

and transforms a point in coordinate system C into coordinate system A. Note that the subscripts cancel, as in (Pique, 1980), which makes complicated transforms easier to derive. The inverse of a transform is denoted by reversing its subscripts:

$$(T_{A\_B})^{-1} = T_{B\_A} \tag{3.3}$$

Figure 3.1 shows a diagram of a point P and its coordinates in coordinate systems A and B, with some example values given.



$$P_A = T_{A\_B} \cdot P_B$$
$$(4,5) = (3,2) + (1,3)$$

T_{A_B} converts points in B to points in A.
T_{A_B} measures the position of B's origin in A.
The vector runs from A to B.

Figure 3.1. The meaning of transform T_{A_B}.

For simplicity, the transform in Figure 3.1 is limited to translation in 2D. The transform T_{A_B} gives the position of the origin of coordinate system B with respect to coordinate system A, and this matches up with the vector going from A to B in Figure 3.1. However, note that transform T_{A_B} converts the point P from B coordinates ($\mathbf{p}_B$) to A coordinates ($\mathbf{p}_A$) – not from A to B as you might expect from the subscript order.

In general, the transform T_{A_B} converts points from coordinate system B to A, and measures the position, orientation, and scale of coordinate system B with respect to coordinate system A.

4. The VQS Representation

Although the 4x4 homogeneous matrix is the most common representation for transformations used in computer graphics, there are other ways to implement common transformation operations. We introduce here an alternative representation for transforms between 3D coordinate systems which was first implemented for and tailored specifically to the needs of virtual-reality systems.

The *VQS* data structure represents the transform between two 3D coordinate systems as a triple [**v**, **q**, s], consisting of a 3D vector **v**, a unit quaternion **q**, and a scalar *s*. The vector specifies a 3D translation, the quaternion specifies a 3D rotation, and the scalar specifies an amount of uniform scaling (in which each of the three dimensions are scaled by the same factor).

4.1 Advantages of the VQS Representation

The VQS representation handles only rotations, translations, and uniform scaling, which is a subset of the transformations handled by the 4 x 4 homogeneous matrix. It cannot represent shear, non-uniform scaling, or perspective transformations. This is both a limitation and an advantage.

We have found that for the core work in our VR system, translations, rotations and uniform scaling are the only transformations we need. Special cases, such as the perspective transformation, can be handled using 4x4 matrices. For operations such as flying, grabbing, scaling and changing coordinate systems, we have found the VQS representation to be superior for the following reasons:

- The VQS representation separates the translation, rotation, and scaling components from one another, which makes it both convenient and intuitive to change these components independently. With homogeneous matrices, it is somewhat more complex to extract the scaling and rotation portions since these two components are combined.
- Renormalizing the rotation component of the VQS representation is simpler and faster than for homogenous matrices, since the rotation and scale components are independent, and because normalization of quaternions is more efficient than normalization of rotation matrices.
- Uniform scaling is useful for supporting the operations of shrinking and expanding virtual objects and the virtual world without changing their shape.
- The VQS representation is tailored specifically for 3D coordinate systems; not for 2D or higher than 3D. This is because the quaternion component of the VQS data structure represents 3D rotations. Again, this aspect of the VQS representation was motivated by the application to VR systems, which deal exclusively with 3D CSs.
- The advantages of the unit quaternion for representing 3D rotation are described in (Shoemake, 1985), (Funda, Taylor, & Paul, 90) and (Cooke, Zyda, Pratt & McGhee, 1992). Briefly, quaternions have several advantages over rotation matrices and Euler angles:

  - Quaternions are more compact than 3x3 matrices (4 components as opposed to 9) and therefore have fewer redundant parameters.

  - Quaternions are elegant and numerically robust (particularly in contrast to Euler angles, which suffer from singularities).

  - Quaternions represent the angle and axis of rotation explicitly, making them trivial to extract.

  - Quaternions allow simple interpolation to make possible a smooth rotation from one orientation to another; this is complex and problematic with both matrices and Euler angles.

  - Quaternions can be more efficient in computation time depending on the application (the tradeoffs are discussed in the references above), especially when operand fetch time is considered.

  Quaternions are an esoteric and obscure bit of mathematics and are generally not familiar to people from their mathematical schooling, but their appropriateness, simplicity, and power for dealing with 3D rotations have won over many sophisticated users, in spite of their unfamiliarity. The ability to use quaternions to interpolate between rotations is sufficient, by itself, to merit adopting them in the VQS representation.

It may be objected that non-uniform scaling and shear are useful modeling operations, and that the perspective transform must also be a part of any VR system. This is absolutely correct. The UNC VR system uses both representations—4x4 homogeneous matrices are used for modeling and for the last few operations in the viewing transformation, and VQS data structures are used everywhere else. While this may seem awkward at first, keep in mind that the viewing transform is hidden from the user code and, in most applications, so are the modeling operations. Thus, the application code often uses only VQS transformations and is generally simpler, more elegant, and more efficient as a result. In addition, there are certain nonlinear modeling operations (for example, twist) and viewing-transform steps (for example, optical distortion correction) that cannot be handled even by 4x4 matrices, so a hybrid system is often necessary in any case.

4.2 VQS Definitions

The triple [**v**, **q**, s], consisting of a 3D vector **v**, a unit quaternion **q**, and a scalar *s*, represents a transform between two 3D coordinate systems. We write the subcomponents of the vector and quaternion as **v** = (v_x, v_y, v_z) and **q** = $[(q_x, q_y, q_z), q_w]$. The vector specifies a 3D translation, the quaternion specifies a 3D rotation, and the scalar specifies an amount of uniform scaling.

In terms of 4x4 homogeneous matrices, the VQS transform is defined by composing a translation matrix, a rotation matrix, and a scaling matrix:

$$[\mathbf{v}, \mathbf{q}, s] \;=\; M_{translate} \cdot M_{rotate} \cdot M_{scale}$$

$$= \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_wq_z & 2q_xq_z+2q_wq_y & 0 \\ 2q_xq_y+2q_wq_z & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_wq_x & 0 \\ 2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_x^2-2q_y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A complete treatment of quaternions for use in computer graphics is given in (Shoemake, 1985). However, we will briefly describe some aspects of how quaternions can be used to represent 3D rotations.

A unit quaternion **q** = $[(q_x, q_y, q_z), q_w]$ specifies a 3D rotation as an axis of rotation and an angle about that axis. The elements q_x, q_y, and q_z specify the axis of rotation. The element q_w indirectly specifies the angle of rotation θ as

$$\theta \;=\; 2\cos^{-1}(q_w)$$

The formulas for quaternion addition, multiplication, multiplication by a scalar, taking the norm, normalization, inverting, and interpolation are given below, in terms of quaternions **q** and **r**, and scalar α:

$$q + r = \left[\left(q_x, q_y, q_z\right), q_w\right] + \left[\left(r_x, r_y, r_z\right), r_w\right] = \left[\left(q_x + r_x, q_y + r_y, q_z + r_z\right), q_w + r_w\right]$$

$$q * r = \left[\left(q_x, q_y, q_z\right), q_w\right] * \left[\left(r_x, r_y, r_z\right), r_w\right] = \begin{bmatrix} \begin{pmatrix} q_xr_w + q_yr_z - q_zr_y + q_wr_x, \\ -q_xr_z + q_yr_w + q_zr_x + q_wr_y, \\ q_xr_y - q_yr_x + q_zr_w + q_wr_z \end{pmatrix}, \\ -q_xr_x - q_yr_y - q_zr_z + q_wr_w \end{bmatrix}$$

$$\|q\| = \left\|\left(q_x, q_y, q_z\right), q_w\right\| = \sqrt{q_x^{\,2} + q_y^{\,2} + q_z^{\,2} + q_w^{\,2}}$$

$$\alpha \cdot q = \alpha \cdot \left[\left(q_x, q_y, q_z\right), q_w\right] = \left[\left(\alpha \cdot q_x, \alpha \cdot q_y, \alpha \cdot q_z\right), \alpha \cdot q_w\right]$$

$$normalize(q) = \frac{1}{\|q\|} \cdot q$$

$$q^{-1} = \left[\left(q_x, q_y, q_z\right), q_w\right]^{-1} = \frac{1}{\|q\|^2} \cdot \left[\left(-q_x, -q_y, -q_z\right), q_w\right]$$

$$nterp(\alpha, q, r) = normalize((1 - \alpha) \cdot q + \alpha \cdot r)$$

Composing two 3D rotations represented by quaternions is done by multiplying the quaternions. The quaternion inverse gives a rotation around the same axis but of opposite angle. Smooth linear interpolation between two quaternions gives a smooth rotation from one orientation to another.

The rotation of a point or vector **p** by a rotation specified by a quaternion **q** is done by

$$\mathbf{q} * \mathbf{p} * \mathbf{q}^{-1}$$

where the vector **p** is treated as a quaternion with a zero scalar component for the multiplication, and the result turns out to have a zero scalar component and so can be treated as a vector. Using this notation, the VQS transform can be defined more concisely as

$$\mathbf{p'} \;=\; [\mathbf{v}, \mathbf{q}, s] \cdot \mathbf{p} \;=\; s\cdot(\mathbf{q} * \mathbf{p} * \mathbf{q}^{-1}) + \mathbf{v}$$

This is completely equivalent to the earlier definition.

It can be verified that the composition of two VQS transforms can be calculated as

$$\begin{aligned} T_{A\_B} \cdot T_{B\_C} &= [\mathbf{v}_{A\_B}, \mathbf{q}_{A\_B}, s_{A\_B}] \cdot [\mathbf{v}_{B\_C}, \mathbf{q}_{B\_C}, s_{B\_C}] \\ &= [\,(s_{A\_B}\cdot(\mathbf{q}_{A\_B} * \mathbf{v}_{B\_C} * \mathbf{q}_{A\_B}^{-1})) + \mathbf{v}_{A\_B},\; \mathbf{q}_{A\_B} * \mathbf{q}_{B\_C}, s_{A\_B} \cdot s_{B\_C}] \end{aligned}$$

The inverse of a VQS transform is:

$$T_{A\_B}^{-1} = [\mathbf{v}_{A\_B},\; \mathbf{q}_{A\_B},\; s_{A\_B}]^{-1} = [1/s_{A\_B} \cdot (\mathbf{q}_{A\_B}^{-1} * (-\mathbf{v}_{A\_B}) * \mathbf{q}_{A\_B}), \mathbf{q}_{A\_B}^{-1}, 1/s_{A\_B}]$$

We will now move on to describing the transformations making up the visual display computation for VR. We believe that the VQS representation of 3D transforms has advantages, but VQS transforms are not required for VR. In the rest of the paper, it should be understood that, wherever VQS data structures are used, 4x4 homogeneous matrices could have been used instead.

5. Coordinate System Graphs

Many coordinate systems coexist within a VR system. All of these CSs exist simultaneously, and although over time they may be moving with respect to one another, at any given moment, a transform exists to describe the relationship between any pair of them. Certain transforms, however, are given a higher status and are designated the *independent* transforms; all other transforms are considered the *dependent* transforms, and may be calculated from the independent ones. The independent transforms are chosen because they are independent of one another: they are either measured by the tracker, constant due to the rigid structure of the HMD, or used as independent variables in the software defining the virtual world.

We have found it helpful to use a diagram of the coordinate systems and independent transforms between them. A CS diagram for an early version of the UNC VR software was presented in (Brooks, 1989) and a later version in (Robinett & Holloway, 1992). We represent a typical multiple-user VR system with the following graph:

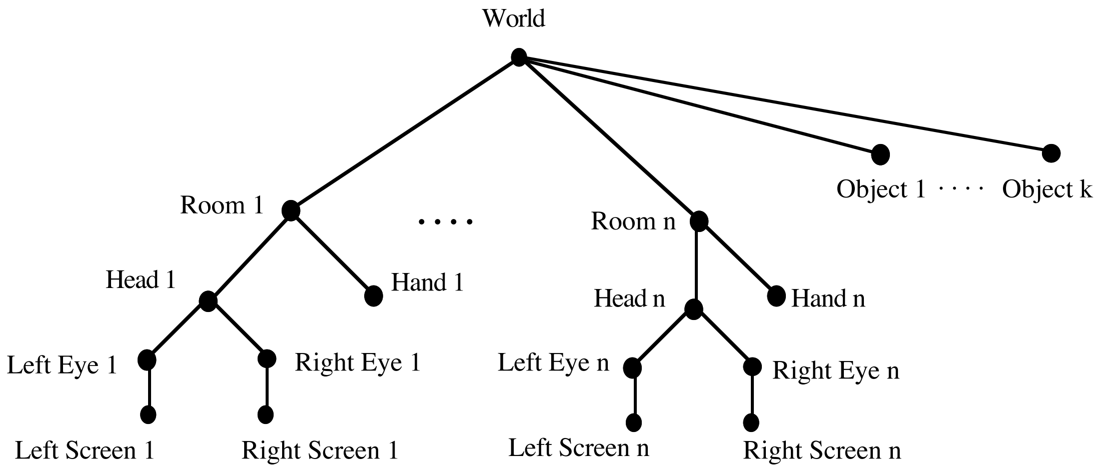


Figure 5.1. Coordinate systems for a multi-user virtual world

Each node represents a coordinate system, and each edge linking two nodes represents a transform between those two CSs. Each user is modeled by the subgraph linking the user's eyes, head, and hand. A transform between any pair of CSs may be calculated by finding a path between corresponding nodes in the graph and composing all the intervening transforms.

This, in a nutshell, is how the display computation for VR works: For each virtual object, a path must be found from the object to each of the screens, and then the points defining the object must be pumped through the series of transforms corresponding to that path. This produces the object's defining points in screen coordinates. This must be done for each screen in the VR system. An object is seen stereoscopically (on two screens) by each user, and in a multi-user system an object may be seen simultaneously from different points of view by different users.

As an example, it may be seen from the diagram that the path to Left Screen 1 from Object 3 is

Left Screen 1, Left Eye 1, Head 1, Room 1, World, Object 3

and thus the corresponding transforms for displaying Object 3 on Left Screen 1 are

$$T_{LS1\_O3} = T_{LS1\_LE1} \cdot T_{LE1\_H1} \cdot T_{H1\_R1} \cdot T_{R1\_W} \cdot T_{W\_O3}$$

As another example, finding a path from Head 1 to Right Screen 2 allows User #2 to see User #1's head.

Note that the CS graph is *connected* and *acyclic*. Disconnected subgraphs are undesirable because we want to express all CSs in screen space eventually; a disconnected subgraph would therefore not be viewable. Cycles in the graph are undesirable because they would allow two ways to get between two nodes, which might be inconsistent.

It is primarily the topology of the CS graph that is significant – it shows which CSs are connected by independent transforms. However, the World CS is drawn at the top of the diagram to suggest that all the virtual objects and users are contained in the virtual world. Likewise, the diagram is drawn to suggest that Head and Hand are contained in Room, that Left Eye and Right Eye are contained in Head, and that each Screen is subordinate to the corresponding Eye.

The independence of each transform in the CS graph can be justified. To have independently movable objects, each virtual object must have its own transform (World_Object) defined in the VR software. Likewise, each user must have a dedicated and modifiable transform (Room_World) to be able to change position, orientation, and scale within the virtual world. (This is subjectively perceived by the user as flying through the world, tilting the world, and scaling the world.) The tracker measures the position and orientation of each user's head and hand (Head_Room, Hand_Room) within the physical room where the tracker is mounted. The user's eyes must have distinct positions in the virtual world to see stereoscopically (Left Eye_Head, Right Eye_Head). The optics and geometry of the HMD define the final transform (Screen_Eye).

We can further characterize transforms as *dynamic* (updated each frame, like the tracker's measurements) or *static* (typically characteristic of some physical, fixed relationship, like the positions of the screens in the HMD relative to the eyes).

There can be many users within the same virtual world, and many virtual objects. The users can see one another if their heads, hands, or other body parts have been assigned graphical representations, and if they are positioned in the same part of the virtual world so as to face one another with no objects intervening. We have represented virtual objects as single nodes in the CS graph for simplicity, but objects with moving subparts are possible, and such objects would have more complex subgraphs.

There are other ways this CS diagram could have been drawn. The essential transforms have been included in the diagram, but it is useful to further subdivide some of the transforms, as we will see in later sections of this paper.

6. The Visual Display Computation

The problem we are solving is that of writing the visual display code for a virtual reality system, with provision that:

- multiple users inhabit the same virtual world simultaneously;
- each user has a stereoscopic display;
- the user's viewpoint is measured by a head tracker;
- the display code matches the geometry of the HMD, tracker, and optics;
- various HMDs and trackers can be supported by changing parameters of the display code;
- the user can fly through the world, tilt the world, and scale the world; and
- the user can grab and move virtual objects.

In this paper, we present a software architecture for the VR display computation which provides these capabilities. This architecture was implemented as the display software for the VR system in the Computer Science Department at the University of North Carolina at Chapel Hill. The UNC VR system is a research system designed to accommodate a variety of models of HMD, tracker, graphics computer, and manual input device. Dealing with this variety of hardware components forced us to create a flexible software system that could handle the idiosyncrasies of many different VR peripherals. We believe, therefore, that the display software that has evolved at UNC is a good model for VR display software in general, and has the flexibility to handle most current VR peripherals.

We present a set of algorithms and data structures for the visual display computation of VR. We note that there are many choices that face the designer of VR display software, and therefore the display code differs substantially among current VR systems designed by different teams. Some of these differences arise from hardware differences between systems, such as the physical geometry of different HMDs, different optics, different size or position of display devices, different geometries for mounting trackers, and different graphics hardware.

However, there are further differences that are due to design choices made by the architects of each system's software. The software designer must decide what data structure to use in representing the transforms between coordinate systems, define the origin and orientation for the coordinate systems used, and define the sequence of transforms that comprise the overall Screen_Object transform, and decide what parameters to incorporate into the display transform.

The VR display algorithm presented in this paper is a general algorithm which can be tailored to most current VR systems by supplying appropriate values for the parameters of the algorithm. For concreteness, we discuss the implementation of this display algorithm on the UNC VR system. The UNC VR software is based on a software library called *Vlib*. Vlib was designed by both authors and implemented by Holloway in early 1991. A brief overview is given in (Holloway, Fuchs & Robinett, 1991).

Vlib was originally written for use with PPHIGS, the graphics library for Pixel-Planes 5 (Fuchs, Poulton, Eyles, Greer, Goldfeather, Ellsworth, Molnar, Turk, Tebbs, & Israel, 1989), the graphics computer in the UNC VR system. However, it was subsequently ported to run on a Silicon Graphics VGX using calls to the GL graphics library. Since Silicon Graphics machines are widely used for VR software, we will describe the GL-based version of Vlib.

6.1 Components of the Visual Display Transform

The Vlib display software maps a point $\mathbf{p}_O$ defined in Object coordinates into a point in Screen coordinates $\mathbf{p}_S$ using this transform:

$$\mathbf{p}_S = T_{S\_E} \cdot T_{E\_H} \cdot T_{H\_R} \cdot T_{R\_W} \cdot T_{W\_O} \cdot \mathbf{p}_O \qquad (6.1)$$

This is consistent with the CS diagram of Figure 5.1. However, there are some complications that make it useful to further decompose two of the transforms above: the Head_Room transform T_{H_R} and the Screen_Eye transform T_{S_E}.

The primary function of the Head_Room transform is to contain the measurement made by the tracker of head position and orientation, which is updated each display frame as the user's head moves around. The tracker hardware measures the position and orientation of a small movable sensor with respect to a fixed frame of reference located somewhere in the room. Often, as with the Polhemus magnetic trackers, the fixed frame of reference is a transmitter and the sensor is a receiver.

The two components of tracker hardware, the tracker's base and the tracker's sensor, have native coordinate systems associated with them by the tracker's hardware and software. If the tracker base is bolted onto the ceiling of the room where the VR system is used, this defines a coordinate system for the room with the origin up on the ceiling and with the X, Y, and Z axes pointing whichever way it was mechanically convenient to mount the tracker base onto the ceiling. Likewise, the sensor is mounted somewhere on the rigid structure of the head-mounted display, and the HMD inherits the native coordinate system of the sensor.

In Vlib, we decided to introduce two new coordinate systems and two new static transforms, rather than use the native CSs of the tracker base and sensor as the Room and Head CSs. This allowed us to choose a sensible and natural origin and orientation for Room and Head space. We chose to put the Room origin on the floor of the physical room and orient the Room CS with X as East, Y as North, and Z as up. We chose to define Head coordinates with the origin midway between the eyes, oriented to match the usual screen coordinates with X to the right, Y up, and Z towards the rear of the head.

Thus, the Head_Room transform is decomposed into

$$T_{H\_R} = T_{H\_HS} \cdot T_{HS\_TB} \cdot T_{TB\_R} \qquad (6.2)$$

where the tracker directly measures the Head-Sensor_Tracker-Base transform T_{HS_TB}. The mounted position of the tracker base in the room is stored in the Tracker-Base_Room transform T_{TB_R}, and the mounted position of the tracker sensor on the HMD is stored in the Head_Head-Sensor transform T_{H_HS}.

When using more than one type of HMD, it is much more convenient to have Head and Room coordinates be independent of where the sensor and tracker base are mounted. The T_{H_HS} and T_{TB_R} transforms, which are static, can be stored in calibration files and loaded at run-time to match the HMD being used, allowing the same display code to be used with all HMDs. If a sensor or tracker is remounted in a different position, it is easy to change the calibration file. To install a new tracker, a new entry is created in the tracker calibration file. Without this sort of calibration to account for the tracker mounting geometry, the default orientation of the virtual world will change when switching between HMDs with different trackers.

The other transform which it is convenient to further decompose is the Screen_Eye transform T_{S_E}, which can be broken down into

$$T_{S\_E} = T_{S\_US} \cdot T_{US\_N} \cdot T_{N\_E} \qquad (6.3)$$

T_{S_US} is the *optical distortion correction transformation*, T_{US_N} is the *3D viewport transformation* described in (Foley, van Dam, Feiner, & Hughes, 1990), and T_{N_E} is the *normalizing perspective transformation*. The 3D viewport transformation is the standard one normally used in computer graphics. The perspective transform is slightly unusual in that it must, in general, use an off-center perspective projection to match the geometry of the HMD being used. The details of this are discussed in a later section. A transformation to model the optics of the HMD is something not normally encountered in standard computer graphics, and it causes some problems which are discussed in more detail later.

Plugging in the decomposed transforms of Equations (6.2) and (6.3) into the overall display transform of (6.1) gives

$$T_{S\_O} = T_{S\_US} \cdot T_{US\_N} \cdot T_{N\_E} \cdot T_{E\_H} \cdot T_{H\_HS} \cdot T_{HS\_TB} \cdot T_{TB\_R} \cdot T_{R\_W} \cdot T_{W\_O} \qquad (6.4)$$

This is the visual display transform used by the UNC VR software. Note that the leftmost five transforms are static and can therefore be precomputed once to yield T_{S_HS}.

Table 6.1 below lists each of the transformations involved in the overall display transform. Note that several instances of each transform in the table must be maintained to allow for multiple objects, multiple users, and the two eyes of each user. Example values for these transforms are given in Table 8.1.

Symbol	**Coordinate Systems**	**Instances**	**Function**	**Static / Dynamic**
T_{W_O}	Object to World	1 per object	position of object in world (changes when object moves)	dynamic
T_{R_W}	World to Room	1 per user	position of room in world (changes when flying, etc.)	dynamic
T_{TB_R}	Room to Tracker Base	1 per user	position of tracker in room	static
T_{HS_TB}	Tracker Base to Head Sensor	1 per user	measurement of head position and orientation by tracker	dynamic
T_{H_HS}	Head Sensor to Head	1 per user	position of sensor on HMD	static
T_{E_H}	Head to Eye	2 per user	positions of left and right eyes	static
T_{N_E}	Eye to Normalized	2 per user	off-center perspective projection	static
T_{US_N}	Normalized to Undistorted Screen	2 per user	convert to device coordinates	static
T_{S_US}	Undistorted Screen to Screen	2 per user	optical distortion correction	static

Table 6.1. Component transforms of the visual display transform in VR

6.2 Coordinate System Definitions

Using transforms between coordinate systems in a VR system requires that the various CSs be precisely defined. A standard orthogonal coordinate system is completely specified by giving its origin, its orientation, and its units. Table 6.2 gives the CSs defined by Vlib. Vlib is not a graphics library and therefore defines only the coordinate systems going from Object space to Eye space. The remaining low-level coordinate systems are defined in the graphics library being used.

symbol	name	origin	orientation	units

O	Object	center of object	X = right
			Y = front
			Z = top
				same as World, unless object was scaled

W	World	initially on floor directly underneath tracker base	same as Room unless world is tilted	same as Room unless world is scaled

R	Room	lower southwest corner of room	X = east
			Y = north
			Z = up
				meters

TB	Tracker Base	center of base	depends on mounting location	meters
HS	Head Sensor	center of sensor	depends on mounting location	meters

H	Head	mid-point between user's eyes	X = right
			Y = up
			Z = backward
				meters

E	Eye	center of pupil of eye	X = right
			Y = up
			Z = backward
				meters

Table 6.2. Vlib coordinate systems

The transforms between the CSs listed above are all represented in Vlib by the VQS representation, since translation, rotation, and uniform scaling are all that are needed.

The transforms going from Eye space to Screen space are handled by the graphics library supporting the graphics hardware of the VR system. These transforms include off-center perspective projection, optical distortion correction, and mapping to screen coordinates. The coordinate systems for SGI's GL are listed in Table 6.3.

symbol	name	origin	orientation	units

E	Eye	center of pupil of eye	X = right
			Y = up
			Z = backward
				meters

C	Clip	same as above	X = right
			Y = up
			Z = forward
				meters

N	Normalized	same as above	X = right
			Y = up
				normalized

W	Window	same as above	X = right
			Y = up
				pixels

US	Undistorted Screen	lower left corner of viewport	X = right
			Y = up
				pixels

S	Screen	lower left corner of viewport	X = right
			Y = up
				pixels

Table 6.3. GL coordinate systems (Silicon Graphics 91)

The US (Undistorted Screen) coordinate system above requires explanation. It is not officially supported by GL. It is included in the table to indicate the point at which nonlinear image predistortion to compensate for optical distortion could be done: namely, in the step from US to S coordinates.

7. Discussion of Component Transforms

We now discuss each of the component transforms of the full visual display transform for VR, as implemented in UNC's Vlib software, running on top of GL on a Silicon Graphics VGX. The complete display transform, again, is

$$T_{S\_O} = T_{S\_US} \cdot T_{US\_N} \cdot T_{N\_E} \cdot T_{E\_H} \cdot T_{H\_HS} \cdot T_{HS\_TB} \cdot T_{TB\_R} \cdot T_{R\_W} \cdot T_{W\_O} \qquad (7.1)$$

In discussing a transformation between two coordinate systems A and B, it is easy to get confused as to whether the transform should be measured from A to B, or from B to A. In the following sections, keep in mind that the transform T_{A_B} contains the position, orientation, and scale of coordinate system B as measured from coordinate system A, as was discussed earlier in Section 3.

7.1 World_Object Transform (Position of Object in Virtual World)

The World_Object transform $T_{W\_O} = [\mathbf{v}_{W\_O}, \mathbf{q}_{W\_O}, s_{W\_O}]$ determines the position, orientation, and size of each object in the virtual world. Each object has its own instance of this transform, which permits each object to be independently moved, rotated, or scaled. The vector $\mathbf{v}_{W\_O}$ defines the object's position, the quaternion $\mathbf{q}_{W\_O}$ defines its orientation, and the scalar s_{W_O} defines its size.

7.2 Room_World Transform (Position of User in Virtual World)

The Room_World transform $T_{R\_W} = [\mathbf{v}_{R\_W}, \mathbf{q}_{R\_W}, s_{R\_W}]$ determines the position, orientation, and size of each user in the virtual world. Each user has a dedicated instance of this transform, and this permits users to have independent locations, orientations, and scales within the virtual world. In a multi-user virtual world, this means that different users can fly through the world independently of one another. Similarly, different users can also see the virtual world from different orientations or different scale factors.

The vector $\mathbf{v}_{R\_W}$ defines the user's position in the virtual world, and it may be incrementally modified to cause flying through the world to occur.

The quaternion $\mathbf{q}_{R\_W}$ defines the user's orientation, and can be modified to tilt the entire virtual world to a new orientation. Because the force of gravity constantly reminds the user of the direction of real-world down, the user perceives the virtual world to be turning, rather than that his or her own body is turning within the virtual world.

The scalar s_{R_W} defines the user's size within the virtual world. This value can be multiplied each frame by a constant slightly greater or less than 1 to cause the virtual world to shrink or expand.

The precise quantities represented by $\mathbf{v}_{R\_W}$, $\mathbf{q}_{R\_W}$, and s_{R_W} are the position, orientation, and scale of World coordinates with respect to Room coordinates. Exactly how these variables must be modified to implement the operations of flying through a virtual world, tilting the world, scaling the world, and grabbing virtual objects is discussed in detail in (Robinett & Holloway, 1992).

7.3 Tracker-Base_Room Transform (Mounting Position of Tracker Base)

The Tracker-Base_Room transform $T_{TB\_R} = [\mathbf{v}_{TB\_R}, \mathbf{q}_{TB\_R}, 1]$ describes the position and orientation of the tracker base (often a transmitter) within the physical room where the VR system is set up. This value is stored in a calibration file for the tracker currently being used. Note that the scale factor is required to be 1 for this transform.

To be precise, the Tracker-Base_Room transform T_{TB_R} contains the position and orientation of Room coordinates as measured from Tracker-Base coordinates. When relocating the tracker, it is usually most convenient to measure the position and orientation of the tracker base with respect to the fixed coordinate system of the room, and then calculate the inverse transform as the value for T_{TB_R}.

7.4 Head-Sensor_Tracker-Base Transform (Measurement by Tracker)

The Head-Sensor_Tracker-Base transform $T_{HS\_TB} = [\mathbf{v}_{HS\_TB}, \mathbf{q}_{HS\_TB}, 1]$ holds the inverse of the measurement of head position and orientation most recently read from the tracker. Most trackers provide the position and orientation of a sensor with respect to the tracker base (T_{TB_HS}).

Not all trackers deliver orientation as a quaternion, so the tracker driver software may have to do a conversion from a 3x3 matrix or from Euler angles. The driver may also have to perform a scaling to convert the position measurement to the required units of meters.

Since the tracker only measures position and orientation, the scale factor for this transform is required to be 1.

The tracker driver outputs a vector and a quaternion. The vector defines the position of the sensor with respect to the tracker's base and the quaternion defines the sensor's orientation with respect to the tracker's base. This vector and quaternion read from the tracker comprise a transform which must be inverted to get the Head-Sensor_Tracker-Base transform T_{HS_TB}. Note that, as discussed in Section 4, inverting a VQS transform is not equivalent to simply inverting its vector, quaternion, and scalar components.

7.5 Head_Head-Sensor Transform (Mounting Position of Sensor on HMD)

The Head_Head-Sensor transform $T_{H\_HS} = [\mathbf{v}_{H\_HS}, \mathbf{q}_{H\_HS}, 1]$ describes the position and orientation of where the head sensor is mounted on the HMD. These measurements are with respect to the Head coordinate system, centered at the midpoint between the eyes. The vector $\mathbf{v}_{H\_HS}$ defines the position of the sensor with respect to the Head CS and the quaternion $\mathbf{q}_{H\_HS}$ defines the sensor's orientation in the Head CS. The scale factor is required to be 1 for this transform also.

This value is stored in a calibration file for the HMD currently being used.

7.6 Eye_Head Transform (Separation of the Eyes)

For each user, the Eye_Head transform has two instances, one for each eye. These two transforms position the viewpoints of the user's eyes at slightly separated points within the virtual world, thus allowing stereoscopic vision through the HMD. Figure 7.1 shows a diagram of this.

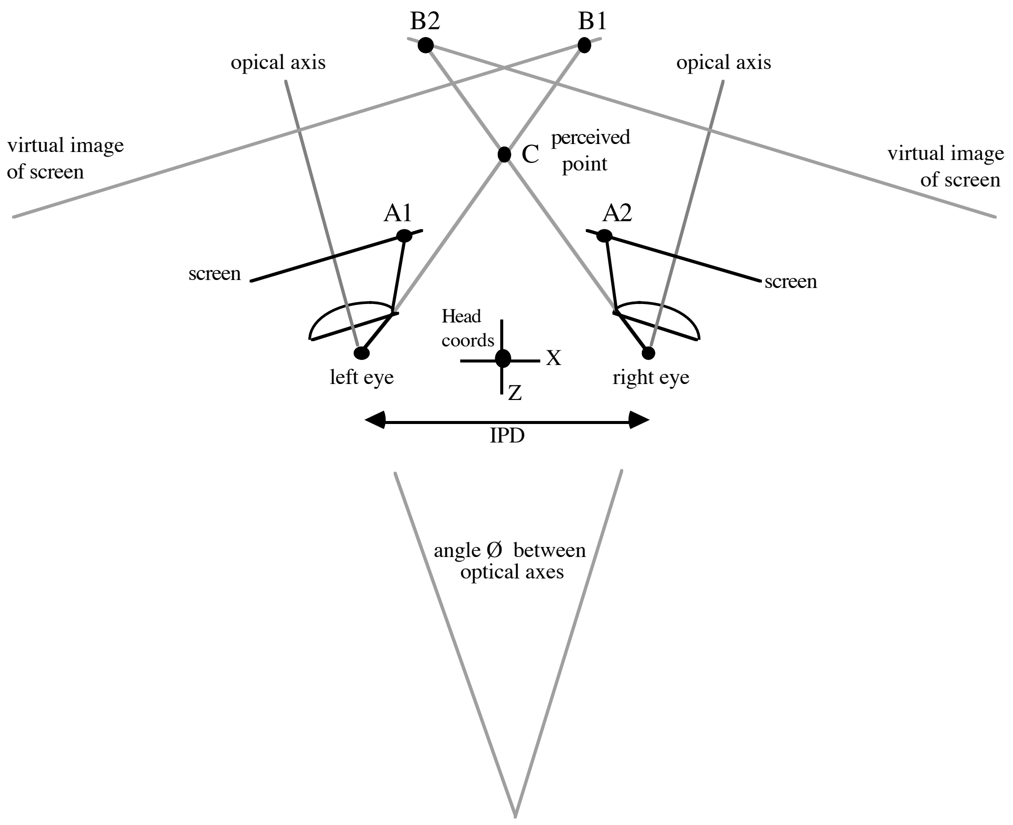


Figure 7.1. Stereoscopic optics model for an HMD

The Left-Eye_Head transform T_{LE_H} = [$\mathbf{v}_{LE\_H}$, $\mathbf{q}_{LE\_H}$, 1] and Right-Eye_Head transform T_{RE_H} = [$\mathbf{v}_{RE\_H}$, $\mathbf{q}_{RE\_H}$, 1] describe the positions of the eyes with respect to the Head coordinate system centered at the midpoint between the user's eyes. (Because we are transforming from Head space to Eye space, T_{E_H} actually describes the position of the Head CS with respect to the Eye CS, not vice versa.) Thus, for a given interpupillary distance (IPD), using the Head coordinate system described in Table 6.2 we have

$$\mathbf{v}_{LE\_H} = (+IPD/2, 0, 0)$$
$$\mathbf{v}_{RE\_H} = (-IPD/2, 0, 0)$$

There are considerable individual differences among the IPDs of adults, with 95% falling in the range from 49 to 75 mm (Woodson, 1981). Wide-eyed and narrow-eyed people will perceive the same scene in an HMD to have different absolute sizes and distances (Rolland, Ariely, & Gibson, 1993). To avoid this problem, the IPD for each user needs to be measured (with a device such as an optician uses) and the user's IPD needs to be entered into a calibration file specific to that user. Since most people wear eyeglasses or contact lenses customized to their vision and facial geometry, it should not be surprising that HMDs and their display software need to be customized for each user.

However, it is often not practical or not worth the trouble to change the IPD setting each time a new user dons the HMD. In practice, the IPD of the UNC HMDs often remains set at an average value of 62 mm.

It is important to emphasize that the geometrical model used in the graphics software must recognize that the eye focuses on the *virtual image* of the screen, not the screen itself. The graphics software for some HMDs erroneously ignores the optics, and models the eyes as focusing directly on screens a few centimeters away (VPL, 1989). If this were accurate, small displacements of the pupil from the assumed center of projection would cause large distortions in the perceived image. Rotation of the eye to gaze at different points in the image, and also the variation in IPD from one user to another, both cause the pupil to be displaced from its assumed location. But since the eyes in fact focus on distant virtual images of the screens, these small displacements of the pupil have a relatively small effect on the perceived image.

The orientation of eye space should match that of the optical system in the HMD. If the HMD being used has parallel optical axes for the two eyes, then the orientation of Left-Eye and Right-Eye space match that of Head space. In this case, the two quaternions will be the identity quaternion $\mathbf{q}_{\text{ident}}$ = [(0,0,0),1], corresponding to zero rotation.

However, some HMDs use diverged optical axes to obtain a wider field of view. The parameter Ø describes the angle between the optical axes, giving divergence angles of -Ø/2 and +Ø/2 around the Y-axis for the two eyes. This translates into quaternions as

$$\mathbf{q}_{\text{LE\_H}} = [\,(0, \sin(+Ø/4), 0),\ \cos(+Ø/4)]$$
$$\mathbf{q}_{\text{RE\_H}} = [\,(0, \sin(-Ø/4), 0),\ \cos(-Ø/4)]$$

As another example of how the Eye_Head transform can be used, an HMD was built at UNC in which one display device (an LCD display for the left eye) was upside-down due to mounting constraints. For correct operation, the left image had to be displayed upside down. This was accomplished by composing a 180˚ rotation around the Z-axis with the $\mathbf{q}_{\text{LE\_H}}$ quaternion.

This concludes the list of Vlib transforms, all of which use the VQS representation. To get from Eye space to Screen space, several different representations of the transforms are necessary. The following transforms are described as they are implemented for the GL-based version of Vlib.

7.7 Normalized_Eye Transform (Perspective Transform)

The perspective projection used in an HMD must match the field of view of the HMD, and must position the eyes correctly relative to the screens' virtual images. This usually requires an off-center perspective projection, since the user's eye is generally not lined up with the center of the screen in an HMD.

Figure 7.2 below shows some of the important parameters in eye space pertaining to the perspective projection and is similar to that used in (Foley, van Dam, Feiner, & Hughes, 1990).

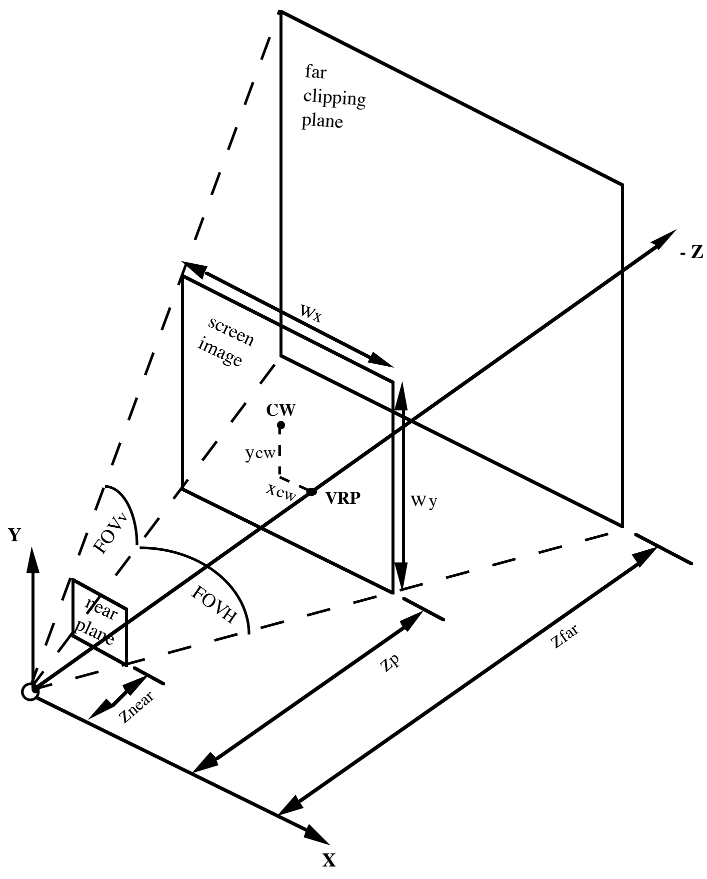


Figure 7.2. Viewing parameters for one eye using an off-center perspective projection

In this diagram, the screen image defines the plane of projection and the eye is at the origin looking along the -Z axis. We have chosen the eye coordinate system so that the -Z axis is parallel to the optical axis, and is perpendicular to the screen image and intersects it at the view reference point (VRP) at $z = z_p$. The center of the screen image or *viewing window* is denoted by CW and is not generally at the VRP. The distances x_{cw} and y_{cw} give the offsets to the screen-image center relative to the VRP. The viewing window size is just the size of the screen image and is denoted by w_x and w_y. The near and far clipping plane distances z_{near} and z_{far} determine when objects are too close or too distant for display.

Although this is a complete model for a single eye, there is a complication introduced by some stereoscopic displays. Because the VRP is defined relative to the eye, as the IPD changes for different users, the VRP's horizontal placement changes as well. If the display does not have a physical IPD adjustment, then the VRP moves laterally with respect to CW. In this case x_{cw} changes with the IPD.

The figure below shows the situation for the left eye.

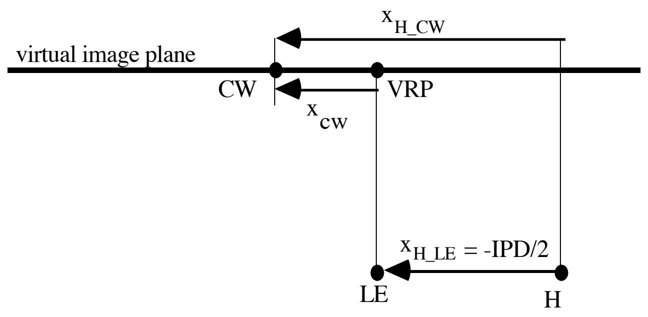


Figure 7.3. Dependence of x_{cw} on IPD (Left eye)

Here, x_{H_CW} is the X coordinate of the left screen's CW relative to Head space (the value for this must be derived from the specifications for the HMD) and x_{H_LE} is the X coordinate of the left eye relative to Head space and for the left eye, which is just:

$$x_{H\_LE} = \frac{-IPD}{2}$$

Thus,

$$x_{CW} = x_{H\_CW} - x_{H\_LE} = x_{H\_CW} + \frac{IPD}{2} \qquad \text{(Left eye)}$$

The situation for the right eye is similar:

$$x_{CW} = x_{H\_CW} - x_{H\_RE} = x_{H\_CW} - \frac{IPD}{2} \qquad \text{(Right eye)}$$

Note that x_{H_CW} is negative for the left eye and positive for the right.

x_{cw} and y_{cw} can also be used to correct for misalignments of the HMD hardware. For example, in the VPL EyePhone, an accidental displacement of the display device in the object plane by 1 mm results in an angular error of roughly 1.5°. Errors in vertical placement on the image plane produce vertical angular offsets, and can result in corresponding pixels in the left and right displays being vertically misaligned. This is called *dipvergence*. Dipvergence can be corrected by adjusting y_{cw} to reflect this offset in the computer graphics model. Similarly, horizontal placement error can be fixed by adjusting x_{cw}.

The calculation of the field of view is also more complicated for off-center projections. Figure 7.4 shows the relationship between the horizontal field-of-view angles and the off-center projection parameters.

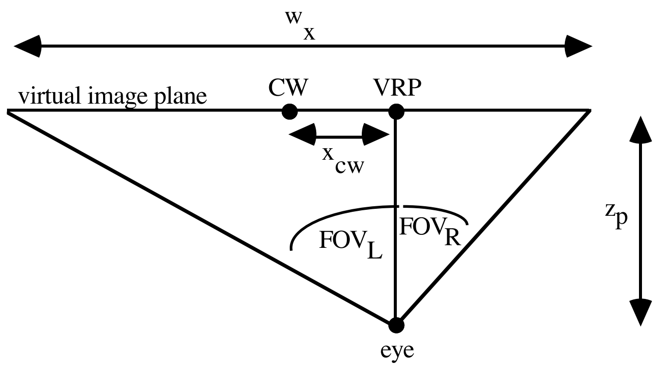


Figure 7.4. Relation of horizontal FOV to off-center projection parameters

Because the eye is off-center with respect to the screen, the left and right components, FOV_L and FOV_R, of the horizontal field of view FOV_H are not equal and must be calculated separately.

$$FOV_L = \tan^{-1}\left(\frac{\frac{w_x}{2} - x_{cw}}{|z_p|}\right)$$

$$FOV_R = \tan^{-1}\left(\frac{\frac{w_x}{2} + x_{cw}}{|z_p|}\right)$$

$$FOV_H = FOV_L + FOV_R$$

The calculation is similar for the vertical field of view FOV_V, which is divided into top and bottom angles.

$$FOV_T = \tan^{-1}\left(\frac{\frac{w_y}{2} + y_{cw}}{|z_p|}\right)$$

$$FOV_B = \tan^{-1}\left(\frac{\frac{w_y}{2} - y_{cw}}{|z_p|}\right)$$

$$FOV_V = FOV_T + FOV_B$$

It is important to note that although the field-of-view angle is an important parameter for characterizing a head-mounted display, it is not the best choice of parameter for specifying off-center projections (for reasons which are beyond the scope of this paper). For example, the GL library has two different calls for setting up the perspective transformation: the *perspective* call is used for on-center projections and takes the field of view as a parameter, whereas the *window* call (discussed below) is intended for off-center projections and does not use the field of view as a parameter.

Now that we have discussed the meanings and proper values for the parameters in specifying the perspective transform, we can move on to a discussion of how these values are used.

The GL coordinate systems listed in Table 6.3 are not usually accessed directly. Normal applications do not need to compose a transformation between each pair of CSs in the table. Rather, a few GL calls are used to set up the transforms from Eye space to Screen space.

The *window* call sets up the normalizing perspective transformation. The syntax is:

window(left, right, bottom, top, near, far)

where *near* and *far* are the positive distances from the eyepoint to the near and far clipping planes. Since our Eye CS is right-handed with the -Z axis away, we will need to negate *near* and *far* before using them in equations for which sign is important.

Note that there is no specification of the projection-plane distance z_p in the *window* call. This is because GL assumes that the window is inscribed in the near clipping plane. Therefore, the window specification describing the virtual image of the screen must be projected onto the near clipping plane (which usually must be closer to the eye than the virtual screen-image distance). Thus we have:

$$\mathit{left} = \frac{-\mathrm{near}\cdot(x_{cw} - w_x/2)}{z_p} \qquad \mathit{right} = \frac{-\mathrm{near}\cdot(x_{cw} + w_x/2)}{z_p}$$

$$\mathit{bottom} = \frac{-\mathrm{near}\cdot(y_{cw} - w_y/2)}{z_p} \qquad \mathit{top} = \frac{-\mathrm{near}\cdot(y_{cw} + w_y/2)}{z_p}$$

(*near* has been negated to match the sign convention for z_p).

The matrix generated is (Silicon Graphics, 91):

$$T_{N\_E} = \begin{bmatrix} \frac{2\cdot\mathrm{near}}{\mathrm{right}-\mathrm{left}} & 0 & \frac{\mathrm{right}+\mathrm{left}}{\mathrm{right}-\mathrm{left}} & 0 \\ 0 & \frac{2\cdot\mathrm{near}}{\mathrm{top}-\mathrm{bottom}} & \frac{\mathrm{top}+\mathrm{bottom}}{\mathrm{top}-\mathrm{bottom}} & 0 \\ 0 & 0 & -\frac{\mathrm{far}+\mathrm{near}}{\mathrm{far}-\mathrm{near}} & -\frac{2\cdot\mathrm{far}\cdot\mathrm{near}}{\mathrm{far}-\mathrm{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

This matrix performs the shear required for off-center projections, as well as scaling and translating the viewing frustum into the unit cube such that $-1 \le x,y,z \le 1$ after the division by *w*.

7.8 Undistorted-Screen_Normalized Transform (Convert to Pixel Coordinates)

The T_{US_N} transformation converts from the normalized viewing volume just described to device coordinates. The *viewport* call of GL implements T_{US_N} and is straightforward:

viewport$(x_{min}, x_{max}\ y_{min}, y_{max})$

where all parameters are pixel coordinates.

The scaling/translation matrix for X and Y that accomplishes this is:

$$T_{US\_N} = \begin{bmatrix} \frac{x_{max}-x_{min}}{2} & 0 & x_{min}+\frac{x_{max}-x_{min}}{2} \\ 0 & \frac{y_{max}-y_{min}}{2} & y_{min}+\frac{y_{max}-y_{min}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix translates the normalized window (after projection) so that the lower left corner is at the origin, then scales it to fit into the given viewport, and finally translates the scaled window so that its lower left corner is at x_{min}, y_{min}. The left- and right-eye views are typically mapped to different viewports for single-frame-buffer systems and then scanned out as separate video signals for display. For systems without distortion correction, this is the final transformation.

7.9 Screen_Undistorted-Screen Transform (Optical Distortion Correction)

Most HMDs (particularly those with wide fields of view) have some degree of optical distortion, which is a non-linear warping of the virtual screen image. This optical aberration can be minimized through careful optical design, electronic prewarping of the image in the display circuitry, or by correcting it in the rendering process. Optical correction has the advantage of not reducing the frame rate, but is not always feasible due to other constraints, such as cost, weight, and minimization of other optical aberrations[1]. Electronic correction does not reduce the frame rate either, but isn't available or feasible for many systems. Thus, although we would prefer to correct the distortion either optically or electronically, these are not always options, and we are forced to either live with it or correct it in the rendering process.

The Virtual Research *Flight Helmet* (which uses the LEEP optics) is a typical case in point. It has significant optical distortion and we know of no system for correcting it electronically. Without correction, lines that fall near the edge of the field of view are noticeably curved. Correcting the distortion in the rendering process requires predistorting the image by a function which is the inverse of the optical distortion function. If this is done correctly, the final image will appear undistorted. The problem with this approach, of course, is that it is computationally expensive, and tends to reduce the frame rate significantly. For this reason, most current systems that have significant distortion (including most of the systems in daily use at UNC) simply ignore the problem. We believe, however, that accurate rendering of scenes in VR will become more important in the future as precision tasks are undertaken with HMDs. With see-through HMDs in particular, the need to accurately register virtual objects with the real world will demand accurate rendering (Janin, Mizell & Caudell, 1993). What follows is a brief description of the distortion problem and one model for correcting it in software, with pointers to other papers on the subject.

As detailed in (Robinett & Rolland, 1992), in systems with optical distortion, the magnification of a point in the image is a function of its distance from the optical axis. If we neglect higher-order terms, this aberration can be modeled to third order as:

$$r_v = m\, r_S + k\,(m\, r_S)^3 \qquad (7.9.1)$$

1 It can be done, however. For example, the sales literature for the CAE FOHMD quotes its distortion as less than 1.5% (CAE, 1986).

where r_v is the radial distance to the displayed point in image space, m is the *paraxial*[2] magnification of the optical system, r_s is the radial distance to the point in screen space, and k is the third-order coefficient of optical distortion. This assumes that the optical system is radially symmetric around an optical axis, which is true for the LEEP optics used in the Flight Helmet, but is not true of all optical systems. Figure 7.9.1 shows a simple model of the optics for a single eye in an HMD.

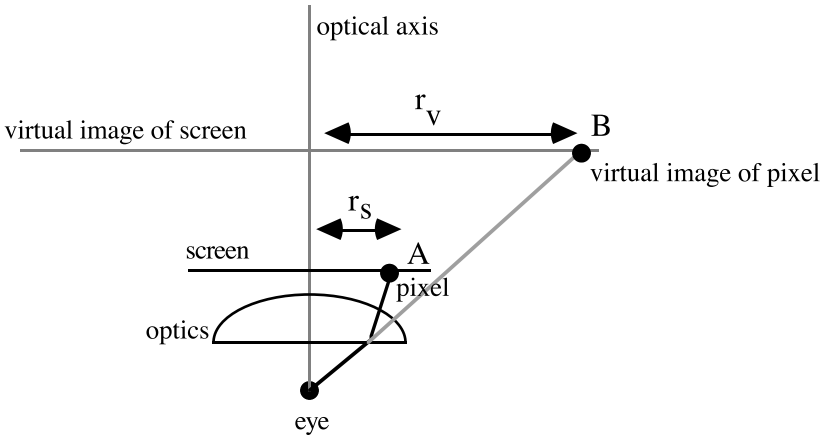


Figure 7.9.1. Single-eye optics model

If k is positive, the magnification increases for off-axis points, and the aberration is called *pincushion distortion* (Figure 7.9.2); if k is negative, the magnification decreases, and it is called *barrel distortion.* Pincushion distortion is more common in HMD systems and will be assumed in what follows.

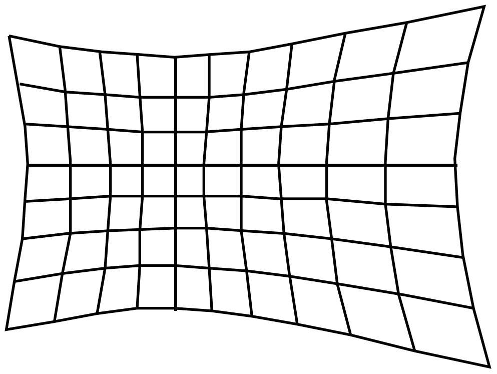


Figure 7.9.2. Pincushion distortion

Note that if $k = 0$, there is no distortion and the virtual image of the screen seen by the user is just a linear magnification of the screen itself. This is what the graphics model typically assumes. The

[2] This term refers to the first-order model for optics which assumes that rays strike the lens at small angles. This approximation breaks down in real systems with finite apertures, resulting in *optical aberrations*, one of which is distortion.

problem with distortion is not only that the image is warped, but also that the scale (i.e., magnification) is different as we move out from the center of the image. Thus, if we set our window parameters w_x and w_y to match the corners of the distorted image, the scale will be right at the corners and wrong in the center; i.e., objects in the center will be scaled down relative to their real sizes, since we used the inflated scale from the image corner (this can also introduce error into the projection window offsets (x_{cw} and y_{cw}) for systems with screens not centered on the optical axes). On the other hand, if we use the paraxial magnification to determine the window parameters, object sizes will be correct in the center but will be stretched out in the periphery. On the whole, using paraxial or near-paraxial values seems to introduce less error than using the distorted values. The bottom line, though, is that there is no good way to approximate a cubic function (the distortion) with a linear function (linear scaling), and if the distortion is not corrected, objects will appear grow and shrink depending on their location in screen space.

Computing the inverse of the distortion function can be done a number of ways. A very accurate but time-consuming method would be to use a ray-tracing program (such as Code V) to compute the distorton and its inverse for a set of r_s values, which could then be interpolated to find the required predistortion for any value of r_s. Another approach is to use the exact closed-form solution to the third-order equation, as is done in (Rolland & Hopkins, 1993). Finally, a third-order approximation to the inverse gives a reasonable fit for many systems, and is described in (Robinett & Rolland, 1992). The second method has been implemented at UNC for Pixel-Planes 5 by Anselmo Lastra, Jannick Rolland, and Terry Hopkins. We present the third method because of its algorithmic simplicity.

Predistortion is a 2D image warp that moves a point on the screen (x_s,y_s) to a new position (x_d, y_d). This warping can either be applied to polygon vertices or to each pixel; the efficiencies and complexities of each approach are beyond the scope of this paper.

In order to simplify the calculations, we can re-express Equation 7.9.1 in the following way:

$$r_{vn} = r_{sn} + k_n r_{sn}^3 \tag{7.9.2}$$

Here, r_{vn} is the normalized radius in image space, r_{sn} is the normalized radius in screen space, and k_n is the normalized coefficient of optical distortion. The normalized coefficient of distortion gives the percentage distortion at some image radius r_{norm}, which is usually chosen to be the distance from the optical axis to one of the edges of the image. Because k_n is defined in terms of r_{norm} and because the choice of r_{norm} is somewhat arbitrary, one can have different values of k_n that describe the same system. The parameter k (from Equation 7.9.1) does not have this dependency, but is not as intuitive for describing distortion.

The inverse of Equation 7.9.2 can be approximated with a third-order polynomial as shown in the following algorithm:

$(x_s, y_s) = (x - x_{axis}, y - y_{axis})$	express the point (x, y) relative to optical axis
$r_s = \sqrt{x_s^2+y_s^2}$	calculate radius
$r_{sn} = \frac{r_s}{r_{norm}}$	normalize
$r_{pdn} = r_{sn} + k_{pd} \cdot r_{sn}^3$	apply inverse distortion (Note: $k_{pd} < 0$)
$r_{pd} = r_{norm} \cdot r_{pdn}$	map back to pixel units
$d = \frac{r_{pd}}{r_s}$	calculate radial scaling factor for this point
$(x_{sv}, y_{sv}) = (d \cdot x_s, d \cdot y_s)$	scale vector centered at optical axis
$(x_d, y_d) = (x_{sv} + x_{axis}, y_{sv} + y_{axis})$	translate back to screen coordinates

Thus, the parameters of the image warp algorithm are:

- the position of the optical axis in screen coordinates: x_{axis}, y_{axis}
- the normalizing radius in pixels: r_{norm}
- the normalized distortion coefficient and the normalized predistortion coefficient: k_n and k_{pd}

These parameters depend on the specifications of the optics and the positioning of the display device relative to the optics.

This algorithm shrinks the image in a non-linear fashion so that when it is distorted by the optics, it will match the paraxial model. In this case, w_x,w_y and x_{cw}, y_{cw} should be set to their paraxial values since predistortion will cancel out any change in these values induced by the distortion (both values for these parameters are given in the table in the next section). Since this scaling of the image leaves some of the frame buffer unused, an alternative method is to scale the predistorted image so that it fills the frame buffer as much as possible and to use the distorted window extents and offsets.

8. Parameters of the Display Transformation

The numerical values of the parameters of the display code tailor the code to a particular VR system. We give the display parameters of the UNC VR system as an example.

In particular, Table 8.1 gives the complete specification of all of the transformation parameters for the Vlib GL version for use with the Virtual Research *Flight Helmet*. Note that the table includes only the static parameters which are known at startup time (a complete listing was given in Table 6.1). Also, some of the transforms in listed have been inverted to their more intuitive form for the sake of clarity.

Transform	**Parameters**	**Value in UNC VR system**	**Description of Parameters**
T_{W_O}	$\mathbf{v}_{W\_O}$ $\mathbf{q}_{W\_O}$ s_{W_O}	(object poses stored in model data)	object's position in world object's orientation in world object's scale in world
T_{W_R}	$\mathbf{v}_{W\_R}$ $\mathbf{q}_{W\_R}$ s_{W_R}	initially (0,0,0) initially [(0,0,0), 1] initially 1	room's position in world room's orientation in world room's scale in world
T_{R_TB}	$\mathbf{v}_{R\_TB}$ $\mathbf{q}_{R\_TB}$	(2.5, 2.0, 2.0) (meters) [(0.5, 0.5, -0.5), -0.5]	tracker base's position within room tracker base's orientation within room
T_{H_HS}	$\mathbf{v}_{H\_HS}$ $\mathbf{q}_{H\_HS}$	(0.0, 0.19, 0.03) (meters) [(0.5, 0.5, -0.5), 0.5]	head sensor's position on the HMD head sensor's orientation on the HMD
T_{H_E}	$\mathbf{v}_{H\_E}$	Left: (-IPD/2, 0, 0) Right: (IPD/2, 0, 0) with IPD = 0.062 m	position of eye CS relative to head CS (in meters)
	$\mathbf{q}_{H\_E}$	Left: [(0, 0, 0), 1] Right: [(0, 0, 0), 1] since Ø = 0	rotation of eye CS relative to head CS
T_{N_E}	FOV_h FOV_v	77° w/ distortion (66.2° paraxial) 60.8° w/ distortion (53.5° paraxial)	horizontal monocular field of view vertical monocular field of view (assuming an eye relief of 25mm)
	z_p	-1.18 m	distance from eye to virtual image of screen (assuming an eye relief of 25mm)
	x_{cw}	Left: –0.187m (paraxial) –0.190m (w/ distortion) Right: 0.187m (paraxial) 0.190m (w/ distortion)	projection window center offsets (assuming IPD = 0.062m)
	y_{cw}	Left: 0 m Right: 0 m	
	w_x w_y	1.57m paraxial, 2.09m w/ distortion 1.19m paraxial, 1.38m w/ distortion	projection window extents
	z_{near}, z_{far}	0.05m, 1000m	near and far clipping planes
T_{US_N}	x_{min}, x_{max}	Left: 0, 640 Right: 640, 1280	screen viewport x & y bounds in pixels[†]
	y_{min}, y_{max}	Left: 512, 1024 Right: 512, 1024	
T_{S_US}	k_n k_{pd} r_{norm} x_{axis}, y_{axis}	0.1933 –0.1 256 (pixels) Left: (395.9, 256) (pixels) Right: (244.1, 256) (pixels)	normalized coefficient of optical distortion normalized coefficient for predistortion normalizing radius in pixels[*] optical axis in screen coordinates (assuming 640 x 512 frame buffer viewport resolution)[†]

Table 8.1 Parameters of Vlib display transformation

Many of the optical parameters listed were measured by Jannick Rolland of UNC, and some are derived in (Rolland & Hopkins, 1993). The value for k_{pd} was derived numerically. The paraxial

[*] The normalizing radius used here is the distance in pixels from the optical axis to the top or bottom of the screen; the choice was arbitrary.

[†] For simplicity, these figures neglect the pixel cropping problem detailed in (Rolland & Hopkins, 1993).

and distorted figures are given for the window parameters since there is no single correct value for systems without distortion correction. Also, small variations in the manufacturing process of HMDs can change the optical parameters substantially, so the numbers given above should be taken as typical values rather than absolutes. Finally, in our non-see-through systems, we have found that a wide range of values for the window parameters will yield an acceptable 3D percept, which suggests that individual, perception-based calibration will still be necessary for certain applications.

It should be clear at this point that making a usable system involves many parameters with complex interactions. The above table is an attempt to list the parameters of most interest to a system designer without geting bogged down in some of the subtler details. More in-depth discussions of optical issues can be found in the papers already cited in this section.

The parameters given in the above table, together with the series of transforms defined in this paper, define the visual display transform for the UNC VR software (Vlib) running on a Silicon Graphics VGX computer. This display transform takes points defined in Object coordinates and transforms them to Screen coordinates. This display algorithm should work for many current VR hardware configurations, provided appropriate values are supplied for the display parameters.

9. Conclusion

We have presented the complete visual display transform for virtual reality, as implemented on the UNC VR system. The considerable number of transforms and coordinate systems in VR requires a systematic method for dealing with them all, and we have presented our method.

The coordinate system graphs were used because they give an intuitive feel for the relationships between coordinate systems. The T_{A_B} notation for transforms was used because it is concise and because it provides a check on correctness by requiring subscripts of adjacent terms to match. The VQS representation was presented as a concise and useful alternative to 4x4 homogeneous matrices for many VR operations. Finally, we have supplied a complete specification of the display transform in the UNC VR system and have also given the numerical parameter values required by the display transform.

We believe that this software architecture for the visual display computation for virtual reality is sufficiently flexible that, with different values for the display parameters, it can handle many different HMDs, trackers, and other hardware devices, and that it can be used in many different applications of virtual reality. This display algorithm is not tied to any particular display hardware, and can be implemented on any computer used to generate graphics imagery for virtual reality.

10. Acknowledgments

We thank the HMD and Pixel-Planes teams at UNC for creating the various components, hardware and software, of our VR system. We thank Fred Brooks for clarifying discussions on nomenclature. An early version of some parts of this display transform was done at NASA Ames Research Center, and we thank Scott Fisher, Jim Humphries, Doug Kerr, and Mike McGreevy for their contributions. We thank Ken Shoemake for help with quaternions. We thank Jannick Rolland for educating us about optics, for help defining the formulas describing optical distortion, and for optical measurements. We thank Anselmo Lastra and Terry Hopkins for discussions about implementation of optical distortion. We thank the Zentrum für Kunst und Medienteknologie in Karlsruhe, Germany for supporting the port of Vlib to the Silicon Graphics VGX. We thank the

Banff Centre for the Arts in Canada, where a draft of this paper was written in a pleasant studio in the woods. Finally, we thank the reviewers for their helpful comments on improving this paper.

This research was supported by the following grants: ARPA #DABT 63-93-C-0048, NSF Cooperative Agreement #ASC-8920219, and ARPA: "Science and Technology Center for Computer Graphics and Scientific Visualization", ONR #N00014-86-K-0680, and NIH #5-R24-RR-02170.

11. References

Blanchard, C., S. Burgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, M. Teitel. (1990). Reality Built for Two: A Virtual Reality Tool. *Proc. 1990 Workshop on Interactive 3D Graphics,* 35-36.

Brooks, F.P., Jr. (1989). Course #29: Implementing and interacting with real-time virtual worlds. *Course Notes: SIGGRAPH '89.*

Buchroeder, R. A., Seeley, G. W., & Vukobradatovich, D. (1981). Design of a Catadioptric VCASS Helmet-Mounted Display. Optical Sciences Center, University of Arizona, under contract to the U.S. Air Force Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Dayton, Ohio, AFAMRL-TR-81-133.

CAE. (1986). Introducing the visual display system that you wear. CAE Electronics, Ltd., C.P. 1800 Saint-Laurent, Quebec, Canada H4L 4X4.

Caudell, T.P. and D.W. Mizell. (1992). Augmented reality: an application of heads-up display technology to manual manufacturing processes. *Proc. Hawaii International Conference on System Sciences.*

Cooke, J.M., M.J. Zyda, D.R. Pratt, and R.B. McGhee. (1992). NPSNET: Flight simulation dynamic modeling using quaternions. *Presence* 1(4).

Craig, John. (1986). *Introduction to robotics.* Addison-Wesley, Reading, Mass.

Fisher, S.S., McGreevy, M., Humphries, J., & Robinett, W. (1986). Virtual Environment Display System. *Proc. 1986 Workshop on Interactive 3D Graphics*, 77-87.

Foley, J., A. van Dam, S. Feiner, J. Hughes. (1990). *Computer Graphics: Principles and Practice* (2nd ed.). Addison-Wesley Publishing Co., Reading MA. 222-226.

Fuchs, H., J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs and L. Israel. (1989). A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics: Proceedings of SIGGRAPH '89.* 23:4:79-88.

Funda, Janez, R H Taylor, R P Paul. 1990. On homogeneous transforms, quaternions, and computational efficiency. IEEE Trans. on robotics and automation. v6n3. June.

Janin, A.L., D.W. Mizell and T.P. Caudell. (1993). Calibration of head-mounted displays for augmented reality applications. *IEEE Virtual Reality Annual International Symposium*, Seattle WA.

Holloway, R.L. (1987). Head-Mounted Display Technical Report. Technical report #TR87-015, Dept. of Computer Science, University of North Carolina at Chapel Hill.

Holloway, R., H. Fuchs, W. Robinett. (1991). Virtual-worlds research at the University of North Carolina at Chapel Hill. *Proc. Computer Graphics '91*. London, England.

Paul, Richard. (1981). *Robot manipulators: Mathematics, programming, and control.* MIT Press, Cambridge, Mass.

Pique, M. (1980). Nested Dynamic Rotations for Computer Graphics. M.S. Thesis, University of North Carolina, Chapel Hill, NC.

Robinett,W., and J.P. Rolland. (1992). A computational model for the stereoscopic optics of a head-mounted display. *Presence*, 1(1). Also UNC Technical Report TR91-009.

Robinett, W., and R. Holloway. (1992). Implementation of flying, scaling, and grabbing in virtual worlds. *ACM Symposium on Interactive 3D Graphics*, Cambridge MA, March.

Rolland, J. P., D. Ariely & W. Gibson. (1993). Towards quantifying depth and size perception in 3D virtual environments. To be published in *Presence*. Also Technical Report #TR93-044, Dept. of Computer Science, University of North Carolina at Chapel Hill.

Rolland, J. P. & T. Hopkins. (1993). A method of computational correction for optical distortion in head-mounted displays. Technical Report #TR93-045, Computer Science Department, University of North Carolina at Chapel Hill. (Available via anonymous ftp from ftp.cs.unc.edu.)

Shoemake, K. (1985). Animating rotations using quaternion curves. *Computer Graphics: Proc. of SIGGRAPH '85*. pp. 245-254.

Silicon Graphics. (1991). *GL Reference Manual.* Silicon Graphics, Inc., Mountain View CA.

Sutherland, I. E. (1968). A head-mounted three-dimensional display. *1968 Fall Joint Computer Conference, AFIPS Conference Proceedings*, 33, 757-764.

VPL. (1989). *VPL EyePhone Operations Manual.* VPL Research, 656 Bair Island Rd., Suite 304, Redwood City, California 94063, p. B-4.

Woodson, W. E. (1981). *Human factors design handbook.* McGraw-Hill.

Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation

Leandra Vicci
Microelectronic Systems Laboratory
Department of Computer Science
University of North Carolina at Chapel Hill

25 September 1998
(updated 9 August 2000)
Technical Report TR01-014

Summary

Think of a quaternion Q as a vector augmented by a real number to make a four element entity. It has a *real* part $Q\rfloor_{re}$ and a *vector* part $Q\rfloor_{ve}$. If $Q\rfloor_{re}$ is zero, Q represents an ordinary vector; if $Q\rfloor_{ve}$ is zero, it represents an ordinary real number. In any case, the ratio between the real part and the *magnitude* of the vector part $|Q\rfloor_{ve}|$ plays an important role in rotations, and is conveniently represented by the parameter $\phi = tan^{-1}(|Q\rfloor_{ve}|/Q\rfloor_{re})$. A unit magnitude quaternion U has a Pythagorean sum of 1 over its four elements, and its product with any vector S_v gives another vector having the same magnitude as S_v but rotated in space. If $S_v \perp U\rfloor_{ve}$, the rotation is by an angle ϕ about the vector $U\rfloor_{ve}$ (or simply about U). If S_v is arbitrary, however, certain cross-terms of the product spoil this convenient relationship. Even in this general case however, these cross-terms cancel in the triple product $R_v = US_vU^{-1}$, where $U^{-1} \equiv 1/U$. The rotations of the two successive products are in the same direction, so R_v represents a rotation of S_v about $U\rfloor_{ve}$ by an angle 2ϕ, which depends only on U. Thus, the operation US_vU^{-1} performs a rotation of S_v which is entirely characterized by the unit quaternion U. The rotation occurs about an axis parallel to U by an amount $2\,tan^{-1}(|U\rfloor_{ve}|/U\rfloor_{re})$. Quaternion notation conveniently handles composition of any number of successive rotations into one equivalent rotation: $U = U_1U_2\cdots U_n$ where each unit quaternion U_i represents one of the succession of rotations. Other operations useful in inertial navigation problems are also presented.

1 Historical background

Quaternions were devised by Sir William Hamilton in his extensions of vector algebras to satisfy the properties of division rings (roughly, quotients exist in the same domain as the operands). In [1], Art.112, Hamilton notes, "...that *for the complete determination,* of what we have called the *geometrical* QUOTIENT *of two Co-initial Vectors,* a *System of Four Elements,* admitting each separately of numerical expression, *is generally required.* ... we have already a *motive* for saying, that 'the Quotient of two Vectors is generally a Quaternion.' "

Quaternions can also be considered to be an extension of classical algebra into the hypercomplex number domain D, satisfying a property that $|p|^2 \cdot |q|^2 = |p \cdot q|^2$ for $(p, q) \in D$ [2]. This domain consists of symbolic expressions of n terms with real coefficients where n may be 1 (real numbers), 2 (complex numbers), 4 (quaternions), 8 (Cayley numbers), but no other possible values (proved by Hurwitz in 1898). Thus, quaternions also share many properties with complex numbers.

While Hamilton provides geometrical interpretations of various proved properties throughout [1], the development itself is fundamentally algebraic, that is, based on the properties of a particular axiomatic set of symbolic operations. The geometric properties of quaternions are nevertheless sweeping, the composition of successive rotations through successive multiplications being just one, albeit an important one.

2 Axiomatic properties of quaternions

Quaternions are defined as sums of 4 terms of the form $Q = 1 \cdot q_1 + i \cdot q_2 + j \cdot q_3 + k \cdot q_4$ where q_1, q_2, q_3, q_4 are reals, 1 is the multiplicative identity element, and i, j, k are symbolic elements having the properties:

$$\begin{gathered} i^2 = -1,\ j^2 = -1,\ k^2 = -1, \\ ij = k,\ ji = -k, \\ jk = i,\ kj = -i, \\ ki = j,\ ik = -j. \end{gathered}$$

Customarily, the extension of an algebra should attempt to preserve the properties of the operators defined in the original algebra. Generalizing from the classical algebra of real and complex numbers to quaternions motivates the following operator rules.

2.1 Addition of quaternions

The addition rule for quaternions is component-wise addition:

$$P+Q = (p_1+ip_2+jp_3+kp_4)+(q_1+iq_2+jq_3+kq_4) = (p_1+q_1)+i(p_2+q_2)+j(p_3+q_3)+k(p_4+q_4).$$

This rule preserves the associativity and commutativity properties of addition, and provides a consistent behavior for the subset of quaternions corresponding to real numbers, i.e.,

$$P_r + Q_r = (p + 0i + 0j + 0k) + (q + 0i + 0j + 0k) = p + q.$$

2.2 Multiplication of quaternions

The multiplication rule for quaternions is the same as for polynomials, extended by the multiplicative properties of the elements i, j, k given above. Written out for close inspection, we have:

$$\begin{aligned} PQ &= (p_1 + ip_2 + jp_3 + kp_4)(q_1 + iq_2 + jq_3 + kq_4) \\ &= (p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4) + i(p_1q_2 + p_2q_1 + p_3q_4 - p_4q_3) \\ &\quad + j(p_1q_3 + p_3q_1 + p_4q_2 - p_2q_4) + k(p_1q_4 + p_4q_1 + p_2q_3 - p_3q_2). \end{aligned}$$

A term-wise inspection reveals that commutativity is not preserved. Associativity and distributivity over addition are preserved, however, the proof being left to the reader. And as desired for the subset of reals, $P_rQ_r = pq$.

2.3 Conjugates of quaternions

Consistent with complex numbers, let us define the *conjugate* operation on a given quaternion Q to be,

$$\overline{Q} = \overline{(q_1 + iq_2 + jq_3 + kq_4)} \equiv (q_1 - iq_2 - jq_3 - kq_4).$$

As with complex numbers, note that both $(Q + \overline{Q})$ and $(Q\overline{Q})$ are real. Moreover, if we define the absolute value or *norm* of Q to be,

$$|Q| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2},$$

then apparently $Q\overline{Q} = \overline{Q}Q = |Q|^2$. The conjugate operation is distributive over addition, that is, $\overline{P + Q} = \overline{P} + \overline{Q}$. With respect to multiplication however, $\overline{PQ} = \overline{Q}\ \overline{P}$, the proof of which is left as an exercise to the reader.

3 Other properties of quaternions

The axioms in the previous section completely *define* quaternions in terms of the desired properties under three basic operations. Many other properties may be proved.

3.1 General properties

Mathematically, the most important property is that the quaternions form a division ring (i.e., quaternion quotients exist).

3.1.1 Division of quaternions

Since multiplication is not commutative, let us derive both a *left quotient* Q_L^{-1} and a *right quotient* Q_R^{-1} by defining the symbolic expression P/Q to be solutions of the following two identities,

$$QQ_L^{-1} = P, \qquad Q_R^{-1}Q = P.$$

Multiplying both sides of these identities respectively on the left and right by $\overline{Q}/|Q|^2$ we have immediately,

$$Q_L^{-1} = \frac{\overline{Q}P}{|Q|^2}, \qquad Q_R^{-1} = \frac{P\overline{Q}}{|Q|^2}.$$

Thus in general two distinct quotients will occur, however in the special case where $P = 1$, we have by definition the multiplicative *inverse* of a quaternion,

$$Q_L^{-1} = Q_R^{-1} = Q^{-1} = \frac{\overline{Q}}{|Q|^2}$$

3.1.2 Quaternion multiplication is distributive over addition

A term-wise expansion of $P(Q+S) = PQ + PS$ proves this property and is left as an exercise for the reader.

3.1.3 Unit quaternions

The subspace U of *unit quaternions* which satisfy the condition $|U| = 1$ have some important properties. A trivially apparent one is,

$$U^{-1} = \overline{U}.$$

A less obvious, but very useful one is,

$$U = U_r cos\,\phi + U_v sin\,\phi = cos\,\phi + U_v sin\,\phi,$$

where $U_r = (1,0,0,0)$ is a real unit quaternion, $U_v = (0, iu_2, ju_3, ku_4)$ is a vector unit quaternion parallel to the vector part of U, and ϕ is a real number. The proof is straightforward:

$$\begin{aligned} |U|^2 &= U\overline{U} = (U_r cos\,\phi + U_v sin\,\phi)\overline{(U_r cos\,\phi + U_v sin\,\phi)} \\ &= U_r\overline{U}_r cos^2\phi + (U_r\overline{U}_v + U_v\overline{U}_r) sin\,\phi\, cos\,\phi + U_v\overline{U}_v sin^2\phi \\ &= cos^2\phi + sin^2\phi = 1. \end{aligned}$$

At this time, let's interpret ϕ as simply quantifying the ratio of the real part to the magnitude of the vector part of a quaternion. Its geometrical representation as specifying an angle of rotation will be presented later.

3.2 Vector properties of quaternions

The quaternion $Q = (q_1 + iq_2 + jq_3 + kq_4)$ can be interpreted as having a real part q_1, and a vector part $(iq_2 + jq_3 + kq_4)$, where the elements $\{i, j, k\}$ are given an added *geometric* interpretation as unit vectors along the x, y, z axes, respectively. Accordingly, the subspace $Q_r = (q_1 + 0i + 0j + 0k)$ of *real quaternions* may be regarded as being equivalent to the real numbers, $Q_r = q$. Similarly, the subspace $Q_v = (0 + iq_2 + jq_3 + kq_4)$ of *vector quaternions* may be regarded as being equivalent to the ordinary vectors, $Q_v = \boldsymbol{q} \equiv (iq_x + jq_y + kq_z)$.

3.2.1 Products of real quaternions

The product of real quaternions is real, and the operation is commutative:

$$P_rQ_r = pq = qp = Q_rP_r.$$

Moreover, the operation is associative:

$$(P_rQ_r)S_r = (pq)s = p(qs) = P_r(Q_rS_r).$$

3.2.2 Product of a real quaternion with a vector quaternion

The product of a real and a vector quaternion is a vector, and the operation is commutative:

$$P_rQ_v = (0 + p_1q_2i + p_1q_3j + p_1q_4k) = (0 + q_2p_1i + q_3p_1j + q_4p_1k) = Q_vP_r.$$

3.2.3 Products of vector quaternions

The product of two vector quaternions has the remarkable property,

$$\begin{aligned} P_vQ_v &= -(p_2q_2 + p_3q_3 + p_4q_4) + (p_3q_4 - p_4q_3)i + (p_4q_2 - p_2q_4)j + (p_2q_3 - p_3q_2)k \\ &= -\boldsymbol{p}\cdot\boldsymbol{q} + \boldsymbol{p}\times\boldsymbol{q}, \end{aligned}$$

where the "$\cdot$" and "$\times$" operators are respectively the "dot" and "cross" products of classical vector algebra. This is clearly a general quaternion except in two special cases: if $P_v \parallel Q_v$ the product is a real quaternion equal to $-\boldsymbol{p}\cdot\boldsymbol{q}$ and if $P_v \perp Q_v$ the product is a vector quaternion equal to $\boldsymbol{p}\times\boldsymbol{q}$.

3.2.4 Parallel and perpendicular quaternions

We call quaternions P and Q *parallel* ($P \parallel Q$) if their *vector parts* $P\rfloor_{ve} = (P - \overline{P})/2$ and $Q\rfloor_{ve} = (Q - \overline{Q})/2$ are parallel; i.e., if $(S - \overline{S}\,) = 0$, where $S = P\rfloor_{ve}Q\rfloor_{ve}$. Similarly, we call them *perpendicular* ($P \perp Q$) if $P\rfloor_{ve}$ and $Q\rfloor_{ve}$ are perpendicular; i.e. if $(S + \overline{S}\,) = 0$.

3.2.5 Product of a unit quaternion and a perpendicular vector quaternion

Properties of this curiously specialized case are useful in understanding how quaternions can be used to rotate vectors in 3-space. Let S_v be a vector quaternion, U be a unit quaternion, and $S_v \perp U$. Then according to section 3.1.3, we can write,

$$T = US_v = (cos\,\phi + sin\,\phi\, U_v)S_v = cos\,\phi\, S_v + sin\,\phi\, U_vS_v,$$

where $U_v \parallel U$. The first term is a vector $T_{v(1)} \parallel S_v$. Since $S_v \perp U_v$, the second term must also be a vector $T_{v(2)}$; moreover $T_{v(2)} \perp S_v$ and $T_{v(2)} \perp U \parallel U_v$. Since the product T is a sum of vectors it must also be a vector, i.e., $T = T_v$. Both $T_{v(1)}$ and $T_{v(2)}$ lie in a plane perpendicular to U. Thus $T_v = T_{v(1)} + T_{v(2)}$ can be geometrically interpreted as a rotation of S_v by an angle ϕ in this plane, i.e., about an axis parallel to U.

Now consider the product,

$$R_v = T_vU^{-1} = T_v\overline{U} = cos\,\phi\, T_v + sin\,\phi\, T_v\overline{U}_v = cos\,\phi\, T_v - sin\,\phi\, T_vU_v.$$

The vector identity $T_vU_v = -U_vT_v$ can be used to rewrite this as,

$$R_v = cos\,\phi\, T_v + sin\,\phi\, U_vT_v,$$

which is another rotation of angle ϕ about U. The rotation ϕ is in the same sense for these two products, so the operation

$$R_v = US_vU^{-1}$$

performs a rotation of S_v about U by an angle 2ϕ.

3.3 General rotations in 3-space; Reference frames

In section 3.2.5 we saw how the operation US_vU^{-1} rotated a *perpendicular* vector S_v about a unit quaternion U. Now let's consider how this operation behaves with an *arbitrary* vector V_v. We can decompose $V_v = W_v + S_v$ where $W_v \parallel U$ and $S_v \perp U$. Then,

$$UV_vU^{-1} = U(W_v + S_v)U^{-1} = UW_vU^{-1} + US_vU^{-1} = UW_vU^{-1} + R_v,$$

where R_v is S_v rotated about U by an angle 2ϕ. To evaluate the first term, note that since $W_v \parallel U$ we can write $W_v = zU_v$, where z is a real number and unit vector $U_v \parallel U$. Thus,

$$UW_vU^{-1} = UzU_vU^{-1} = zUU_vU^{-1} = zU_vUU^{-1} = zU_v = W_v.$$

That $UU_v = U_vU$ is left as an exercise to the reader. Finally then, we have:

$$UV_vU^{-1} = W_v + R_v.$$

Geometrically, we interpret this as a rotation of V_v about U by an angle of 2ϕ.

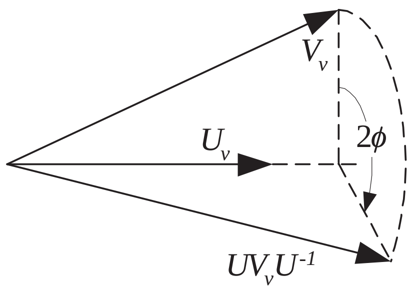


Figure 1:
Arbitrary vector V_v is rotated by unit quaternion U about a unit vector $U_v \parallel U$, through angle 2ϕ.

This operation performs the same rotation on *all* vectors including the unit vectors of a coordinate system. Therefore, it can be used to rigidly transform the coordinates of any *reference frame* into a new frame of different orientation. This is a very useful property.

3.4 Composition of successive rotations

Let Q_1 and Q_2 be two unit quaternions representing arbitrary rotations in 3-space as described in section 3.3. Applying them in succession to a vector V_v,

$$Q_2(Q_1 V_v Q_1^{-1})Q_2^{-1} = (Q_2 Q_1)V_v(Q_1^{-1}Q_2^{-1}) = (Q_2 Q_1)V_v(Q_2 Q_1)^{-1} = Q_i V_v Q_i^{-1},$$

where the unit quaternion $Q_i = Q_2 Q_1$ is the successive composition of two rotations. This property generalizes to the composition of any number of rotations. In this reverse order composition, each successive rotation is relative to the *initial* reference frame as is illustrated in Figure 2a.

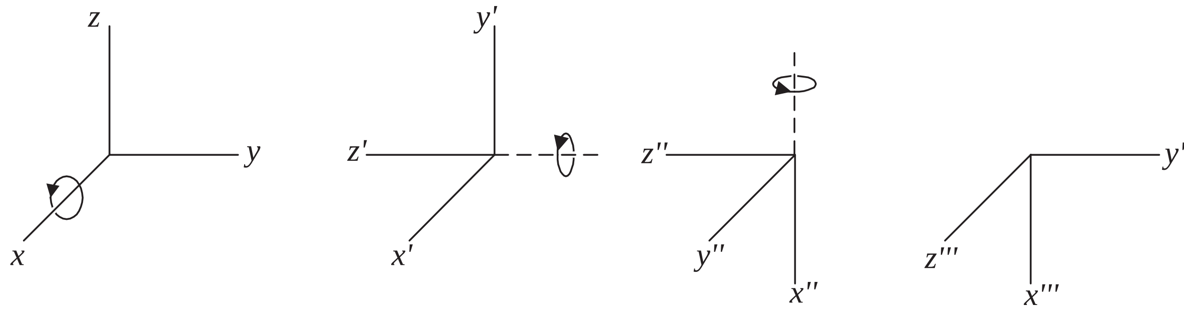


Figure 2a: 90° rotations of a reference frame about the initial x, y, z axes, respectively

Composing a rotation in the forward order, $Q_c = Q_1 Q_2 \ldots$, has the effect of performing each successive rotation relative to its *current* reference frame, illustrated in Figure 2b.

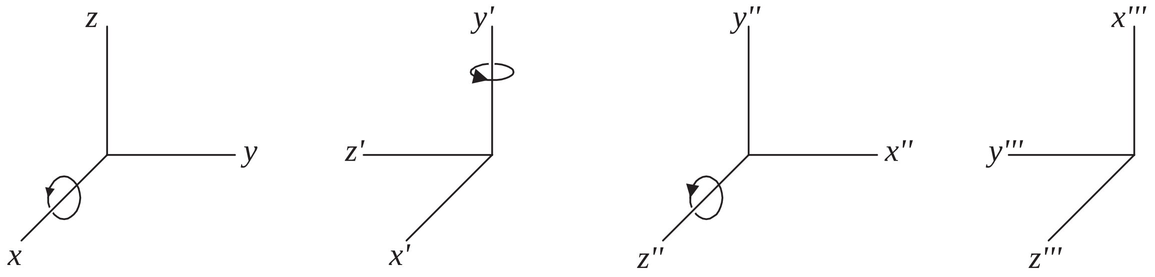


Figure 2b: 90° rotations of a reference frame about its current x, y, z axes, respectively.

4 Strapdown inertial navigation system (INS) applications

Usage of quaternions by this branch of engineering is common, but the notation often differs in some respects from the above, and a more detailed annotation is provided to relate variables to reference frames. Specifically in this section, I'll follow the notation used in Titterton and Weston [3]. I will introduce this notation, then derive expressions for some of the commonly used operations for INS engineering.

4.1 Frames and coordinates

It is often convenient to represent the same physical situation in a number of different frames of reference which may differ by displacement, rotation, and system of coordinates. Each frame comprises a complete definition of these parameters. A privileged, *inertial* family of frames are those in which physical objects experience no inertial forces.

Cartesian coordinate systems, while not necessary, are generally used as coordinate systems of the frames discussed in [3]. The non-scalar data types used are vectors, matrices, and quaternions. Distinct from the data types, are the kinds of variables treated, i.e., positions, linear velocities, and angular rates.

4.2 Superscripts and subscripts

Superscripts and subscripts are used to associate certain attributes of a variable with coordinate frames. On a gross level, the notation is consistent, but there are fine nuances, depending on the kind of the variable but not its type.

Superscripts are used consistently for all kinds of variables. S^i indicates that the variable S is expressed in the coordinates of the i^{th} frame.

4.2.1 The position variable X_j^i

X_j^i represents the position of a point relative to the origin of the j^{th} frame, expressed in the coordinates of the i^{th} frame. In most cases $i = j$, and it is common to use implicit notations. X_j and X^j both represent X_j^j, where the choice of super- or subscript depends on what is being emphasized.

4.2.2 The velocity variable V_j^i

The variable V_j^i represents a velocity taken relative to the j^{th} frame, expressed in coordinates of the i^{th} frame. The velocity in any frame is not dependent on the location of the origin of the frame; rather it may be taken relative to the velocity of *any fixed point* in that frame. Just as for position variables, $V_j = V_j^j$ is implied.

4.2.3 The angular rate variable Ω_{jk}^i

The variable Ω_{jk}^i represents an angular rate of rotation of the k^{th} entity relative to the j^{th} frame, expressed in coordinates of the i^{th} frame. Just as for velocities, the location of origin of reference frame j is not relevant; rather the angular rate is taken relative to the angular rate of any fixed point in the j^{th} frame. Often, the k^{th} entity is another frame, so this notation conveniently expresses the angular rate of rotation of the k^{th} frame relative to the j^{th}.

4.3 A pure vector representation of a rotation

It is also possible to completely represent a 3D rotation with a pure vector. The geometric properties of algebraic operations on this representation are naturally quite different than for unit quaternions. For some purposes these properties are particularly useful.

Let vector $\boldsymbol{a} = a\hat{\boldsymbol{a}}$ represent a rotation where its unit vector $\hat{\boldsymbol{a}}$ specifies the axis of rotation and its magnitude a specifies the angular amount of rotation. From this we can uniquely construct a unit quaternion, $A = cos(a/2) + sin(a/2)\hat{\boldsymbol{a}}$, such that $AS_v\overline{A}$ performs a rotation of S_v about $\hat{\boldsymbol{a}}$ by an angle equal to a.

Let us define a transform $\mathcal{Q}$ of the vector representation $\boldsymbol{a}$ to the unit quaternion representation A of a 3D rotation:

$$A = \mathcal{Q}(\boldsymbol{a}) = \mathcal{Q}(a\hat{\boldsymbol{a}}) = cos(a/2) + sin(a/2)\hat{\boldsymbol{a}}.$$

Likewise, let us define the inverse transform,

$$\boldsymbol{a} = \mathcal{Q}^{-1}(A) = \mathcal{Q}^{-1}(A_r + A_v\hat{\boldsymbol{a}}) = 2tan^{-1}(A_v/A_r)\hat{\boldsymbol{a}}.$$

4.4 Time derivative of a rotation quaternion

Assume a b-frame that is rotating with respect to a reference n-frame. At any instant, let the unit quaternion U represent a rotation of an arbitrary constant vector C^b in the b-frame into a vector $C^n = UC^b\overline{U}$ in the n-frame. Since this rotation progresses continuously in time, $U = U(t)$ has a time derivative $\dot{U}$ which we now derive.

Applying the derivative of products rule to C^n, we have, (since $\dot{C}^b = 0$),

$$\dot{C}^n = \dot{U}\,C^b\,\overline{U} + U\,C^b\,\dot{\overline{U}} = \dot{U}\,C^b\,\overline{U} + \overline{\dot{U}\,\overline{C^b}\,\overline{U}} = \dot{U}\,C^b\,\overline{U} - \overline{\dot{U}\,C^b\,\overline{U}}.$$

In the vector formulation of classical mechanics [4], a vector $\boldsymbol{p}$ is used to represent an instantaneous rate of rotation, $\dot{\boldsymbol{c}} = \boldsymbol{p} \times \boldsymbol{c}$, where $\boldsymbol{c}$ is an arbitrary vector, and $\dot{\boldsymbol{c}}$ is its variation with time. In the n-frame, a quaternion formulation of this equation is,

$$\dot{C}^n = (P^nC^n - \overline{P^nC^n})/2.$$

Since $\boldsymbol{c}$ is arbitrary, this equation can be applied to an entire coordinate system, and we can represent the rate of rotation of the b-frame in the n-frame as $P^n = P^n_{nb}$.

Equating the expressions for $\dot{C}^n$, we have, $\dot{U}C^b\overline{U} = P^n_{nb}C^n/2 = P^n_{nb}(UC^b\overline{U})/2$, or $\dot{U} = P^n_{nb}U/2$. It is often the case that the rotational rate is measured in the rotating b-frame, so we can substitute the identity $P^n_{nb} = UP^b_{nb}\overline{U}$, to obtain

$$\dot{U} = UP^b_{nb}/2.$$

4.5 Interpolation between rotations

Given two arbitrary rotations U_{10}, U_{20} from the 0-frame to the 1 and 2-frames respectively, geometric intuition would suggest an interpolation between them would be along the single rotation U_{21} taking the 1-frame into the 2-frame. In fact, this can be visualized as a great circle on a unit 4-sphere which connects the images of U_{10} and U_{20}. This great circle lies in a plane normal to $U_{21}\rfloor_{ve}$. The locus of points lying between U_{10} and U_{20} on the great circle corresponds to a rotational angle of between 0 and $cos^{-1}(U_{21}\rfloor_{re})$.

Now $U_{20} = U_{10}U_{21} \Rightarrow U_{21} = \overline{U_{10}}U_{20}$. Let $U_{21} = cos(\phi_{21}) + \hat{\boldsymbol{u}}_{21}sin(\phi_{21})$, whence we can calculate $\phi_{21} = cos^{-1}(U_{21}\rfloor_{re})$ and $\hat{\boldsymbol{u}}_{21} = U_{21}\rfloor_{ve}/sin(\phi_{21})$.

Given $\phi_{x1} \ni (0 \leq \phi_{x1} \leq \phi_{21})$ we construct $U_{x1} = cos(\phi_{x1}) + \hat{\boldsymbol{u}}_{21}sin(\phi_{x1})$, from which we calculate the interpolated rotation,

$$U_{x0} = U_{10}U_{x1}.$$

APPENDIX A – Summary of formal properties

A.1 Notation

r	a scalar (real) number
$\boldsymbol{v}$	a vector
$\hat{\boldsymbol{u}}$	a unit vector, $\boldsymbol{u} \cdot \boldsymbol{u} = 1$
i, j, k	symbolic constants with special properties (section 2)
Q	a quaternion $[q_1, q_2, q_3, q_4] = q_1 + iq_2 + jq_3 + kq_4$
$\overline{Q}$	the conjugate $[q_1, -q_2, -q_3, -q_4]$ of quaternion Q
$\|Q\|$	the norm, or magnitude $\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$ of quaternion Q
Q^{-1}	the reciprocal $Q/(Q\overline{Q})$, or multiplicative inverse of quaternion Q
Q_r	a (purely) real quaternion $[q_1, 0, 0, 0]$
Q_v	a (purely) vector quaternion $[0, q_2, q_3, q_4]$
U	a unit quaternion, $\|Q\| = 1$
$Q\rfloor_{re}$	the real part $q = q_1$ of quaternion Q
$Q\rfloor_{ve}$	the vector part $\boldsymbol{q} = [q_2, q_3, q_4]$ of quaternion Q
$Q\|\|P$	(the vector parts of) P and Q are parallel
$Q \perp P$	(the vector parts of) P and Q are perpendicular

A.2 Properties

$P + (Q + S) = (P + Q) + S$	addition is associative
$P + Q = Q + P$	addition is commutative
$P(QS) = (PQ)S$	multiplication is associative
$PQ \neq QP$	multiplication is not commutative
$pQ = Qp$	scalar multiplication is commutative
$P(Q + S) = PQ + PS$	left multiplication is distributive over addition
$(P + Q)S = PS + QS$	right multiplication is distributive over addition
$\|Q\| = \sqrt{Q\overline{Q}} = \sqrt{\overline{Q}Q}$	the norm of Q
$Q\rfloor_{re} = (Q + \overline{Q})/2$	the real part of Q
$Q\rfloor_{ve} = (Q - \overline{Q})/2$	the vector part of Q
$Q^{-1} = \overline{Q}/\|Q\|^2$	the reciprocal of Q
$U^{-1} = \overline{U}$	the reciprocal of unit U
$Q^{-1}P = \overline{Q}P/\|Q\|^2$	the left quotient
$PQ^{-1} = P\overline{Q}/\|Q\|^2$	the right quotient
$\overline{PQ} = \overline{Q}\,\overline{P}$	conjugate of a product
$P_vQ_v = -\boldsymbol{p} \cdot \boldsymbol{q} + \boldsymbol{p} \times \boldsymbol{q}$	product of vector quaternions

References

[1] Sir William Rowan Hamilton, “Elements of Quaternions,” Third Edition, Chelsea Publishing Co., New York, 1963.

[2] I. L. Kantor and A. S. Solodovnikov, “Hypercomplex Numbers,” English translation, Springer-Verlag, New York, 1989.

[3] D. H. Titterton and J. L. Weston, “Strapdown inertial navigation technology,” Peter Peregrinus, Ltd., IEE, Stevenage, UK, 1997.

[4] Herbert Goldstein, “Classical Mechanics,” Addison-Wesley, Reading MA, 1950, pp. 132–134.

[5] J. P. Ward, “Quaternions and Cayley Numbers,” Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

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.)

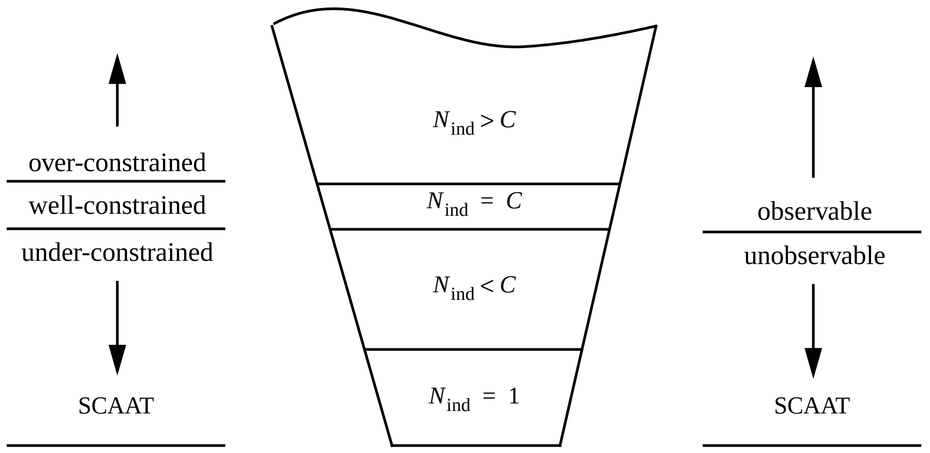


Figure 1: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{\text{ind}}$ and $N_{\text{ind}} \geq C$, 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 $N = N_{\text{ind}} = 1$ 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

$$\begin{aligned} t_e &= Nt_m + t_c \ , \\ r_e &= \frac{1}{t_e} = \frac{1}{Nt_m + t_c} \ . \end{aligned} \tag{1}$$

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.

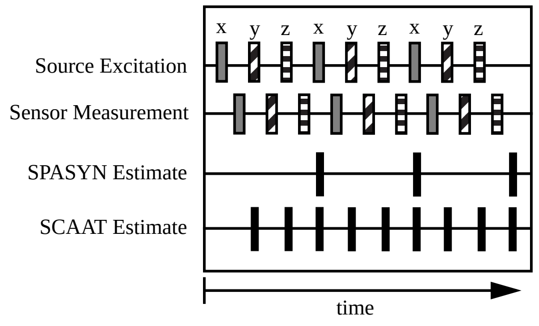


Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

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:

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

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

$$\vec{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{3}$$

and the four-element external orientation quaternion

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

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 $1 \leq i \leq m_\sigma$ and $1 \leq j \leq n$. 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).

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

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

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

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 $\vec{h}_\sigma(\bullet)$ 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 $100\mu s$ 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

$$\vec{z}_t = \left[\vec{z}^{\,T}_{\sigma_1, t_1} \ \dots \ \vec{z}^{\,T}_{\sigma_N, t_N}\right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \dots t_N$, 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 $\vec{z}_{\sigma, t}$. 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 $\vec{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\vec{b}_t$ and $\vec{c}_t$, 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.

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

3.2.1 Device Filters

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

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 $\tilde{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\tilde{b}_t$ and $\tilde{c}_t$ ($\hat{x}_{b,t}$ 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), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3+9+9) = 504{,}000$ 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 $\alpha 1$ 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

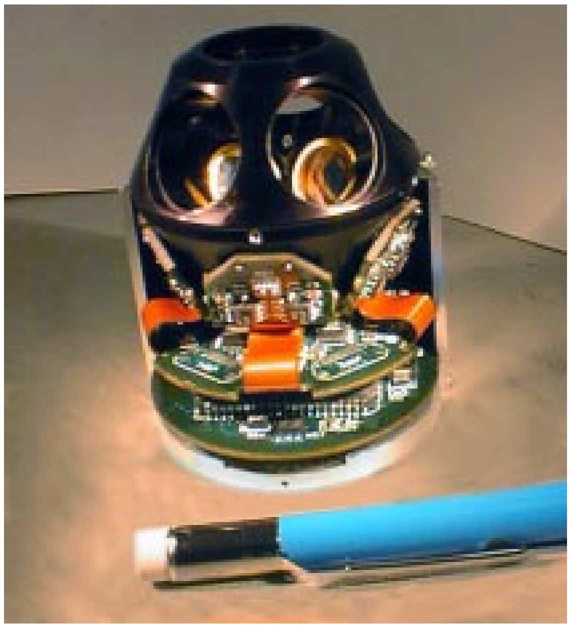


Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size 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 $\vec{z}_\sigma = [u, v]^T$. The measurement function $\vec{h}_\sigma(\bullet)$ 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 $\vec{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\vec{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\vec{b}_t$ (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 $\vec{\eta}$ 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 $\widehat{x}^-$) to project a distance-dependent electrical variance onto the camera.

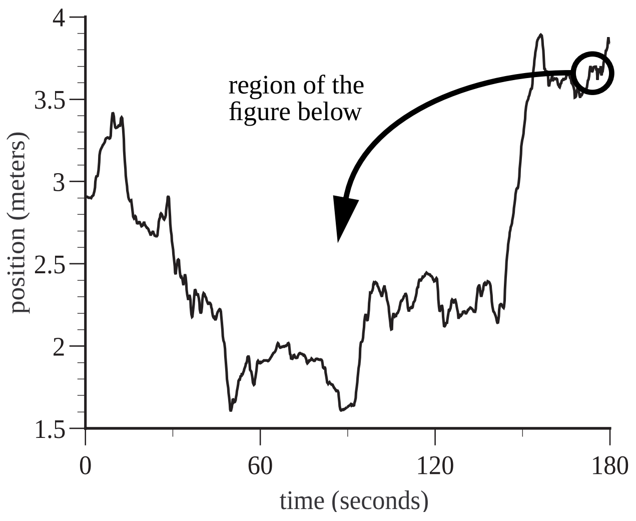
4.2 Initialization

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.)



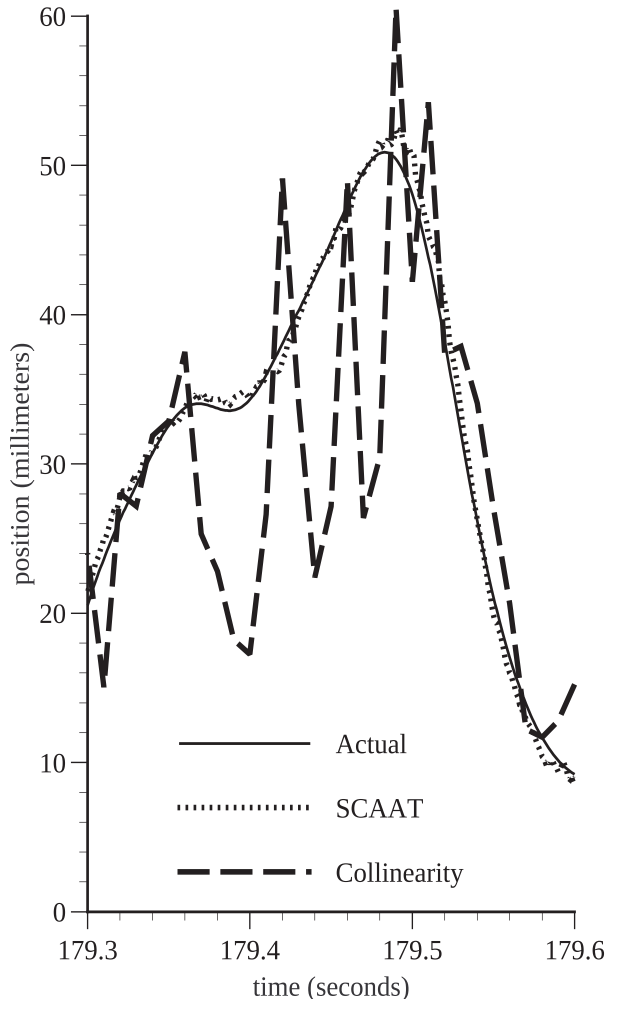


Figure 4: The upper plot depicts the entire 3 minutes of *x*-axis position data from user motion data set ‘a’ of sets ‘a’-’f’. The lower plot shows a close-up of a short portion of the simulation. Collinearity here used $N = 10$ beacons per observation, hence its lower estimate rate. On the other hand,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

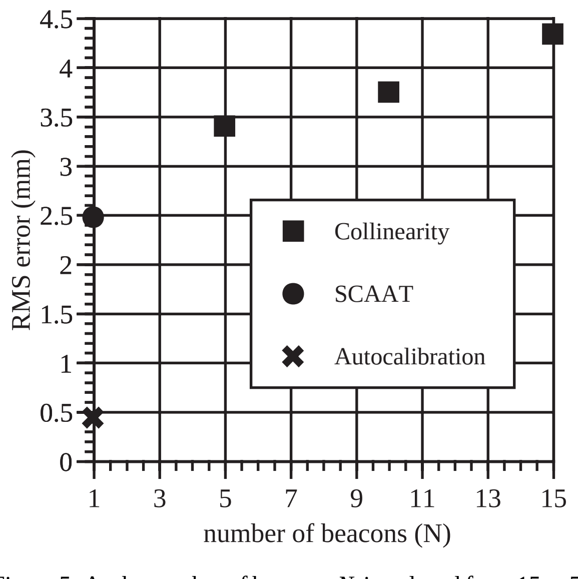


Figure 5: As the number of beacons N is reduced from 15 to 5, the Collinearity results improve slightly. (The Collinearity algorithm generally becomes unstable with $N \leq 4$.) The SCAAT results, with $N = 1$ beacons, are better, and especially good once autocalibration is turned on.

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.

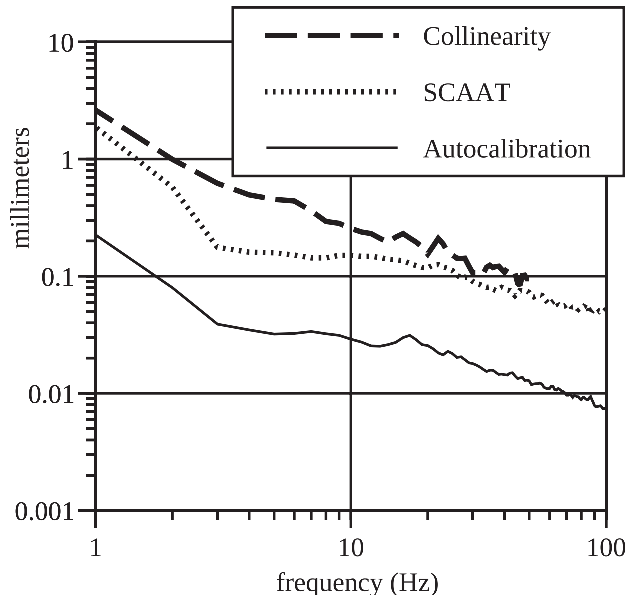


Figure 6: Here we show an error spectra comparison for the Collinearity method with $N = 10$ beacons, and the SCAAT method with and without autocalibration.

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

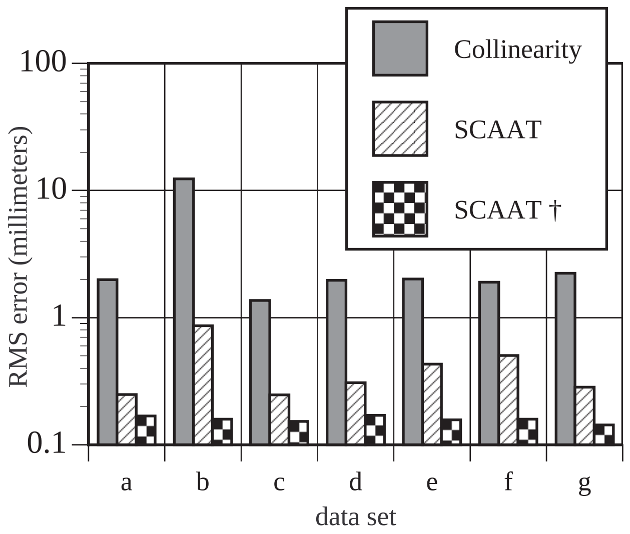


Figure 7: RMS error results for simulations of all seven real user motion data sets. The † symbol indicates a second pass through the motion data set, this time using the already autocalibrated beacons.

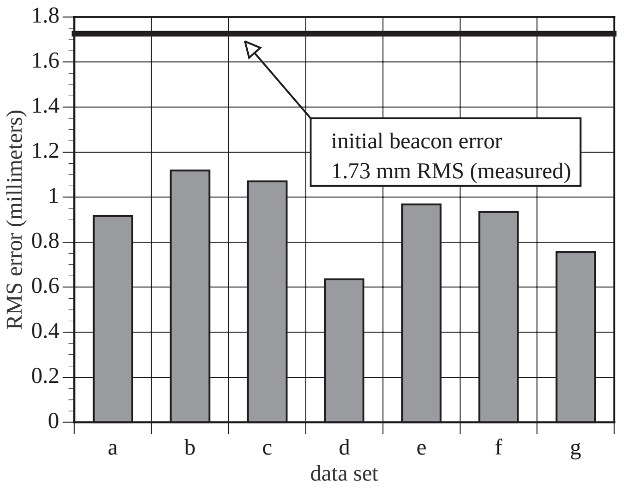


Figure 8: Autocalibration in action. Here we show the final beacon position error for runs through each of the seven user motion data sets.

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).

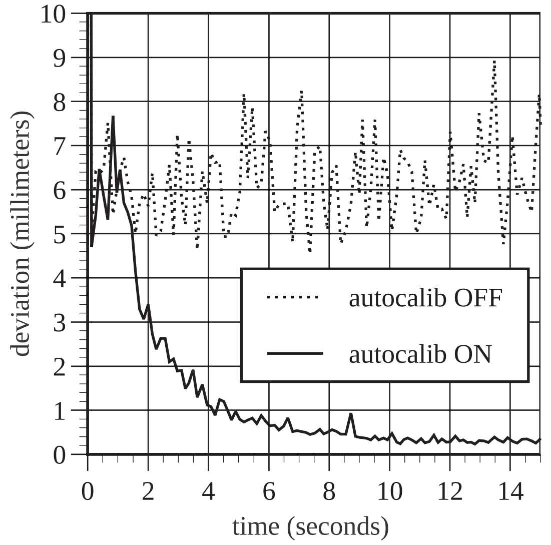


Figure 9: SCAAT position (only) estimate deviation for a Hiball sitting still on a flat surface, with and without autocalibration.

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.

High-Performance Wide-Area Optical Tracking
The HiBall Tracking System

Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, and Kurtis Keller:

University of North Carolina at Chapel Hill
Department of Computer Science, CB# 3175
Chapel Hill, NC 27599-3175 USA
01-919-962-1700
{welch, gb, vicci, brumback, keller}@cs.unc.edu

D'nardo Colucci:

Alternate Realities Corporation
27 Maple Place
Minneapolis, MN 55401 USA
01-612-616-9721
colucci@virtual-reality.com

High-Performance Wide-Area Optical Tracking
The HiBall Tracking System

ABSTRACT

Since the early 1980's the Tracker Project at the University of North Carolina at Chapel Hill has been working on wide-area head tracking for Virtual and Augmented Environments. Our long-term goal has been to achieve the high performance required for accurate visual simulation throughout our entire laboratory, beyond into the hallways, and eventually even outdoors.

In this article we present results and a complete description of our most recent electro-optical system, the *HiBall Tracking System*. In particular we discuss motivation for the geometric configuration, and describe the novel optical, mechanical, electronic, and algorithmic aspects that enable unprecedented speed, resolution, accuracy, robustness, and flexibility.

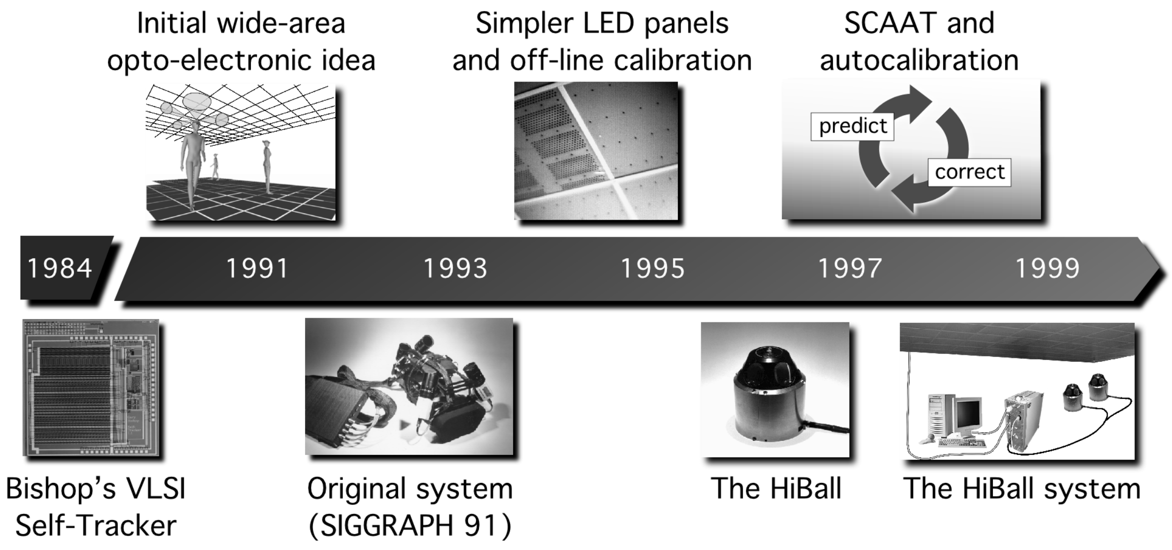

Initial wide-area opto-electronic idea
Simpler LED panels and off-line calibration
SCAAT and autocalibration
predict
correct
1984
1991
1993
1995
1997
1999
Bishop's VLSI Self-Tracker
Original system (SIGGRAPH 91)
The HiBall
The HiBall system

Figure 1

1. INTRODUCTION

Systems for *head tracking* for interactive computer graphics have been explored for over 30 years (Sutherland, 1968). As illustrated in Figure 1, the authors have been working on the problem for over twenty years (Azuma, 1993, 1995; Azuma & Bishop, 1994a, 1994b; Azuma & Ward, 1991; Bishop, 1984; Gottschalk & Hughes, 1993; UNC Tracker Project, 2000; Wang, 1990; J.-F. Wang et al., 1990; Ward, Azuma, Bennett, Gottschalk, & Fuchs, 1992; Welch, 1995, 1996; Welch & Bishop, 1997; Welch et al., 1999). From the beginning our efforts have been targeted at *wide-area* applications in particular. This focus was originally motivated by applications for which we believed that actually walking around the environment would be superior to virtually "flying." For example, we wanted to interact with room-filling virtual molecular models, and to naturally explore life-sized virtual architectural models. Today we believe that a wide-area system with high performance everywhere in our laboratory provides increased flexibility for all of our graphics, vision, and interaction research.

1.1 Previous Work

In the early 1960's Ivan Sutherland implemented both mechanical and ultrasonic (carrier phase) head tracking systems as part of his pioneering work in virtual environments. He describes these systems in his seminal paper "A Head-Mounted Three Dimensional Display" (Sutherland, 1968). In the ensuing years, commercial and research teams have explored mechanical, magnetic, acoustic, inertial, and optical technologies. Complete surveys include (Bhatnagar, 1993; Burdea & Coiffet, 1994; Meyer, Applewhite, & Biocca, 1992; Mulder, 1994a, 1994b, 1998). Commercial magnetic tracking systems for example (Ascension, 2000; Polhemus, 2000) have enjoyed popularity as a result of a small user-worn component and relative ease of use. Recently inertial hybrid systems (Foxlin, Harrington, & Pfeifer, 1998; Intersense, 2000) have been gaining popularity for similar reasons, with the added benefit of reduced high-frequency noise and direct measurements of derivatives.

An early example of an optical system for tracking or motion capture is the *Twinkle Box* system by Burton (Burton, 1973; Burton & Sutherland, 1974). This system measured the positions of user-worn flashing lights with optical sensors mounted in the environment behind rotating slotted disks. The *Selspot* system (Woltring, 1974) used fixed camera-like photo-diode sensors and target-mounted infrared light-emitting diodes that could be tracked in a one-meter-square volume. Beyond the HiBall Tracking System, examples of current optical tracking and motion capture systems include the *FlashPoint*© and *Pixsys*™ systems by Image Guided Technologies (IGT, 2000), the *laserBIRD*™ system by Ascension Technology (Ascension, 2000), and the *CODA Motion Capture System* by B & L Engineering (BL, 2000). These systems employ analog optical sensor systems to achieve relatively high sample rates for a moderate number of targets. Digital cameras (two-dimensional image-forming optical devices) are used in motion capture systems such as the *HiRes 3D Motion Capture System* by the Motion Analysis Corporation (Kadaba & Stine, 2000; MAC, 2000) to track a relatively large number of targets, albeit at a relatively low rate because of the need for 2D image processing.

1.2 Previous Work at UNC-Chapel Hill

As part of his 1984 dissertation on *Self-Tracker*, Bishop put forward the idea of outward looking tracking systems based on user-mounted sensors that estimate user *pose*[1] by observing landmarks in the environment (Bishop, 1984). He described two kinds of landmarks: high signal-to-noise-ratio beacons such as LEDs (light emitting diodes) and low signal-to-noise-ratio landmarks such as naturally occurring features. Bishop designed and demonstrated custom VLSI chips (Figure 2) that combined image sensing and processing on a single chip (Bishop & Fuchs, 1984). The idea was to combine multiple of these chips into an outward-looking cluster that estimated cluster motion by observing natural features in the un-modified environment. Integrating the resulting motion to estimate pose is prone to accumulating error, so further development required a complementary system based on easily detectable landmarks (LEDs) at known locations. This LED-based system was the subject of a 1990 dissertation by Jih-Fang Wang (Wang, 1990).

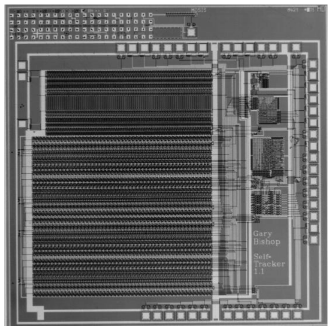


Figure 2

[1] We use the word *pose* to indicate both position and orientation (six degrees of freedom).

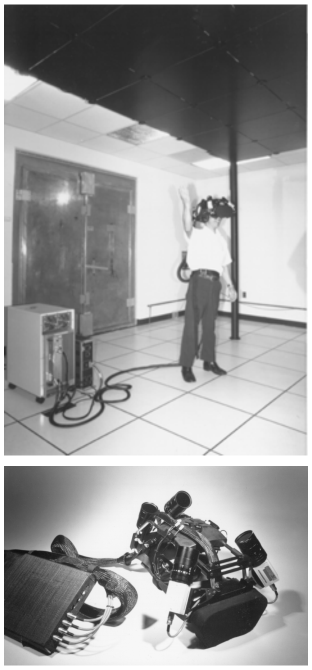


Figure 3

In 1991 we demonstrated a working scalable electro-optical head-tracking system in the *Tomorrow's Realities* gallery at that year's ACM SIGGRAPH conference (J.-F. Wang et al., 1990; Wang, Chi, & Fuchs, 1990; Ward et al., 1992). The system (Figure 3) used four head-worn lateral effect photo-diodes that looked upward at a regular array of infrared LEDs installed in precisely machined ceiling panels. A user-worn backpack contained electronics that digitized and communicated the photo-coordinates of the sighted LEDs. Photogrammetric techniques were used to compute a user's head pose using the known LED positions and the corresponding measured photo-coordinates from each LEPD sensor (Azuma & Ward, 1991). The system was ground-breaking in that it was unaffected by ferromagnetic and conductive materials in the environment, and the working volume of the system was determined solely by the number of ceiling panels. (See Figure 3, top.)

1.3 The HiBall Tracking System

In this article we describe a new and vastly improved version of the 1991 system. We call the new system the *HiBall Tracking System*. Thanks to significant improvements in hardware and software this HiBall system offers unprecedented speed, resolution, accuracy, robustness, and flexibility. The bulky and heavy sensors and backpack of the previous system have been replaced by a small *HiBall* unit (Figure 4, bottom). In addition, the precisely machined LED ceiling panels of the previous system have been replaced by lower-tolerance panels that are relatively inexpensive to make and simple to install (Figure 4, top; Figure 10). Finally, we are using an unusual Kalman-filter-based algorithm that generates very accurate pose estimates at a high rate with low latency, and simultaneously self-calibrates the system.

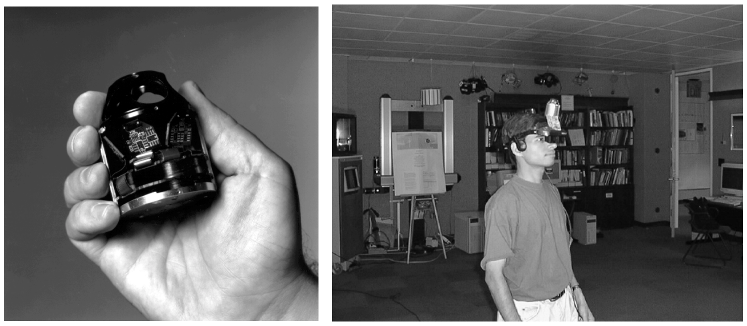


Figure 4

As a result of these improvements the HiBall Tracking System can generate over 2000 pose estimates per second, with less than one millisecond of latency, better than 0.5 millimeters and 0.03 degrees of absolute error and noise, everywhere in a 4.5 by 8.5 meter room (with over two meters of height variation). The area can be expanded by adding more panels, or by using checkerboard configurations which spread panels over a larger area. The weight of the user-worn HiBall is about 300 grams, making it lighter than one optical sensor in the 1991 system. Multiple HiBall units can be daisy-chained together for head or hand tracking, pose-aware input devices, or precise 3D point digitization throughout the entire working volume.

2. DESIGN CONSIDERATIONS

In all of the optical systems we have developed (see Section 1.2) we have chosen what we call an *inside-looking-out* configuration, where the optical sensors are on the (moving) user and the *landmarks* (e.g., LEDs) are fixed in the laboratory. The corresponding *outside-looking-in* alternative would be to place the landmarks on the user, and to fix the optical sensors in the laboratory. (One can think about similar outside-in and inside-out distinctions for acoustic and magnetic technologies.) The two configurations are depicted in Figure 5.

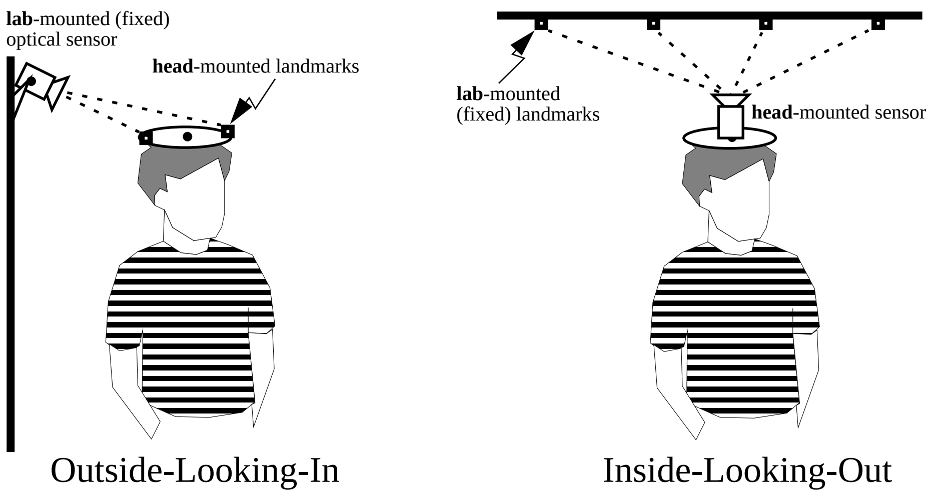


Figure 5

There are some disadvantages to the inside-looking-out approach. For small or medium-sized working volumes, mounting the sensors on the user is more challenging than mounting them in the environment. It is difficult to make user-worn sensor packaging small, and communication from the moving sensors to the rest of the system is more complex. In contrast, there are fewer mechanical considerations when mounting sensors in the environment for an *outside-looking-in* configuration. Because landmarks can be relatively simple, small, and cheap, they can often be located in numerous places on the user, and communication from the user to the rest of the system can be relatively simple or even unnecessary. This is particularly attractive for full-body motion capture (BL, 2000; MAC, 2000).

However there are some significant advantages to the inside-looking-out approach for head tracking. By operating with sensors on the user rather than in the environment, the system can be scaled indefinitely. The system can evolve from using dense active landmarks to fewer, lower signal-to-noise ratio, passive, and some day natural features for a Self-Tracker that operates entirely without landmark infrastructure (Bishop, 1984; Bishop & Fuchs, 1984; Welch, 1995).

The inside-looking-out configuration is also motivated by a desire to maximize sensitivity to changes in user pose. In particular, a significant problem with an outside-looking-in configuration is that only position estimates can be made directly, and so orientation must be inferred from position estimates of multiple fixed landmarks. The result is that orientation sensitivity is a function of both the *distance to the landmarks* from the sensor and the *baseline between the landmarks* on the user. In particular, as the distance to the user increases or the baseline between the landmarks decreases the sensitivity goes down. For sufficient orientation sensitivity one would likely need a baseline that is considerably larger than the user's head. This would be undesirable from an ergonomic standpoint and could actually restrict the user's motion.

With respect to translation, the change in measured photo-coordinates is the same for an environment-mounted (fixed) sensor and user-mounted (moving) landmark as it is for a user-mounted sensor and an environment-mounted landmark. In other words, the translation and corresponding sensitivity are the same for either case.

3. SYSTEM OVERVIEW

The HiBall Tracking System consists of three main components (Figure 6). An outward-looking sensing unit we call the *HiBall* is fixed to each user to be tracked. The HiBall unit observes a subsystem of fixed-location infrared LEDs we call the *Ceiling*[1]. Communication and synchronization between the host computer and these subsystems is coordinated by the *Ceiling-HiBall Interface Board* (CIB). In Section 4 we describe these components in more detail.

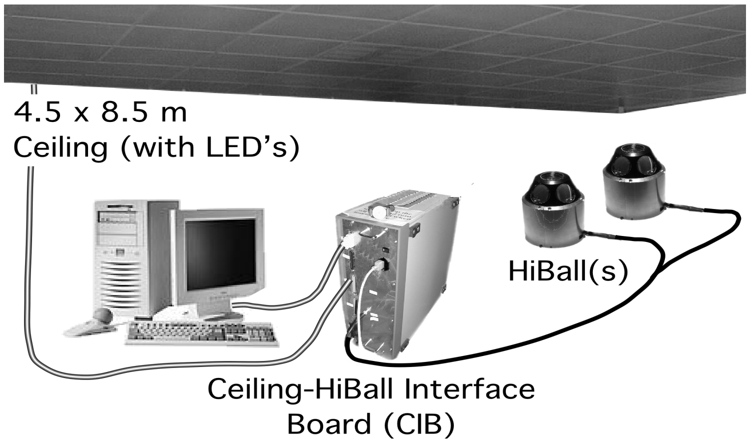


Figure 6

Each HiBall observes LEDs through multiple sensor-lens *views* that are distributed over a large solid angle. LEDs are sequentially flashed (one at a time) such that they are seen via a diverse set of views for each HiBall. Initial *acquisition* is performed using a brute force search through LED space, but once initial lock is made, the selection of LEDs to flash is tailored to the views of the active HiBall units. Pose estimates are maintained using a Kalman-filter-based prediction-correction algorithm known as *single-constraint-at-a-time* or SCAAT tracking. This technique has been extended to provide self-calibration of the Ceiling, concurrent with HiBall tracking. In Section 5 we describe the methods we employ, including the initial acquisition process and the SCAAT approach to pose estimation, with the *autocalibration* extension.

4. SYSTEM COMPONENTS

4.1 The HiBall

The original electro-optical tracker (Figure 3) used independently-housed lateral effect photo-diode units (LEPDs) attached to a light-weight tubular framework. As it turns out, the mechanical framework would flex (distort) during use, contributing to estimation errors. In part to address this problem the HiBall sensor unit was designed as a single rigid hollow ball having dodecahedral symmetry, with lenses in the upper six faces and LEPD on the insides of the opposing six lower faces (Figure 7). This immediately gives six primary "camera" *views* uniformly spaced by 57 degrees. The views efficiently share the same internal air space, and are rigid with respect to each other. In addition, light entering any lens sufficiently off axis can be

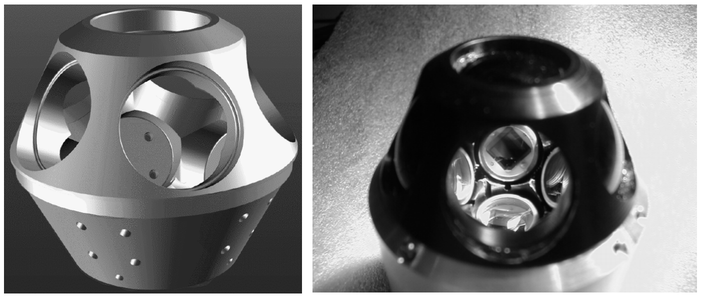


Figure 7

[1] At the present time, the LEDs are in fact entirely located in the ceiling of our laboratory, hence the sub-system name *Ceiling*, but LEDs could as well be located on walls or other fixed locations.

seen by a neighboring LEPD, giving rise to five secondary views through the top or central lens, and three secondary views through the five other lenses. Overall, this provides 26 fields of view which are used to sense widely separated groups of LEDs in the environment. While the extra views complicate the initialization of the Kalman filter as described in Section 5.5, they turn out to be of great benefit during steady-state tracking by effectively increasing the overall HiBall field of view without sacrificing optical sensor resolution.

The lenses are simple plano-convex fixed focus lenses. Infrared (IR) filtering is provided by fabricating the lenses themselves from RG-780 Schott glass filter material which is opaque to better than 0.001% for all visible wavelengths, and transmissive to better than 99% for IR wavelengths longer than 830 nm. The longwave filtering limit is provided by the DLS-4 LEPD silicon photodetector (UDT Sensors, Inc.) with peak responsivity at 950 nm but essentially blind above 1150 nm.

The LEPDs themselves are not imaging devices; rather they detect the centroid of the luminous flux incident on the detector. The x-position of the centroid determines the ratio of two output currents, while the y-position determines the ratio of two other output currents. The total output current of each pair are commensurate, and proportional to the total incident flux. Consequently, focus is not an issue, so the simple fixed-focus lenses work well over a range of LED distances from about half a meter to infinity. The LEPDs and associated electronic components are mounted on a custom rigid-flex printed circuit board (Figure 8). This arrangement makes efficient use of the internal HiBall volume while maintaining isolation between analog and digital circuitry, and increasing reliability by alleviating the need for inter-component mechanical connectors.

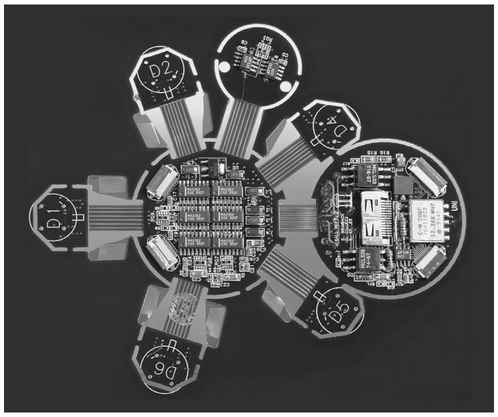


Figure 8

Figure 9 shows the physical arrangement of the folded electronics in the HiBall. Each LEPD has four transimpedance amplifiers (shown together as one “Amp” in Figure 9), the analog outputs of which are multiplexed with those of the other LEPDs, then sampled, held, and converted by four 16-bit Delta-Sigma analog-to-digital (A/D) converters. Multiple samples are integrated via an accumulator. The digitized LEPD data are organized into packets for communication back to the CIB. The packets also contain information to assist in error-detection. The communication protocol is simple, and while presently implemented by wire, the modulation scheme is amenable to a wireless implementation. The present wired implementation allows multiple HiBall units to be daisy-chained so a single cable can support a user with multiple HiBall units.

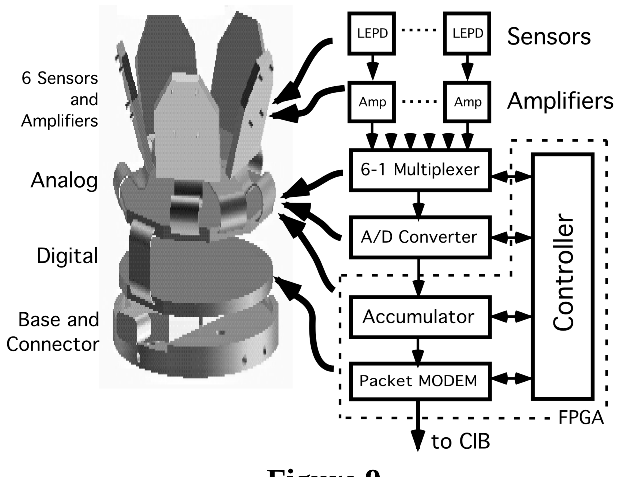


Figure 9

4.2 The Ceiling

As presently implemented, the infrared LEDs are packaged in 61 centimeter square *panels*, to fit a standard false ceiling grid (Figure 10, top). Each panel uses five printed circuit boards: a main controller board and four identical transverse-mounted *strips* (bottom). Each strip is populated with eight LEDs for a total of 32 LEDs per panel. We mount the assembly on top of a metal panel such that the LEDs protrude through 32 corresponding holes. The design results in a Ceiling with a rectangular LED pattern with periods of 7.6 and 15.2 centimeters. This spacing is used for the initial estimates of the LED positions in the lab, then during normal operation the SCAAT algorithm continually refines the LED position estimates (Section 5.4). The SCAAT *autocalibration* not only relaxes design and installation constraints, but provides greater precision in the face of initial and ongoing uncertainty in the Ceiling structure.

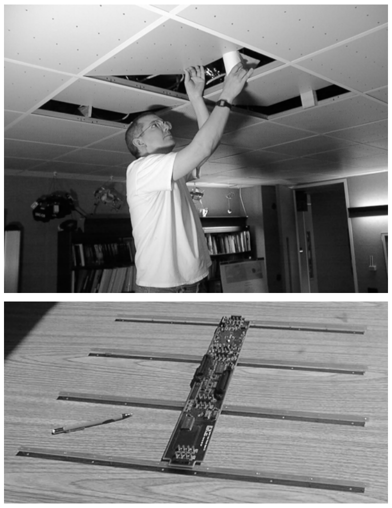


Figure 10

We currently have enough panels to cover an area approximately 5.5 by 8.5 meters with a total of approximately 3,000 LEDs.[1] The panels are daisy-chained to each other, and panel selection encoding is position (rather than device) dependent. Operational commands are presented to the first panel of the daisy chain. At each panel, if the panel select code is zero the controller decodes and executes the operation; else it decrements the panel select code and passes it along to the next panel (controller). Upon decoding, a particular LED is selected and the LED is energized. The LED brightness (power) is selectable for *automatic gain control* as described in Section 5.2.

We currently use Siemens SFH-487P GaAs LEDs which provide both a wide angle radiation pattern and high peak power, emitting at a center wavelength of 880 nm in the near IR. These devices can be pulsed up to 2.0 Amps for a maximum duration of 200 μs with a 1:50 (on:off) duty cycle. While the current Ceiling architecture allows flashing of only one LED at a time, LEDs may be flashed in any sequence. As such no single LED can be flashed too long or too frequently. We include both hardware and software protection to prevent this.

4.3 The Ceiling-HiBall Interface Board

The Ceiling-HiBall Interface Board or CIB (Figure 11) provides communication and synchronization between a host personal computer, the HiBall (Section 4.1), and the Ceiling (Section 4.2). The CIB has four Ceiling ports allowing interleaving of ceiling panels for up to four simultaneous LED flashes and/or higher Ceiling bandwidth. (The Ceiling bandwidth is inherently limited by LED power restrictions as described in Section 4.2, but this can be increased by spatially multiplexing the Ceiling panels.) The CIB has two tether interfaces that can communicate with up to four daisy-chained HiBall units. The full-duplex communication with the HiBall units uses a modulation scheme (BPSK)

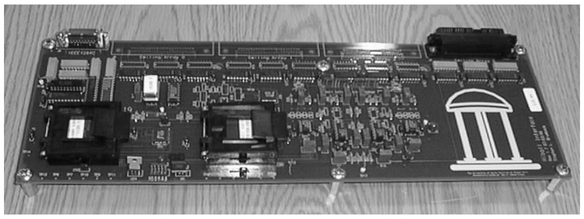


Figure 11

[1] The area is actually L-shaped; a small storage room occupies one corner.

allowing future wireless operation. The interface from the CIB to the host PC is the stable IEEE1284C extended parallel port (EPP) standard.

The CIB comprises analog drive and receive components as well as digital logic components. The digital components implement store and forward in both directions and synchronize the timing of the LED "on" interval within the HiBall dark-light-dark intervals. The protocol supports full-duplex flow control. The data are arranged into packets that incorporate error detection.

5. METHODS

5.1 Bench-Top (Off-Line) HiBall Calibration

After each HiBall is assembled we perform an off-line calibration procedure to determine the correspondence between image-plane coordinates and rays in space. This involves more than just determining the view transform for each of the 26 views. Non-linearities in the silicon sensor and distortions in the lens (e.g., spherical aberration) cause significant deviations from a simple pin-hole camera model. We dealt with all of these issues through the use of a two-part camera model. The first part is a standard pin-hole camera represented by a 3x4 matrix. The second part is a table mapping real image-plane coordinates to ideal image-plane coordinates.

Both parts of the camera model are determined using a calibration procedure that relies on a goniometer (an angular positioning system) of our own design. This device consists of two servo motors mounted together such that one motor provides rotation about the vertical axis while the second motor provides rotation about an axis orthogonal to vertical. An important characteristic of the goniometer is that the rotational axes of the two motors intersect at a point at the center of the HiBall optical sphere; this point is defined as the origin of the HiBall. (It is this origin that provides the reference for the HiBall state during run time as described in Section 5.3.) The rotational positioning motors were rated to provide 20 arc-second precision; we further calibrated them to 6 arc seconds using a surveying grade theodolite—an angle measuring system.

In order to determine the mapping between sensor image-plane coordinates and three-space rays, we use a single LED mounted at a fixed location in the laboratory such that it is centered in the view directly out of the top lens of the HiBall. This ray defines the Z or up axis for the HiBall coordinate system. We sample other rays by rotating the goniometer motors under computer control. We sample each view with rays spaced about every 6 minutes of arc throughout the field of view. We repeat each measurement 100 times in order to reduce the effects of noise on the individual measurements and to estimate the standard deviation of the measurements.

Given the tables of approximately 2500 measurements for each of the 26 views, we first determine a 3 by 4 view matrix using standard linear least-squares techniques. Then we determine the deviation of each measured point from that predicted by the ideal linear model. These deviations are re-sampled into a 25 by 25 grid indexed by sensor-plane coordinates using a simple scan conversion procedure and averaging. Given a measurement from a sensor at run time (Section 5.2) we convert it to an "ideal" measurement by subtracting a deviation bilinearly interpolated from the nearest 4 entries in the table.

5.2 On-Line HiBall Measurements

Upon receiving a command from the CIB (Section 4.3), which is synchronized with a CIB command to the ceiling, the HiBall selects the specified LEPD and performs three measurements, one before the LED flashes, one during the LED flash, and one after the LED flash. Known as "dark-light-dark", this technique is used to subtract out DC bias, low frequency noise, and background light from the LED signal. We then convert the measured sensor coordinates to “ideal” coordinates using the calibration tables described in Section 5.1.

In addition, during run time we attempt to maximize the signal-to-noise ratio of the measurement with an automatic gain control scheme. For each LED we store a target signal strength constant. We compute the LED current and number of integrations (of successive accumulated A/D samples) by dividing this strength constant by the square of the distance to the LED, estimated from the current position estimate. After a reading we look at the strength of the actual measurement. If it is larger than expected we reduce the gain, if it is less than expected we increase the gain. The increase and decrease are implemented as on-line averages with scaling such that the gain constant decreases rapidly (to avoid overflow) and increases slowly. Finally we use the measured signal strength to estimate the noise on the signal using (Chi, 1995), and then use this as the measurement noise estimate for the Kalman filter (Section 5.3).

5.3 Recursive Pose Estimation (SCAAT)

The on-line measurements (Section 5.2) are used to estimate the pose of the HiBall during operation. The 1991 system 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* (Azuma & Ward, 1991) to estimate the pose of the sensor fixture shown in Figure 3 (bottom). 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 a group of measurements (typically 20 or more) 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, the approach we use with the new HiBall system produces tracker reports as each new measurement is made, rather than waiting to form a complete collection of observations. 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 & Bishop, 1997). The key is that the single measurements provide *some* information about the user's 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 the HiBall (Section 5.4).

We use a Kalman filter (Kalman, 1960) to fuse the measurements into an estimate of the HiBall *state* $\bar{x}$ (the pose of the HiBall). We use a Kalman filter—a minimum variance stochastic estimator—both because the sensor measurement noise and the typical user motion dynamics can be modeled as normally-distributed random processes, and because we want an efficient on-line method of estimation. A basic introduction to the Kalman filter can be found in Chapter 1 of (Maybeck, 1979), while a more complete introductory discussion can be found in (Sorenson, 1970), which also contains some interesting historical narrative. More extensive references can be found in (Brown & Hwang, 1992; Gelb, 1974; Jacobs, 1993; Lewis, 1986; Maybeck, 1979; Welch & Bishop, 1995). Finally, we maintain a Kalman filter web page (Welch & Bishop, 2000) with introductory, reference, and research material.

The Kalman filter has been used previously to address similar or related problems. See for example (Azarbayejani & Pentland, 1995; Azuma, 1995; Emura & Tachi, 1994; Fuchs (Foxlin), 1993; Mazuryk & Gervautz, 1995; Van Pabst & Krekel, 1993). A relevant example of a Kalman filter used for sensor fusion in wide-area tracking system is given by (Foxlin et al., 1998) which describes a hybrid inertial-acoustic system that is commercially-available today (Intersense, 2000).

The SCAAT approach is described in detail in (Welch, 1996; Welch & Bishop, 1997). Included there is discussion of the benefits of using the approach, as opposed to a *multiple-constraint* approach such as (Azuma & Ward, 1991). However one key benefit warrants discussion here. There is a direct relationship between the *complexity* of the estimation algorithm, the corresponding *speed* (execution time per estimation cycle), and the *change* in HiBall pose between estimation cycles (Figure 12). As the algorithmic complexity increases, the execution time increases, which allows for significant non-linear HiBall motion between estimation cycles, which in turn implies the need for a more complex estimation algorithm.

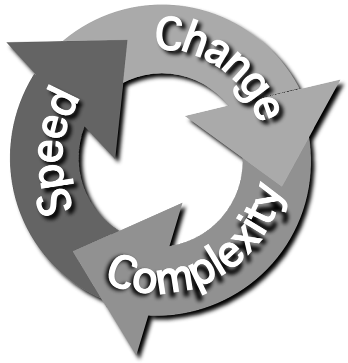


Figure 12

The SCAAT approach on the other hand is an attempt to reverse this cycle. Because we intentionally use a single constraint per estimate, the algorithmic complexity is drastically reduced, which reduces the execution time, and hence the amount of motion between estimation cycles. Because the amount of motion is limited we are able to use a simple dynamic (process) model in the Kalman filter, which further simplifies the computations. In short, the simplicity of the approach means it can run very fast, which means it can produce estimates very rapidly, with low noise.

The Kalman filter requires both a model of the process dynamics, and a model of the relationship between the process state and the available measurements. In part due to the simplicity of the SCAAT approach we are able to use a simple PV (position-velocity) process model (Brown & Hwang, 1992). Consider the simple example state vector $\bar{x}(t) = [\bar{x}_p(t), \bar{x}_v(t)]^T$, where the first element $\bar{x}_p(t)$ is the pose (position or orientation) and the second element $\bar{x}_v(t)$ is the corresponding velocity, i.e. $\bar{x}_v(t) = \frac{d}{dt}\bar{x}_p(t)$. We model the continuous change in the HiBall state with the simple differential equation

$$\frac{d}{dt}\bar{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_p(t) \\ \bar{x}_v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \mu \end{bmatrix} u(t), \tag{1}$$

where $u(t)$ is a normally-distributed white (in the frequency spectrum) scalar noise process, and the scalar μ represents the magnitude or *spectral density* of the noise. We use a similar model with a distinct noise process for each of the six pose elements. We determine the individual noise magnitudes using an off-line simulation of the system and a non-linear optimization strategy that seeks to minimize the variance between the estimated pose and a known motion path. (See Section 6.2.2.) The differential equation (1) represents a continuous integrated random walk, or an integrated *Wiener* or *Brownian-motion* process. Specifically, we model each component of the linear and angular HiBall velocities as a random walk, and then use these (assuming constant inter-measurement velocity) to estimate the HiBall pose at time $t + \delta t$ as follows:

$$\bar{x}(t + \delta t) = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix} \bar{x}(t) \tag{2}$$

for each of the six pose elements. In addition to a relatively simple process model, the HiBall measurement model is relatively simple. For any Ceiling LED (Section 4.2) and HiBall view (Section 4.1), the 2D sensor measurement can be modeled as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \bar{c}_x / \bar{c}_z \\ \bar{c}_y / \bar{c}_z \end{bmatrix} \tag{3}$$

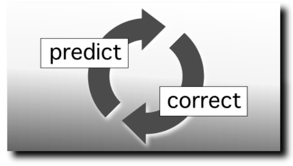
where

$$\begin{bmatrix} \bar{c}_x \\ \bar{c}_y \\ \bar{c}_z \end{bmatrix} = VR^{\mathrm{T}}(\dot{l}_{xyz} - \bar{x}_{xyz}), \tag{4}$$

V is the camera viewing matrix from Section 5.1, $\dot{l}_{xyz}$ is the position of the LED in the world, $\bar{x}_{xyz}$ is the position of the HiBall in the world, and R is a rotation matrix corresponding to the orientation of the HiBall in the world. In practice we maintain the orientation of the HiBall as a combination of a global (external to the state) quaternion and a set of incremental angles as described in (Welch, 1996; Welch & Bishop, 1997).

Because the measurement model (3)-(4) is non-linear we use an *extended Kalman filter,* making use of the Jacobian of the non-linear HiBall measurement model to transform the covariance of the Kalman filter. While this approach does not preserve the presumed Gaussian nature of the process, it has been used successfully in countless applications since the introduction of the (linear) Kalman filter. Based on observations of the statistics of the HiBall filter residuals, the approach also appears to work well for the HiBall. In fact it is reasonable to expect that it would, as the speed of the SCAAT approach minimizes the distance (in state space) over which we use the Jacobian-based linear approximation. This is another example of the importance of the relationship shown in Figure 12.

At each estimation cycle, the next of the 26 possible views is chosen randomly. Four points corresponding to the corners of the LEPD sensor associated with that view are projected into the world using the 3 by 4 viewing matrix for that view, along with the current estimates of the HiBall pose. This projection, which is the inverse of the measurement relationship described above, results in four rays extending from the sensor into the world. The intersection of these rays and the approximate plane of the Ceiling determines a 2D bounding box on the Ceiling, within which are the candidate LEDs for the current view. One of the candidate LEDs is then chosen in a least-recently-used fashion to ensure a diversity of constraints.



Once a particular view and LED have been chosen in this fashion, the CIB (Section 4.3) is instructed to flash the LED and take a measurement as described in Section 5.2. This single measurement is compared with a prediction obtained using (3), and the difference or *residual* is used to update the filter state and covariance matrices using the *Kalman gain* matrix. The Kalman gain is computed as a combination of the current filter covariance, the measurement noise variance (Section 6.2.1), and the Jacobian of the measurement model. This recursive prediction-correction cycle continues in an ongoing fashion, a single constraint at a time.

A more detailed discussion of the HiBall Kalman filter and the SCAAT approach is beyond the scope of this paper. For additional information see (Welch, 1996; Welch & Bishop, 1997).

5.4 On-line LED Autocalibration

Along with the benefit of simplicity and speed, the SCAAT approach offers the additional capability of being able to estimate the 3D positions of the LEDs in the world concurrently with the pose of the HiBall, on line, in real time. This capability is a tremendous benefit in terms of the accuracy and noise characteristics of the estimates. Accurate LED position estimates are so important that prior to the introduction of the SCAAT approach a specialized off-line approach was developed to address the problem (Gottschalk & Hughes, 1993).

The method we now use for autocalibration involves defining a distinct SCAAT Kalman filter for each LED. Specifically, for each LED we maintain a state $\bar{l}$ (estimate of the 3D position) and a 3x3 Kalman filter covariance. At the beginning of each estimation cycle we form an augmented state vector $\widehat{x}$ using the appropriate LED state and the current HiBall state: $\widehat{x} = [\bar{x}^{\mathrm{T}}, \bar{l}^{\mathrm{T}}]^{\mathrm{T}}$. Similarly we augment the Kalman filter error covariance matrix with that of the LED filter. We then follow the normal steps outlined in Section 5.3, with the result being that the LED portion of the filter state and covariance is updated in accordance with the measurement residual. At the end of the cycle we extract the LED portions of the state and covariance from the augmented filter, and save them externally. The effect is that as the system is being used, it continually refines its estimates of the LED positions, thereby continually improving its estimates of the HiBall pose. Again, for additional information see (Welch, 1996; Welch & Bishop, 1997).

5.5 Initialization and Re-Acquisition

The recursive nature of the Kalman filter (Section 5.3) requires that the filter be initialized with a known state and corresponding covariance before steady-state operation can begin. Such an initialization or *acquisition* must take place prior to any tracking session, but also upon the (rare) occasion when the filter diverges and “loses lock” as a result of blocked sensor views for example.

The acquisition process is complicated by the fact that each LEPD sees a number of different widely separated views (Section 4.1). Therefore detecting an LED provides at best an ambiguous set of potential LED directions in HiBall coordinates. Moreover, before acquisition no assumptions can be made to limit the search space of visible LEDs. As such, a relatively slow brute-force algorithm is used to acquire lock.

We begin with an exhaustive LED scan of sufficiently fine granularity to ensure that the central primary field of view is not missed. For the present Ceiling, we flash every 13th LED in sequence, and look for it with the central LEPD until we get a hit. Then a sufficiently large patch of LEDs, centered on the hit, is sampled to ensure that several of the views of the central LEPD will be hit. The fields of view are disambiguated by using the initial hits to estimate the yaw of the HiBall (rotation about vertical), and finally more selective measurements are used to refine the acquisition estimate sufficiently to switch into tracking mode.

6. RESULTS

Three days after the individual pieces of hardware were shown to be functioning properly we demonstrated a complete working system. After months of subsequent tuning and optimization, the system continues to perform both qualitatively and quantitatively as well, or in some respects *better*, than we had anticipated (Section 6.1). The articulation of this success is not meant to be self-congratulatory, but to give credit to the extensive and careful modeling and simulation performed prior to assembly (Section 6.2). In fact, the Kalman filter parameters found by the optimization procedure described in Section 6.2.2 were, and continue to be, used directly in the working system. Likewise much of the software written for the original simulations continues to be used in the working system.

6.1 On-Line Operation

The HiBall system is in daily use as a tool for education and research. For example, it was used by Martin Usoh et al. to perform Virtual Reality experiments comparing virtual "flying", walking in place, and real walking (Usoh et al., 1999). The researchers used the HiBall system to demonstrate that as a mode of locomotion, real walking is simpler, more straightforward, and more natural, than both virtual flying and walking in place. The unprecedented combination of large working volume and high performance of the HiBall system led the researchers to claim that there was nowhere else that they could have meaningfully performed the experiments.

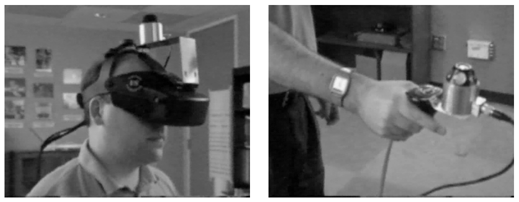


Figure 13

6.1.1 Robustness. As a result of a mechanical design trade-off, each sensor field of view is less than six degrees. The focal length is set by the size of the sensor housing, which is set by the diameter of the sensors themselves. Energetics is also a factor, limiting how small the lenses can be while maintaining sufficient light collecting area. As a result of these design trade-offs, even a momentary small error in the HiBall pose estimate can cause the recursive estimates to diverge and the system to lose lock after only a few LED sightings. And yet the system is quite robust. In practice users can jump around, crawl on the floor, lean over, even wave their hands in front of the sensors, and the system does not lose lock. During one session we were using the HiBall as a 3D *digitization probe*, a HiBall on the end of a pencil-shaped fiberglass wand (Figure 14, left). We laid the probe down on a table at one point, and were amazed to later notice that it was still tracking, even though it was only observing 3 or 4 LEDs near the edge of the Ceiling. We picked up the probe and continued using it, without it ever losing lock.

6.1.2 Estimate Noise. The simplest quantitative measurement of estimate noise is the standard deviation of the estimates when a HiBall is held stationary. With a tracker as sensitive as the HiBall it is important to be certain that it really is stationary. The raised floor in our laboratory allows motion, for example when a person walks by, that is larger than the expected error in the HiBall. We made careful measurements by resting the support for the HiBall on the concrete sub-floor in our laboratory. The standard deviation of the HiBall estimates while stationary was about 0.2 millimeters and 0.03 degrees. The distribution of the noise fit a normal distribution quite well.

To make measurements of the noise when the HiBall is in motion we rely on the assumption that almost all of the signal resulting from normal human motion is at frequencies below 2 Hz. We use a high-pass filter (Welch, 1967) on the pose estimates, and assume the output is noise. The resulting statistics are comparable to those made with the HiBall stationary, except at poses for which there are very few LEDs visible in only one or two views. In these poses, near the edge of the ceiling, the geometry of the constraints results in amplification of errors. For nearly all of the working volume of the tracker the standard-deviation of the noise on measurements while the HiBall is still or moving is about 0.2 millimeters and 0.03 degrees.

6.1.3 Absolute Accuracy. We have performed several experiments to measure the accuracy of the HiBall system, however the most objective experiment took place in July of 1999. Boeing Phantom Works scientists David Himmel and David Princehouse (Associate Technical Fellows) visited our laboratory for two days to assess the accuracy of the HiBall system, and its potential use in providing assembly workers with real-time feedback on the pose of hand-held pneumatic drills during the aircraft manufacturing process. (The right image in Figure 14 shows the HiBall attached to a pneumatic drill.)

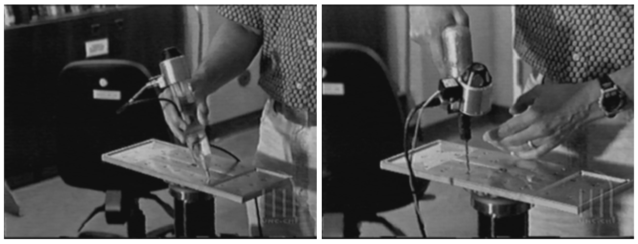


Figure 14

The scientists designed some controlled experiments to asses the accuracy of the HiBall system. They brought with them an aluminum "coupon" (see Figure 14 and Figure 15) with 27 shallow holes pre-drilled on 1.5 inch centers using a numerically-controlled milling machine with a stated accuracy of $1/1000$ inch. The holes (except one) were not actually drilled through the coupon, but instead formed conical dimples with a fine point at the center. The center-most hole (hole 14) was actually drilled completely through to provide a mounting point. Using that hole we attached the coupon to a military-grade tripod situated on the (false) floor of our laboratory, under the HiBall Ceiling. As shown in the left image of Figure 14 we mounted the HiBall on our standard probe, a rigid plastic pencil-like object with a pointed steel tip. We used one of the pre-drilled coupon holes to perform our normal HiBall probe calibration procedure, which involves placing the tip of the probe in the hole, pivoting the probe about the point while collecting several seconds of pose data, and then estimating the transformation from the HiBall to the probe tip. (We have a standard application that assists us with this procedure.) Together with Himmel and Princehouse we performed several experiments where we placed the tip of the HiBall probe in each hole in succession, sampling the HiBall pose estimates only when we pressed the probe button. We performed several such sessions over the course of one afternoon and the next morning (we re-calibrated the probe in the morning).

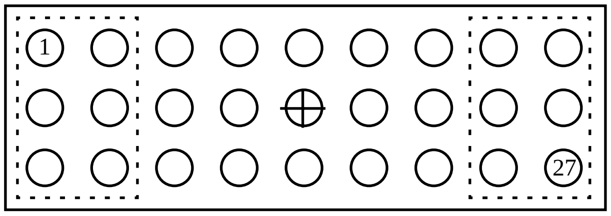


Figure 15

For the data from each session we used a least-squares optimization method to find an estimate of the full 6D transformation (translation and rotation) that minimized the Euclidian distance from the probe data to a 2D plane with 27 holes on 1.5 inch spacing. The resulting fit consistently corresponded to an average positioning error of $20/1000$ inch ($1/2$ millimeter) at the metal tip of the HiBall probe, which is within the target Boeing specifications. The system might actually be more accurate than our experiments indicated. For one, the diameter of the (rounded) tip of the HiBall probe is $1/2$ millimeter. In addition, at the time of the experiments we unfortunately did not heed our own advice to position the experimental platform on the rigid concrete sub-floor. In any case we are encouraged by the results, and are excited about the possibility that the HiBall system has uses beyond tracking for Virtual Reality.

6.2 Off-Line Simulation and Modeling

During the design of the HiBall system we made substantial use of simulation, in some domains to a very detailed level. For example, Zemax (Focus Software, 1995) was used extensively in the design and optimization of the optical design, including the design of the filter glass lenses, and geometry of the optical component layout. AutoCAD™ was used to design, specify, and fit-check the HiBall body mechanicals, to visualize the physical design, and to transmit the design to our collaborators at the University of Utah for fabrication by the *Alpha 1* System (Thomas, 1984; University of Utah Computer Science, 1999). A custom ray-tracing system was built by Stefan Gottschalk (UNC) for the purpose of evaluating the optical behavior and energetics of the primary, secondary, and tertiary fields of view; the results were used by the noise model developed in (Chi, 1995) as described in the next section.

In addition, a complete simulator of the system was written in C++. This simulator, discussed further in Section 6.2.2, was used to evaluate the speed, accuracy, and robustness of the system. In addition it was used to "tune" the Kalman filter for realistic motion dynamics. This simulator continues to be used to evaluate mechanical, optical, and algorithmic alternatives.

6.2.1 HiBall Measurement Noise Model. Signal-to-noise performance is a prime determiner of both accuracy and speed of the system, so an in-depth study (Chi, 1995) was performed to develop a detailed noise model accounting for properties of the LED, the LEPD (sensor), the optical system, the physical distance and pose, the electronics, and the dark-light-dark integrations described in Section 5.2. The predominant noise source is shot noise, with Johnson noise in the sheet resistivity of the LEPD surfaces being the next most significant. Careful measurements made in the laboratory with the actual devices yielded results that were almost identical to those predicted by the sophisticated model in (Chi, 1995). A simplified version of this model is used in the real system with the automatic gain control (Section 5.2) to predict the measurement noise for the Kalman filter (Section 5.3).

6.2.2 Complete System Simulations. To produce realistic data for developing and tuning our algorithms we collected several motion paths *(*sequences of pose estimates) from our first generation electro-optical tracker (Figure 3) at its 70 Hz maximum report rate. These paths were recorded from both naive users visiting our monthly "demo days" and from experienced users in our labs. In the same fashion as we had done for (Azuma & Bishop, 1994a) we filtered the raw path data with a non-causal zero-phase-shift low-pass filter to eliminate energy above 2 Hz. The output of the low-pass filtering was then re-sampled at whatever rate we wanted to run the simulated tracker, usually 1000 Hz. For the purposes of our simulations we considered these re-sampled paths to be the "truth"—a perfect representation of a user's motion. Tracking error was determined by comparing the "true" path to the estimated path produced by the tracker.

The simulator reads camera models describing the 26 views, the sensor noise parameters, the LED positions and their expected error, and the motion path described above. Before beginning the simulation, the LED positions are perturbed from their ideal positions by adding normally distributed error to each axis. Then, for each simulated cycle of operation, the "true" pose are updated using the input motion path. Next, a view is chosen and a visible LED within that view is selected, and the image-plane coordinates of the LED on the chosen sensor are computed using the camera model for the view and the LED as described in Section 5.3. These sensor coordinates are then perturbed based on the sensor noise model (Section 6.2.1) using the distance and angle to the LED. Now these noise corrupted sensor readings are fed to the SCAAT filter to produce an updated position estimate. The position estimate is compared to the true position to produce a scalar error metric described next.

The error metric we used combines the error in pose in a way that relates to the effects of tracker error on a head-worn display user. We define a set of points arrayed around the user in a fixed configuration. We compute two sets of coordinates for these points; the true position using the true pose, and their estimated position using the estimated pose. The error metric is then the sum of the distances between the true and estimated positions of these points. By adjusting the distance of the points from the user we can control the relative importance of the orientation and the position error in the combined error metric. If the distance is small, then the position error is weighted most heavily; if the distance is large then the orientation error is weighted most heavily. Our two error metrics for the entire run are the square-root of the sum of the squares of all the distances, and the peak distance.

6.2.3 Tuning. Determining the magnitudes of the SCAAT Kalman filter noise parameters (Section 5.3) is called *system identification* or *tuning*. We use Powell's method (Press, Teukolsky, Vetterling, & Flannery, 1990) to minimize the error metric described above. Starting with a set of parameters we run the simulator over a full motion run to determine the total error for the run. The optimizer makes a small adjustment to the parameters and the process is repeated. These runs required hours of computer time and some skill (and luck) in choosing the initial parameters and step sizes. Of course, it is important to choose motion paths that are representative of expected target motion. For example, a run in which the target is very still would result in very different tuning from a run in which the target moves very vigorously.

7. FUTURE WORK

7.1 Improving the HiBall

The current SCAAT filter form (Section 5.3) and tuning values (Section 6.2.3) are a compromise between the responsiveness desired for high dynamics, and the heavy filtering desired for smooth estimates during very slow or no motion. As such we are investigating the use of a multi-modal or *multiple-model* Kalman filter framework (Bar-Shalom & Li, 1993; Brown & Hwang, 1992). A multiple-model implementation of the HiBall should be able automatically, continuously, and smoothly choose between one Kalman filter tuned for high dynamics, and another tuned for little or no motion. We have this working in simulation, but not yet implemented in the real system.

As mentioned in Section 4.3, the system was designed to support wireless communication between the HiBall and the CIB, without significant modification or added information overhead. Despite the fact that commercial head-worn displays are themselves tethered at this time, we are beginning work on a completely wireless HiBall and head-worn display system. We also intend to use the wireless HiBall with projector-based displays where the user is otherwise wearing only polarized glasses. Furthermore the HiBall was designed with extra built-in digital input-output capabilities. We are considering possibilities for providing access to these signals for (wireless) user-centered input devices and even body-centric limb tracking.

Finally we note that a private startup company called 3rdTech (3rdTech, 2000) has negotiated a technology license with UNC for the existing HiBall Tracking System. 3rdTech is now marketing an updated system with simpler LED "strips" instead of ceiling panels.

7.2 Wide-Field-of-View HiBall

Beyond improving the existing system, we continue to head down a path of research and development that will lead to systems with reduced dependency on the laboratory infrastructure. For example, our current Ceiling panel design with 32 LEDs per panel, provides far more dense coverage than we believe is necessary. The density of Ceiling LEDs is a result of design based on the original sensor fixture show in Figure 3. Given a more sparse field of LEDs we believe that we could achieve similar performance with a version of the HiBall that has a small number of *wide field of view* optical sensor units. This would further reduce the packaging size of the user-worn sensor component.

7.3 To the Hallway and Beyond

By leveraging the knowledge gained from successful work in the laboratory, our long term goal is to achieve similar performance with little or no explicit infrastructure, for example throughout a building or even (some day) outdoors. While high-performance 6D tracking outdoors is a tremendous challenge that is unlikely to be solved any time soon, we believe that the eventual solution will involve a clever and careful combination of multiple complementary technologies. In particular we are pursuing the hybrid approach initially presented in (Welch, 1995). We look forward to a day when high-performance 6D tracking outdoors enables pose-aware devices for work such as Feiner's outdoor augmented reality (Feiner, MacIntyre, Höllerer, & Webster, 1997; Höllerer, Feiner, Terauchi, Rashid, & Hallaway, 1999), the "WorldBoard" initiative (Spohrer, 1999a, 1999b), and other wonderful applications.

8. ACKNOWLEDGEMENTS

We acknowledge former Tracker Project members and contributors (alphabetically): Ronald Azuma, Matthew Cutts, Henry Fuchs, Stefan Gottschalk, Jeffrey Juliano, Pawan Kumar, Benjamin Lok, John Thomas, Greg Turk, Jih-Fang Wang, Scott Williams, Hans Weber, Mary Whitton, and Philip Winston. We thank Al Barr (California Institute of Technology) and John "Spike" Hughes (Brown University) for their contributions to the original off-line LED calibration work that led to the simpler Ceiling panels (Figure 1 and Figure 10). Finally we want to acknowledge our many collaborators in the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (below), and in particular our collaborators in mechanical design and fabrication at the University of Utah: Rich Riesenfeld, Sam Drake, and Russ Fish.

This work was supported in part by DARPA/ETO contract number DABT 63-93-C-0048, "Enabling Technologies and Application Demonstrations for Synthetic Environments", Principal Investigators Frederick P. Brooks Jr. and Henry Fuchs (UNC); and by the National Science Foundation Cooperative Agreement no. ASC-8920219: "Science and Technology Center for Computer Graphics and Scientific Visualization," Center Director Rich Riesenfeld (University of Utah). Principal Investigators Al Barr (Caltech), Don Greenberg (Cornell University), Henry Fuchs (UNC), Rich Riesenfeld, and Andy van Dam (Brown University).

REFERENCES

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

Ascension. (2000). *Ascension Technology Corporation*, [HTML]. Ascension Technology Corporation. Available: http://www.ascension-tech.com/ [2000, September 15].

Azarbayejani, A., & Pentland, A. (1995). Recursive Estimation of Motion, Structure, and Focal Length. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *17*(6).

Azuma, R. T. (1993, July). Tracking Requirements for Augmented Reality. *Communications of the ACM*, *36*, 50-51.

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. (1994a). A Frequency-Domain Analysis of Head-Motion Prediction, *Computer Graphics* (SIGGRAPH 94 Conference Proceedings ed., pp. 401-408). Los Angeles, CA: ACM Press, Addison-Wesley.

Azuma, R. T., & Bishop, G. (1994b). 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.

Bhatnagar, D. K. (1993). *Position trackers for Head Mounted Display systems: A survey* (Technical Report TR93-010). Chapel Hill, NC USA: University of North Carolina at Chapel Hill.

Bishop, G. (1984). *The Self-Tracker: A Smart Optical Sensor on Silicon.* Unpublished Ph.D. Dissertation, University of North Carlina at Chapel Hill, Chapel Hill, NC USA.

Bishop, G., & Fuchs, H. (1984, January 23-25). *The Self-Tracker: A Smart Optical Sensor on Silicon.* Paper presented at the Advanced Research in VLSI, Massachusetts Institute of Technology.

BL. (2000). *CODA mpx30 Motion Capture System*, [html]. B & L Engineering. Available: http://www.bleng.com/animation/coda/codamain.htm [2000, April 27].

Brown, R. G., & Hwang, P. Y. C. (1992). *Introduction to Random Signals and Applied Kalman Filtering* (Second ed.): Wiley & Sons, Inc.

Burdea, G., & Coiffet, P. (1994). *Virtual Reality Technology* (First ed.): John Wiley & Sons, Inc.

Burton, R. P. (1973). *Real-Time Measurement of Multiple Three-Dimensional Positions.*, University of Utah, Salt Lake City, UT USA.

Burton, R. P., & Sutherland, I. E. (1974). Twinkle Box: Three-Dimensional Computer-Input Devices, *Proceedings of the National Computer Conference*. Chicago, IL USA.

Chi, V. L. (1995). *Noise Model and Performance Analysis Of Outward-looking Optical Trackers Using Lateral Effect Photo Diodes* (TR95-012). Chapel Hill, NC USA: University of North Carlina at Chapel Hill.

Emura, S., & Tachi, S. (1994). *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.

Feiner, S., MacIntyre, B., Höllerer, T., & Webster, A. (1997). A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring Urban Environments. *Personal Technologies, 1*(4), 208-217.

Focus Software. (1995). *ZEMAX Optical Design Program User's Guide, , Version 4.5*. Tucson, AZ USA.

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 (manual)*. Unpublished M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA USA.

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

Gottschalk, S., & Hughes, J. F. (1993). Autocalibration for Virtual Environments Tracking Hardware, *Computer Graphics* (SIGGRAPH 93 Conference Proceedings ed.). Anaheim, CA USA: ACM Press, Addison Wesley.

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., & Hallaway, D. (1999). Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics, 23*, 779-785.

IGT. (2000). *Image Guided Technologies*, [HTML]. Image Guided Technologies,. Available: http://www.imageguided.com/ [2000, September 15].

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.

Kadaba, M. P., & Stine, R. (2000). *Real-Time Movement Analysis Techniques and Concepts for the New Millennium in Sports Medicine*, [HTML]. Motion Analysis Corporation, Santa Rosa, CA USA. Available: http://www.motionanalysis.com/applications/movement/rtanalysis.html [2000, September 15].

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

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

MAC. (2000). *HiRes 3D Motion Capture System*, [html]. Motion Analysis Corporation. Available: http://www.motionanalysis.com/applications/movement/gait/3d.html [2000, September 15].

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).

Meyer, K., Applewhite, H., & Biocca, F. (1992). A Survey of Position Trackers. *Presence, a publication of the Center for Research in Journalism and Mass Communication*.

Mulder, A. (1994a). *Human Movement Tracking Technology* (Technical Report TR 94-1): School of Kinesiology, Simon Fraser University.

Mulder, A. (1994b, May 8, 1998). *Human Movement Tracking Technology: Resources*, [HTML]. School of Kinesiology, Simon Fraser University. Available: http://www.cs.sfu.ca/people/ResearchStaff/amulder/personal/vmi/HMTT.add.html [2000, September 15].

Mulder, A. (1998, May 8, 1998). *Human Movement Tracking Technology*, [HTML]. School of Kinesiology, Simon Fraser University. Available: http://www.cs.sfu.ca/people/ResearchStaff/amulder/personal/vmi/HMTT.pub.html [2000, September 15].

Polhemus. (2000). *Polhemus*, [HTML]. Polhemus. Available: http://www.polhemus.com/home.htm [2000, September 15].

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1990). *Numerical Recipes in C; The Art of Scientific Computing* (Second ed.): Cambridge University Press.

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

Spohrer, J. (1999a). Information in Places. *IBM Systems Journal, Pervasive Computing, 38*(4).

Spohrer, J. (1999b, June 16). *WorldBoard; What Comes After the WWW?*, [HTML]. Learning Communities Group, ATG, (c)Apple Computer, Inc. Available: http://worldboard.org/pub/spohrer/wbconcept/default.html [1999, December 24, 1999].

Sutherland, I. E. (1968). A head-mounted three dimensional display, *Proceedings of the 1968 Fall Joint Computer Conference, AFIPS Conference Proceedings* (Vol. 33, part 1, pp. 757-764). Washington, D.C.: Thompson Books.

Thomas, S. W. (1984, December). *The Alpha_1 Computer-Aided Geometric Design System in the Unix Environment.* Paper presented at the Computer Graphics and Unix Workshop.

UNC Tracker Project. (2000, July 10). *Wide-Area Tracking; Navigation Technology for Head-Mounted Displays*, [HTML]. Available: http://www.cs.unc.edu/~tracker [2000, July 18].

University of Utah Computer Science. (1999). *Alpha 1 Publications*, [HTML]. University of Utah, Department of Computer Science. Available: http://www.cs.utah.edu/projects/alpha1/a1_publications.html [1999, May 28].

Usoh, M., Arthur, K., Whitton, M. C., Bastos, R., Steed, A., Slater, M., & Brooks, F. P., Jr. (1999). Walking > Walking-in-Place > Flying, in Virtual Environments. In A. Rockwood (Ed.), *Computer Graphics* (SIGGRAPH 99 Conference Proceedings ed., pp. 359-364). Los Angeles, CA USA: ACM Press, Addison Wesley.

Van Pabst, J. V. L., & Krekel, P. F. C. (1993, September 20 - 22). *Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space.* Paper presented at the 7th International Conference on Image Analysis and Processing—, Capitolo, Monopoli, Italy.

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.

Wang, J.-F., Azuma, R. T., Bishop, G., Chi, V., Eyles, J., & Fuchs, H. (1990, April 16-20). *Tracking a Head-Mounted Display in a Room-Sized Environment with Head-Mounted Cameras.* Paper presented at the SPIE 1990 Technical Symposium on Optical Engineering and Photonics in Aerospace Sensing, Orlando, FL.

Wang, J.-f., Chi, V., & Fuchs, H. (1990, March 25-28). *A Real-time Optical 3D Tracker for Head-mounted Display Systems.* Paper presented at the Symposium on Interactive 3D Graphics, Snowbird, UT.

Ward, M., Azuma, R. T., Bennett, R., Gottschalk, S., & Fuchs, H. (1992, March 29 - April 1). *A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems.* Paper presented at the Symposium on Interactive 3D Graphics, Cambridge, MA USA.

Welch, G. (1995). *Hybrid Self-Tracker: An Inertial/Optical Hybrid Three-Dimensional Tracking System* (TR95-048). Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, Department of Computer Science.

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. (1995). *An Introduction to the Kalman Filter* (TR95-041). Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, Department of Computer Science.

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. (2000, January 23, 2000). *The Kalman Filter*, [html]. University of North Carlina at Chapel Hill. Available: http://www.cs.unc.edu/~welch/kalman/index.html [2000, April 29].

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, P. D. (1967). The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Transactions on Audio Electroacoust*, *AU*(15), 70-73.

Woltring, H. J. (1974). New Possibilities for Human Motion Studies by Real-Time Light Spot Position Measurement. *Biotelemetry, 1*, 132-146.