Registration Errors in Augmented Reality Systems

by

Richard Lee Holloway

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science

Chapel Hill

1995

Approved by:

________________________________ Advisor
Frederick P. Brooks, Jr.

________________________________ Reader
Vernon Chi

________________________________ Reader
Henry Fuchs

________________________________ Reader
Stephen Pizer

RICHARD LEE HOLLOWAY. Registration Errors in Augmented Reality Systems. (Under the direction of Frederick P. Brooks, Jr.)

ABSTRACT

Augmented reality (AR) systems combine three-dimensional computer-generated imagery with the view of the real environment in order to make unseen objects visible or to present additional information. A critical problem is that the computer-generated objects do not currently remain correctly registered with the real environment—objects aligned from one viewpoint appear misaligned from another and appear to swim about as the viewer moves. This *registration error* is caused by a number of factors, such as system delay, optical distortion, and tracker measurement error, and is difficult to correct with existing technology.

This dissertation presents a registration error model for AR systems and uses it to gain insight into the nature and severity of the registration error caused by the various error sources. My thesis is that a mathematical error model enables the system architect to determine

- which error sources are the most significant,
- the sensitivity of the net registration error to each error,
- the nature of the distortions caused by each type of error,
- the level of registration accuracy one can expect,

and also provides insights on how best to calibrate the system.

Analysis of a surgery planning application yielded the following main results:

- Even for moderate head velocities, system delay causes more registration error than all other sources combined;
- Using the eye's center of rotation as the eyepoint in the computer graphics model reduces the error due to eye rotation to zero for points along the line of gaze. This should obviate the need for eye tracking;
- Tracker error is a significant problem both in head tracking *and* in system calibration;
- The World coordinate system should be omitted when possible;

- Optical distortion is a significant error source, but correcting it computationally in the graphics pipeline often induces delay error larger than the distortion error itself;
- Knowledge of the nature of the various types of error facilitates identification and correction of errors in the calibration process.

Although the model was developed for see-through head-mounted displays (STHMDs) for surgical planning, many of the results are applicable to other HMD systems as well.

ACKNOWLEDGMENTS

I would like to thank my advisor, Fred Brooks, for his wisdom, advice, and guidance for all these years. Thanks also to my committee members:

Vern Chi, for insights into real-world systems and for the idea of a "locally correct center of projection," which was the seed for the idea of using the eye's center of rotation as the viewpoint;

Jefferson Davis, for being a collaborator and taking the time to tutor a Computer Science student in the art of plastic surgery;

Henry Fuchs, for support, encouragement, and inspiration over the years;

Steve Pizer, for guidance in the areas of medical imaging, image processing, error analysis, and Dutch pronunciation;

Jannick Rolland, for the many meetings to educate me about optics, for advice and suggestions on many aspects of this research, for the optical design of the UNC see-through HMD, and for her contributions in the area of system calibration.

I would also like to thank the following people:

Gary Bishop for his steadfast support, encouragement, and advice, as well as discussions about "no swimming" and ideas on registration error sources.

Warren Robinett for the idea of doing a dissertation in the area of augmented reality, and for all the discussions about transformations, quaternions, and VR software.

David Harrison and John Hughes for help through the years in building one contraption or another, and for help in procuring equipment.

Fay Ward and Kathy Tesh for lots of help on many occasions over the years.

Peggy Wetzel for help with video filming and editing.

Ron Azuma for sharing his data and methods, and for discussions about static and dynamic registration.

Mark Mine for discussions about delay and just-in-time rendering.

Russell Taylor for his network software and troubleshooting help.

The HMD, Ultrasound, and Tracker project members for their help.

Marc Olano, Anselmo Lastra, Brice Tebbs, Greg Turk and other Pixel-Planes folks who helped me understand the inner workings of the PPHIGS rendering code.

Vincent Carrasco for being the physician subject in the head-motion study.

Dinesh Manocha for help with mathematics.

Linda Houseman for help through the years, and for helping with the user study.

Kurtis Keller for designing and building various widgets for the see-through HMD.

Doug Holmgren for his part in the design and building of the UNC see-through HMD

Marc Levoy for suggesting craniofacial surgery as a driving problem.

Bernard Adelstein for information on dynamic tracker error.

Greg Turk for the alignment code used in the simulation in Chapter 6.

Dave Allen at Tektronix for CRT accuracy information.

Steve Work at Ascension for information on tracker accuracy.

All the Pixel-Planes 5 users who put up with my clutter and let me hog the machine for hours at a time.

Terry Yoo and Don Stone for providing the Grumbling Grad Student Support Network and Pool Hall.

Christopher Gould for suggesting I get a Master’s in Computer Science instead of doing a double major.

My family for their love and encouragement.

Alexa for joy and inspiration.

Most of all I thank

Barbara

for her steadfast support, patience, and love.

Financial support for this work came from the following grants:

- ARPA Contract DABT63-94-C-0048: "Enabling Technologies and Application Demonstrations for Synthetic Environments"
- The NSF/ARPA Science and Technology Center for Computer Graphics and Visualization (NSF prime contract 8920219)
- ONR Contract N00014-86-K-0680.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AR	Augmented reality
CAVE	CAVE automatic virtual environment
CRT	Cathode ray tube
CS	Coordinate system
CT	Computed tomography
FOV	Field of view
HMD	Head-mounted display
Hz	Hertz (cycles per second)
IPD	Interpupillary distance
LC	Lateral color
LCD	Liquid-crystal display
LM	Landmark
MI/P	Medical imaging and polygonalization
MRI	Magnetic resonance imaging
mm	Millimeter
ms	Millisecond
NTSC	National Television System Committee
RGB	Red, green, and blue
RMS	Root-mean square
STHMD	See-through head-mounted display

UNC	University of North Carolina at Chapel Hill
VCASS	Visually coupled airborne systems simulator
VE	Virtual environments
VR	Virtual reality
2D	Two dimensions, two-dimensional
3D	Three dimensions, three-dimensional

LIST OF SYMBOLS

The following is a complete list of all variables used; it is meant to serve as a quick reference rather than as a set of definitions. See Chapter 5 for a full explanation of notational conventions.

Primes indicate measured or calculated quantities; double primes indicate quantities with propagated error. Quantities with "left or right" designation may be prefixed with "L" or "R". **Boldface** quantities are vectors or quaternions.

A,B,C:	General point variables
D,D':	Display-device coordinate system (CS) (left or right)
E,E':	General eyepoint (left or right)
G,G':	Location in device coordinates of the displayed points (left or right)
H:	Midpoint between the eyepoints
I,I',I'':	Image CS (left or right); in matrix operations: identity matrix
O:	Object CS
M:	Matrix/General operator
N:	General purpose counting variable, usually used as limit of sum
N_{LM}:	Number of landmark points used in alignment algorithm
P:	Real landmark point on the patient
P':	Apparent landmark location after digitization, etc.
P'':	Location of P' after rigid-body transformation errors
P*:	Final, perceived point using the real eyepoints and real imaged points
Q, Q':	General displayed point in image CS (left or right)
R:	Rotation operator
S,S',S'':	Sensor CS

$S_{\mathbf{v}}(T)$:	Sensitivity of the operator T
T,T':	Tracker CS
T_{A_B}:	Transformation operator
W:	World CS
a:	General purpose variable
b:	Net registration error due to a particular error source (usually with subscript)
c:	Eyepoint calibration error magnitude
d, d':	Eye-to-screen image distance
d_{er}:	Eye-relief distance
e:	Viewing error magnitude
f:	focal distance of optical system
f:	General-purpose variable
g_x,g_y:	Size of display screen in mm
h_x,h_y:	Pixel dimensions in mm
i,i':	One half of interpupillary distance, also counting variable
j:	Counting variable
k:	Coefficient of optical distortion
m:	Magnification of optics
n:	General purpose counting variable
n:	Vector portion of quaternion (usually with subscript)
p:	Pupil location for optical system
p_x,p_y:	Frame buffer resolution in pixels
q:	Display error magnitude
q:	Quaternion (usually with subscript)
r:	Radial distance (general-purpose variable)
s:	Lateral error
s_i:	Image distance for optics

s_o:	Object distance for optics
t:	Depth error
$\mathbf{u}$, $\mathbf{v}$:	Vector (usually with subscript)
$\mathbf{v}_{de}$:	Vector characterizing display error for limited case
$\mathbf{w}$:	General purpose vector
w:	Scalar portion of quaternion (usually with subscript)
w_x, w_y:	Size of virtual image in mm
x,y,z:	Coordinates of point; subscripts or primes follow that of point name
x_e, y_e, z_e:	Shorthand coordinates for the point E
Ø:	General angular variable
α:	General variable; Also used in viewing and display error sections as angular direction from modeled eyepoint to actual eyepoint
β:	General variable
χ:	Angle of eye calibration error vector with straight-ahead direction
γ:	Angular display error
δ:	Angular viewing error
λ:	Wavelength (generally blue or red; used for lateral color analysis)
μ:	General variable (used in lateral color derivation)
η:	CRT nonlinearity as a percentage
κ:	Scaling variable (used for pure interpupillary-distance error)
θ:	Viewing direction (angle from eye to displayed point)
ρ:	Percent distortion

1. Introduction

1.1. Background

[Fuchs et al. 92] define *virtual environments* (VE) (also known as *virtual reality* or VR) as "real-time interactive graphics with three-dimensional models, when combined with a display technology that gives the user immersion in the model world and direct manipulation." A common method used for presenting the visual stimuli is to have the user wear a head-mounted display (HMD), a head-worn device which has tiny screens and optics mounted in front of the user's eyes which give a stereoscopic view of a virtual world. The images on the screens are generated by a graphics computer which uses the information from a head-tracking device (or *tracker*) to make the images on the screens correspond to the position and orientation of the user's head in the virtual world.

Augmented reality (AR) systems use the same approach as that just described, with the exception that the synthetic visual stimuli *augment* the view of the real scene rather than replace it. That is, instead of blocking the real-world view with the screens and optics, the real and synthetic views are merged with a *see-through head-mounted display* (STHMD)[1], as shown below.

[1] This research concentrates on *optical* STHMDs, which use beam-splitters to optically merge the real and virtual views. *Video* STHMDs, such as the one described in [Bajura et al. 92], acquire the real-world view via miniature video cameras and merge the real and synthetic views electronically. Many of the results of this work apply equally well to video STHMDs.

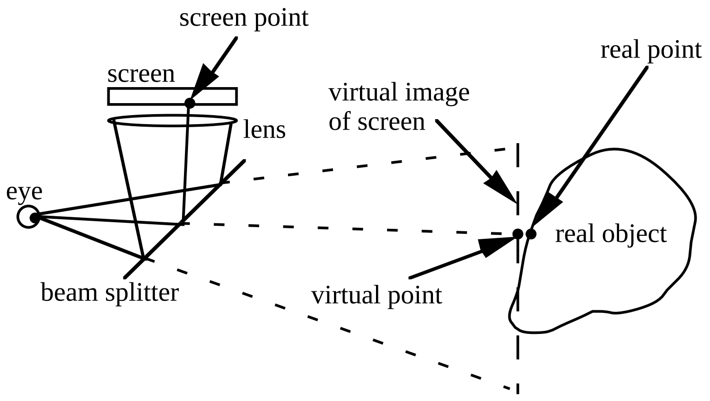


Figure 1.1.1. See-through head-mounted display

The basic operation of an STHMD system is as follows: The virtual object is rendered onto the screen by a computer graphics system, and the lens and beam splitter create an enlarged virtual image of the screen. Because the beam splitter is both transmissive and reflective, the real-world view is also visible to the user, and the net effect is that of a virtual object superimposed into the real environment. A tracker measures the position and orientation of the user's head and relays this information to the graphics system so that it can draw the virtual object in such a way that it appears to be fixed in the real environment.

The promise of AR systems is that they can augment the real-world view with additional information that will aid the user in performing a given task. For example, by superimposing 3D medical data onto the real patient, an AR system could give a surgeon "x-ray vision"; i.e., the ability to "see inside" the patient. Properly done, this superimposition should be helpful for procedures such as biopsies and surgery planning.

AR systems have existed since the mid-1960s, when Ivan Sutherland [Sutherland 68] built the first see-through head-mounted display system for displaying real-time computer-generated images. A number of AR systems have been built since Sutherland's pioneering work. The VCASS system described in [Furness 86] used an STHMD to superimpose flight and target data onto the field of view of a pilot. An STHMD system for biochemistry and architectural visualization was built at the University of North Carolina at Chapel Hill (UNC) in 1986 and is described in [Holloway 87] and [Chung et al. 89]. CAE Electronics began selling military STHMDs in 1986 for flight simulation systems [Barrette 92]. A system was developed at Boeing for airline manufacturing which superimposes wiring information onto the user's field of view [Caudell & Mizell 92]. A video STHMD system was used by [Bajura et al. 92] to merge ultrasound data with the view of the real patient's abdomen to allow the physician to "see inside" a pregnant woman. [Feiner et al. 93] describe a knowledge-based AR system which superimposes instructions onto a laser printer to aid end users in a maintenance task. Numerous non-head-mounted AR-like systems have also been built. Some use half-silvered mirrors to superimpose a view of a CRT screen into a workspace [Schmandt 83][Friets et al. 89], while others use video mixing to merge the views of the real and virtual scenes [Milgram & Drascic 91][Chesnais 85][Altobelli et al 91].

1.2. Motivation

A critical problem with AR systems is that the computer-generated objects do not currently remain correctly aligned, or *registered*, with the real environment—objects aligned from one viewpoint appear misaligned from another viewpoint and appear to swim about as the user moves her head. This is clearly unacceptable to a user seeking understanding of the relationship between the virtual and real objects, since this registration error causes the relationship to vary as the viewpoint changes. This swimming effect is also a problem with the more common, non-see-through (opaque) HMDs: even though the real-world reference is gone, it is painfully obvious with most systems that supposedly stationary virtual objects wobble as the user changes her viewpoint.

Most of the references cited in the previous section cite the correction of registration error as a critical step for making a their systems truly usable[2]. The reason that no one has solved the problem yet is that good registration demands accuracy and speed from nearly every component of the system and near-perfect system calibration. The number of error sources is large and the interactions and sensitivities of the system have not been explored in detail until now. While some progress has been made at correcting for some of the most egregious error sources (such as system delay), no previous work has completely enumerated the registration error sources and modeled their effect on the net registration error.

1.3. Research Summary

The purpose of this research is to identify the sources of registration error and analyze their effects using a mathematical model of an STHMD system. My thesis is that a mathematical error model for augmented reality systems enables the system architect to determine

[2] An exception to this is the CAE system [Barrette 92], which uses a STHMD to superimpose synthetic out-the-window imagery with the view of a physical cockpit. While this system is certainly usable, its registration requirements are generally less demanding than that of a surgical application.

1. what the registration error sources are and which ones are the most significant contributors to the total error,
2. the sensitivity of the net registration error to input errors in each part of the system,
3. the nature of the distortions caused by each type of input error, and
4. the level of registration accuracy one can expect as a function of the input errors,

and also provides insights on how to best calibrate the system.

In other words, the model tells the system architect where to spend his time and money in order to improve the system's registration, and also gives some idea of what level of registration he can expect for a given set of hardware and software.

Because certain aspects of an AR system are application-specific, I had to choose a representative application on which to base the model and the analysis. I chose craniofacial surgery planning as the target application for the following reasons:

- the real and virtual objects are at arm's length, which is representative of a large class of applications such as maintenance, surgery, and human-scale modeling/design;
- the real and virtual objects are complex and difficult to visualize well in 2D;
- the virtual dataset (the skull) is essentially rigid and immobile and therefore does not need to be scanned continuously (unlike a moving fetus);
- the main task is simply looking at the virtual objects rather than interacting with them, which means the hands do not need to be tracked and the virtual objects do not change.

The application is described in more detail in Chapter 3. Even though the model and its analysis is tailored to this application, most of the treatment applies readily to other STHMD systems.

The model breaks the registration error sources into four main categories:

1. Acquisition/alignment error: Error in acquiring the data for the virtual anatomy (*i.e.*, error in the CT scan of the skull and polygonalizing the resulting CT volume) and aligning it with the real patient in the laboratory.
2. Head-tracking error: Any error in determining the position and orientation of the STHMD worn by the user (the physician). This includes error due to delay in incorporating the most recent tracker data in the currently displayed image.

3. Viewing error: Error in the modeled location of the user's eyepoints in the computer graphics model.

4. Display error: Error within the STHMD itself. This includes optical distortion, misalignment of the virtual images, aliasing, etc.

Application of the model to a system designed for surgery planning yielded the following main results:

- System delay dominates other error sources: even for moderate head velocities, delay causes more registration error than all other sources combined. Moreoever, the net registration error is so sensitive to delay that the problem cannot be solved by simply building faster systems; systems will have to use predictive head tracking in order to achieve good registration.
- Study of the nature of viewing error revealed that using the eye's center of rotation as the eyepoint in the computer graphics model should obviate the need for eye tracking;
- Tracker error is a significant problem both in head tracking *and* in system calibration;
- The use of a World (or reference) coordinate system adds error and should be avoided when possible;
- Optical distortion causes significant registration error in the image periphery, but correcting it in the graphics pipeline often induces delay error larger than the distortion it corrects.

The contributions of this work are

- The mathematical model itself: the model is a general-purpose tool which is useful for analyzing optical STHMD systems as well as video STHMD and opaque HMD systems (with minor modifications); it provides
  - A listing of error sources;
  - Analytical equations which make explicit the relationship between the net registration error and the error sources and other system parameters;
  - A set of registration error metrics (linear, lateral, depth, and angular error);
  - The ability to model the different types of object distortions and displacements due to error of each type;
- The analysis results just cited: these apply readily to many other AR and VR systems, and give insight into the fundamental difficulties in registering the real and

virtual views, as well as giving a ranking of error sources according to how they contribute to the net registration error;

- The experimental results:
  - A test system which achieves static registration to within 1-2 mm most of the time;
  - Information on head velocities for a surgical planning application;
  - Jitter measurements for a commonly used magnetic tracker;
  - Insights into calibration methods, including the observation that errors which are hard to detect in the calibration process are typically those for which the net-error sensitivity is low;

1.4. Dissertation Overview

The rest of the dissertation is organized as follows:

- Chapter 2 describes related work in the areas of: models for VR/AR systems, models for registration error, and methods for correcting registration error.
- Chapter 3 describes the driving problem and the prototype system used as a case study for the error model.
- Chapter 4 describes the notation used and gives mathematical background for the model, including derivations of some expressions used in the model.
- Chapter 5 is the model derivation. It begins by defining the registration error metrics that will be used, then describes the four categories of input error, and finishes with derivations of analytical expressions for the error sources.
- Chapter 6 is an analysis of the prototype surgery planning system using the model from Chapter 5. System parameters for the target application and input error bounds from currently available hardware are used to determine which error sources contribute the most to the net registration error for this system. From this analysis, we determine an approximate ranking of error sources for this application.
- Chapter 7 describes the experiments done to determine error bounds for some of the input errors and experiments done to verify the model's completeness and accuracy.
- Chapter 8 examines the implications of the model and the analysis for future systems, and summarizes the results and contributions of the work.

2. Related Work

2.1. Overview

To my knowledge, there has been no previous attempt at a complete listing and quantitative analysis of error sources in VR or AR systems, although a number of papers have addressed different pieces of the problem. This chapter discusses papers that address the issue of errors in VR/AR *systems*, rather than those discussing errors in system components (such as trackers). Work in this latter category will be discussed in the corresponding section of the error model derivation or its analysis (Chapters 5 and 6).

The papers which discuss errors in VR/AR systems fall into two categories: 1) those presenting models for VR or AR systems, and 2) those describing actual systems which attempt to correct or compensate for various registration error sources. Each will be discussed in turn.

2.2. Models

[Robinett & Rolland 91] present a computational model for HMDs that applies readily to both see-through and opaque HMDs. They list a number of error sources, including incorrect modeling of the eyepoints, mismodeling the field of view, and optical distortion. They also present a model for optical distortion which can be used (and is used in Chapters 5 and 6) to model the error due to distortion.

[Hodges & Davis 93] list a number of error sources in a paper that discusses geometric considerations for VR systems. In particular, they list viewing error, head-tracking error, aliasing, optical distortion, and mismodeling the field of view as common error sources. They discuss the problem of non-antialiased displays and the effect this has on the accuracy with which a point can be placed in depth. They also characterize some of the object distortions due to viewing error due to rigid-pair translation of the eyes and the effect of eye rotation (both of which are discussed in detail in Section 6.4). This excellent discussion of geometric issues for VR systems does not attempt to present a complete model for registration error.

[Min & Jense 94] present methods for optimizing the viewing parameters in VR systems. They divide potential error sources into two categories: HMD-specific and general. The HMD-specific errors they list are failure to model optics and screen locations properly, failure to incorporate the interpupillary distance (IPD), mismodeling the field of view, optical distortion, and lateral color. The general errors they list are accommodation/ convergence errors, incorrect IPD, and incorrect view matrix parameters. They then attempt to determine the relative importance of each of these errors by allowing the user to manipulate some of the system's parameters until the images look correct. The parameters manipulated were IPD and predistortion coefficient (distortion is discussed in Section 5.7). They subjectively rank the parameters in the following order: correct perspective projection parameters, predistortion coefficient, and field of view. Some of the parameters they list as independent (*e.g.,* IPD and off-center projection offsets) are actually related, which makes it harder to interpret their results. They do not attempt to describe or quantify the effects of the errors.

2.3. Systems

[Deering 92] created a head-tracked stereo system using LCD shutter glasses which registered a real, tracked stylus with virtual objects drawn on a workstation screen. In one application, the stylus is used as a cutting tool for a virtual lathe. In order to achieve registration on the order of 10 mm, Deering modeled and corrected for most of the major error sources in a head-tracked stereo system, many of which are relevant to AR systems as well. In particular, his system accounted for the following error sources: eyepoint movement due to eye rotation, CRT faceplate refraction and curvature, head-tracking error, and system delay (compensated for by prediction). In addition to listing and correcting for these error sources, he derived input bounds for some of them; however, since his emphasis was on building a system, he concentrated on correcting them rather than modeling their effects.

[Bajura & Neumann 95] present a model for video-based AR systems which dynamically corrects the registration error by forcing alignment of the real and virtual images. Because their system uses video cameras for the real-world view, they can delay the real-world imagery to match the delay in the synthetic imagery. They also compensate for errors in the tracking data by adjusting the reported head rotation so that the real and virtual images match. The error sources they list are World-Tracker transformation error (discussed in Section 5.4), object alignment errors (Section 5.3), error in the modeled camera location/orientation (akin to viewing error), head-tracking error (Section 5.4), error in the modeled camera-to-image mapping, which includes optical distortion. As with Deering's work, the emphasis is on dynamic correction of error rather than describing or quantifying it.

[Janin et al. 93] describe methods explored at Boeing for the calibration of their optical see-through HMD. They give a thorough discussion of the parameters involved and discuss approaches taken for determining their values. These methods are discussed in more detail the analysis chapter (Chapter 6). Their list of parameters corresponds to a similar set of error sources as those listed by Bajura and Neumann. They also discuss in general terms the behavior of the net registration error as a function of tracker error.

[Azuma & Bishop 94] and [Azuma 95] describe a system for improving static and dynamic registration in AR systems. The papers begin with a description of a set of calibration procedures for achieving good initial alignment of the real and virtual objects, and then describe a system which uses inertial sensors and rate gyros to aid in predictive head tracking for maintaining the alignment as the user moves about. The chief error sources listed are optical distortion, misalignments in the HMD, head-tracking errors, incorrect viewing parameters, and system latency. The results of this work are discussed in more detail in Chapter 6.

[Rolland et al. 95] describe an AR system developed for investigating depth and size perception for real and virtual objects. They begin with a discussion of calibration techniques and issues for accurate display of virtual objects and then describe experiments done to determine how well users judge depth and size relationships between real and virtual objects using a bench-mounted STHMD. The discussion of calibration issues includes a number of important issues which are also treated in this research, including 1) which point within the eye to use as the center of projection for the computer graphics model, 2) misalignment of the display screens with respect to the optics, 3) errors in the system's field of view due to calibration error and distortion, 4) a model for error caused by mismodeling the eyepoint locations, and 5) compensating for cropping of the image on the displays' screens. These issues and this paper are discussed in more detail in later chapters.

3. The Driving Problem: Craniofacial Surgery Planning

3.1. Overview

In craniofacial surgery, bones of the skull are cut and repositioned by the surgeon in order to correct congenital or trauma-induced malformations. In order to achieve a harmonious external appearance, the relationship between the underlying bone and the soft tissue envelope must be carefully evaluated. Because of the complexity of the skull's shape and the depth variation of the surrounding tissue, the process of visualizing the relationship between the soft tissue and bone is a difficult one, requiring the surgeon to construct a mental model of the 3D anatomy through a combination of physical examination, measurement, x-rays, photographs, and CT (computed tomography) or MRI (magnetic resonance imaging) scans.

A number of other researchers have built systems to aid in visualizing 3D reconstructions of skull data and for simulating craniofacial surgical procedures [Marsh & Vannier 89][Altobelli et al. 91][Cutting et al. 86]. A natural extension of this work would be to build an AR system for visualizing the bone data superimposed on the actual patient. One might expect that visualizing the data *in situ* would be more useful and natural than viewing it on a computer screen away from the patient (although this remains to be proven, of course).

In pursuit of this goal, I collaborated with Dr. Jefferson Davis[3] to design and build such a system. However, after seeing the gross misregistration in the first prototype, I realized that the first step in making a useful system would have to be a thorough analysis of the registration error. I then used the prototype system as a test case for the AR registration error model. I will describe the prototype system next, and then will examine the implications of choosing this particular application.

[3] When this work was started, Dr. Davis was in the Department of Plastic and Reconstructive Surgery and Surgery of the Hand at UNC Hospitals; Dr. Davis has since moved to private practice in Albany, GA.

3.2. Overview of Prototype System

There are four main components to the system: the STHMD, the tracker, the graphics engine, and the host computer:

STHMD: As described in Chapter 1, the STHMD displays an image of the virtual objects superimposed into the user's field of view as shown in the following figure.

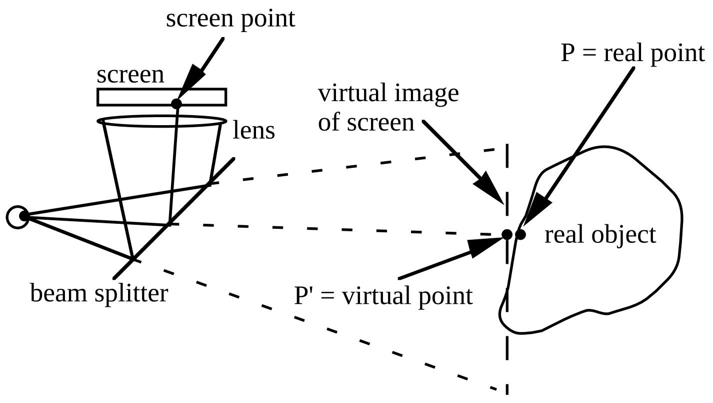


Figure 3.2.1. STHMD operation for one eye

The user sees the virtual image of the screen (created by the lens) reflected off of the beam splitter and can see the real environment as well. The next figure gives a top view showing the situation for both eyes.

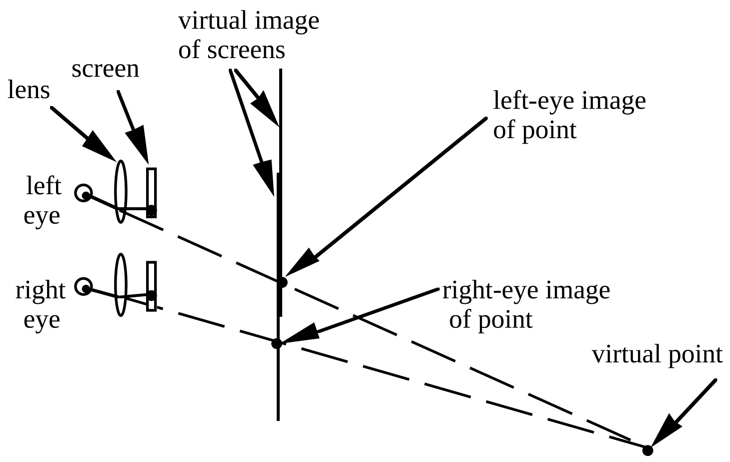


Figure 3.2.2. Top view of binocular case showing perceived point

As shown in the figure, the virtual point is defined as the intersection of the projectors that pass through the left- and right-eye image points and the left and right eyepoints.

The STHMD used in this research (pictured below) was designed so that adjustments for interpupillary distance (IPD), virtual-image distance, and positioning relative to the eyes could be done easily and repeatably. It incorporates low-resolution LCDs (approximately 340x240 primary-color elements) and simple optics (two spherical lenses and a beamsplitter for each eye). For an overview of the design, see [Holmgren 92]; [Rolland et al. 95] contains a description of the optics.

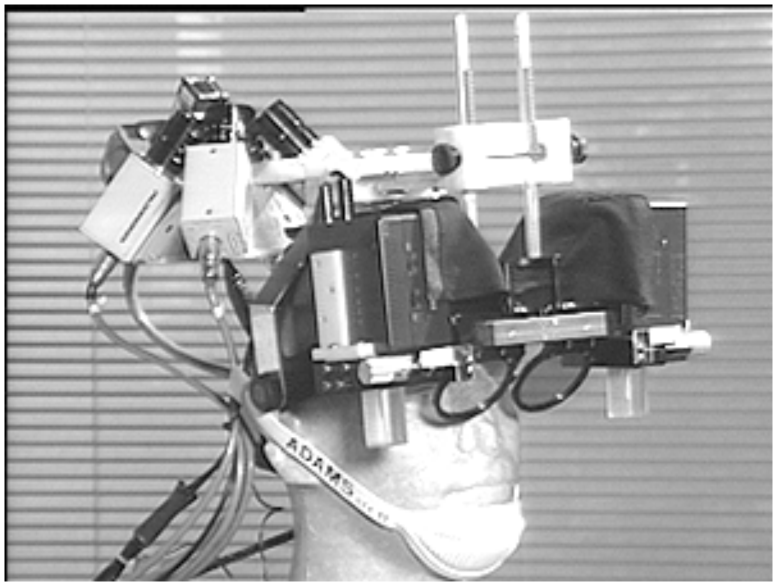


Figure 3.2.3. The current optical STHMD in use at UNC
(shown here with cameras from UNC's optical tracker mounted on the back)

Tracker: The task of the tracking subsystem is to report the position and orientation of the user's head for each display cycle. Magnetic trackers (such as the Polhemus Fastrak and the Ascension Flock of Birds) use a three-axis transmitter mounted on the ceiling and a three-axis receiver mounted on the STHMD; the transmitter pulses its three orthogonal coils in sequence and derives the induced currents in the three receiver coils in order to determine its position and orientation. For some of the experiments, a Faro mechanical tracker was used because of its higher precision (its unwieldiness makes it unsuitable for use in a real application, however).

Host computer: The host (a Sun 4) coordinates the activities of the tracker and the image-generation system. It receives position and orientation information about the user's head from the tracker, processes it and sends it on to the graphics system.

Graphics engine: The graphics engine generates images for each eye based the tracker data. The graphics engine for this system is Pixel-Planes 5 [Fuchs et al. 89], a highly parallel raster graphics engine developed at UNC. The images are rendered into two NTSC frame buffers, whose signals are passed through NTSC encoders and then on to the displays in the STHMD. Pixel-Planes 5 can currently render over two million Phong-shaded triangles per second, which makes it well suited for displaying complex medical data sets.

A typical session with this system would be as follows:

- The patient's head is scanned using a CT scanner. The resulting data is converted to polygons expressed in scanner coordinates (this process is described in more detail in Chapter 5). This set of polygons defines the virtual object to be displayed.
- The patient is seated in the examination room and his head is immobilized (with a frame or bite bar).
- Corresponding landmark points on the skull dataset and on the real patient are digitized. The skull dataset landmarks are identified and measured at a workstation, and the real-patient landmarks are digitized using a 3D stylus or tracker.
- An alignment algorithm is run to determine the transformation required for a least-squares alignment of the skull dataset with the real patient in the examining room.
- The surgeon then dons the STHMD and runs through a calibration procedure to make sure it is correctly positioned with respect to her eyes.
- The surgeon then examines the patient using the AR system, and sees the 3D skull dataset superimposed on the real patient. She can make measurements, judge soft-tissue depths, and use the system to create a surgical plan based on the augmented planning session.

More details on each of these steps will be given in subsequent chapters.

3.3. Effect of the Application on the Error Model

The choice of this application has a number of implications for the error model and analysis; in particular, it determines

- the level of registration required. Dr. Davis estimates that for the system to be useful, it should superimpose the virtual bone within a millimeter of the real skull.
- the viewing distance, which is an important parameter in the mathematical model of the STHMD. Dr. Davis estimated his average working distance to be about 500 mm,

and an experiment (described in Chapter 7) showed that another surgeon’s working distance ranged from 275 to 600 mm, with a mean of 430 mm.

- the nature and size of the errors in the virtual dataset, in this case, the polygonalized skull. These errors are discussed in Section 5.3.
- the amount of error in the real landmark points. Because the real skull (i.e., inside the patient) is covered with soft tissue, we can only get an approximate measurement of the real landmarks.
- the expected head velocities for the surgeon wearing the STHMD. As we shall see, system delay is a major error source, and the net registration error is a function of both the delay and the velocity of the surgeon’s head as he examines the patient.
- the field of view for the STHMD. Because we know the range of object sizes (human heads) and working distances, we can determine the field of view required for the application. In this case, if we use the 95th percentile for head height from [Woodson & Conover 64] as the maximum head dimension (259 mm), the angle subtended ranges from 24˚ to 50˚ for the range of working distances cited above.

The next chapter gives background material for the error model and Chapter 5 presents the model itself.

4. Mathematical Background

"Quaternions came from Hamilton... and have been an unmixed evil to those who have touched them in any way. Vector is a useless survival... and has never been of the slightest use to any creature." – Lord Kelvin

This chapter begins by discussing the notation used in the rest of this dissertation, then discusses the mathematical background material for the derivations in this and the following chapter, and ends with a number of derivations that will be used in the next chapter.

4.1. Notational Conventions

- The terms "coordinate system" (CS) and "space" will be used interchangeably;[4] the name of the CS will be capitalized: Tracker space, World CS.
- Points and CSs will be denoted with capital letters. Points' and CSs' modeled or measured locations/orientations will be marked with a prime ('). Points and CSs whose locations/orientations are uncertain due to propagated error (e.g., calculated using a concatenation of transformations each of which contains measurement or modeling error) will be marked with a double-prime ("). The real point on the patient is denoted by P and the final, displayed point is denoted by P*. (Section 5.2.2 contains a complete list of all points and CSs used.)
- Transformations: $T_{A'\_B''}$ will denote a transformation from coordinate system B" to coordinate system A'. R_{A_B} will be used in some cases when the transformation is a pure rotation.

[4] Mathematically, the term "space" is more general and absolute, while "coordinate system" is relative and limited (e.g., three-space contains all possible 3D coordinate systems). However, for brevity, I will use the broader term "space" to denote a Euclidean 3-space in which all points are expressed within the named orthonormal 3D coordinate system.

- Vectors will be denoted by a boldface, lower-case letter (usually **v**) with subscripts indicating the vector's starting and ending points. When the first subscript denotes a coordinate system (rather than a simple point), the vector is understood to be expressed within that CS (i.e., with its units[5] and relative to its orientation). When the first subscript is the name of a point, the text will indicate which CS's orientation will be used. Examples: $\mathbf{v}_{A\_P}$ is a vector in A that goes from A's origin to the point P; $\mathbf{v}_{P\_P'}$ goes from P to P' and may be expressed with the orientation of any CS.

- Transformations of vectors from one coordinate system to another will use the notation $\mathbf{v}_{A\_P} = T_{A\_B} \cdot \mathbf{v}_{B\_P}$; i.e., to express where a point P is in the A coordinate system given its location in the B coordinate system, we apply the transform T_{A_B}. Note how the subscripts cancel; this property will be used extensively in what follows.

- Because a point's position may be expressed relative to any coordinate system and because there are so many coordinate systems, I will use vectors to indicate a point's location relative to each coordinate system. (A common alternative is to add a subscript to the point's name indicating which coordinate system it has been expressed in, but this is problematic when using figures of multiple CSs, since a single point in the figure may have several names.) Thus, multiple vectors may be used to describe the location of a point P relative to different coordinate systems: $\mathbf{v}_{A\_P}$, $\mathbf{v}_{B\_P}$, etc. Accordingly, only these vectors will be transformed rather than the points themselves, in order to make it clear which coordinate systems and points are involved in each transformation.

- Quaternions[6] will be represented in the format $\mathbf{q}_{A\_B}$, which gives the rotation of coordinate system B relative to A. (This also follows the convention used for transformations.)

- The variable *b* will be used to represent the bound on the magnitude of the registration error from a given error source; that is, $b_{P*\_P} \geq \|\mathbf{v}_{P*\_P}\|$. Subscripts will be used to make it clear which error type *b* is a bound for.

- The pure error in a quantity will be denoted by a preceding "δ", following [Pizer 75]. Thus the error in the transformation T will be denoted δT; the error in the angle Ø is denoted δØ, etc.

5 The units used for all coordinate systems in this research will be millimeters (mm).

6 Appendix A contains a brief introduction to quaternions.

- The "·" operator will be used to represent the *composition* or *concatenation* of two transformations.

4.2. General Discussion of Generated and Propagated Error

This section describes and gives brief derivations for the error methods used in this research. It is not intended to be a complete listing nor a complete description of these methods. The interested reader may find more details in [Pizer 75] and [Pizer and Wallace 83].

For most of this work, we are concerned with bounding the error in a 2D or 3D vector. This vector typically is from the correct point P to the final, erroneous point P^*. The error vector may be written relative to any coordinate system; for example,

$$\mathbf{v}_{P\_P*} = \mathbf{v}_{A\_P*} - \mathbf{v}_{A\_P} \tag{4.2.1}$$

expresses the error vector from P to P* as a difference of two vectors in the A coordinate system.

In a discussion of misregistration, we typically are most concerned with the distance between the real landmarks and their virtual counterparts. Thus, it is the magnitude rather than the direction of the error vector which is of most interest. In addition, we are more likely to have a bound on the error vector magnitude for a tracker or some other piece of hardware than to have a bound on each component of the error vector.

Because we are interested in the length of vectors, we will use the *Euclidean norm* for describing both vectors and any matrices which are used as operators on them. For a vector, the Euclidean or l_2 norm is denoted by $||\mathbf{v}||_2$ and is defined as

$$||\mathbf{v}||_2 = \sqrt{x^2 + y^2 + z^2} \tag{4.2.2}$$

which is just the vector's length in three-space. The error bounds describing certain components of the system (such as the tracker's translational error) will often be given as a bound on the l_2 norm of the error vector.

In the system under consideration, there are also numerous operators which transform vectors in the system. We must also characterize the effect of these operators on the errors in the vectors which are transformed by them. In particular, we want to know how error vectors are magnified by an operator T. (In all that follows, we will assume that the operators are linear). The operator's magnification is described by the *operator norm* $||T||$, defined as

$$\|T\| = \max \frac{\text{output error size}}{\text{input error size}} = \max_{\delta\mathbf{v}\neq 0} \frac{\|T(\delta\mathbf{v})\|}{\|\delta\mathbf{v}\|} \tag{4.2.3}$$

The operator norm depends on two vector norms which might in principle be different, but in all that follows we will use the l_2 norm in both the numerator and the denominator because of its geometrical significance for this application. For a real matrix A, the l_2 norm is

$$\|A\|_2 = \sqrt{\text{max eigenvalue of } A^TA} \tag{4.2.4}$$

where A^T is the transpose of A.

Some properties of vector and matrix norms (from [Pizer 75]) that will prove to be useful are presented without proof:

$$\|\mathbf{u} + \mathbf{v}\| \le \|\mathbf{u}\| + \|\mathbf{v}\| \quad \text{(triangle inequality)} \tag{4.2.5}$$

$$\|T \cdot \mathbf{v}\| \le \|T\| \cdot \|\mathbf{v}\| \tag{4.2.6}$$

$$\|T + S\| \le \|T\| + \|S\| \tag{4.2.7}$$

$$\|T \cdot S\| \le \|T\| \cdot \|S\| \tag{4.2.8}$$

For $\mathbf{u} = T(\mathbf{v})$, the magnitude of the output error vector is bounded using $\|T\|$ and the magnitude of the input error vector:

$$\|\delta\mathbf{u}\| \le \|T\| \cdot \|\delta\mathbf{v}\| \tag{4.2.9}$$

4.3. Derivation of Some Common Expressions

This section contains derivations of operator norms used later in the error model derivation. We begin with two-dimensional transformations and then work up to three-dimensional transforms.

4.3.1. Norms Involving Rotation Matrices

4.3.1.1. The Norm of a Rotation Matrix

The Euclidean norm for a matrix R is just the square root of the maximum eigenvalue of R^TR. For rotation matrices, $R^{-1} = R^T$. Thus,

$$R^TR = I, \tag{4.3.1.1}$$

$$\|R\| = \|R^{-1}\| = 1 \tag{4.3.1.2}$$

and the eigenvalues are all equal to 1. This makes geometric sense, since a rotation matrix does none of the “stretching” discussed previously.

4.3.1.2. The Norm of R - I in 2D

Another common situation involving rotation matrices is when there is some error in the angle of rotation. Given the correctly rotated vector

$$\mathbf{u} = R \cdot \mathbf{v} \tag{4.3.1.2}$$

we have the incorrectly rotated vector

$$\mathbf{u}'' = R' \cdot \mathbf{v} \tag{4.3.1.3}$$

We can view this operation as a combination of the desired rotation, R, followed by a pure-error rotation, δR:

$$\mathbf{u}'' = R' \cdot \mathbf{v} = \delta R \cdot R \cdot \mathbf{v} = \delta R \cdot \mathbf{u} \tag{4.3.1.4}$$

Thus, we can express the error vector δ**u** in terms of this formulation:

$$\delta\mathbf{u} = R' \cdot \mathbf{v} - R \cdot \mathbf{v} = (\delta R - I) \cdot R \cdot \mathbf{v} \tag{4.3.1.5}$$

Taking the norm of this equation and using the results of the previous section yields

$$\|\delta\mathbf{u}\| = \|\delta R - I\| \cdot \|R\| \cdot \|\mathbf{v}\| \tag{4.3.1.6}$$

Since ||R|| = 1, we are interested in the norm of δR - I. In 2D, we can characterize δR with a single angle, δØ, and express δR - I as

$$\delta R - I = \begin{bmatrix} \cos\delta Ø - 1 & -\sin\delta Ø \\ \sin\delta Ø & \cos\delta Ø - 1 \end{bmatrix} \tag{4.3.1.7}$$

As described previously, the norm is the square root of the largest eigenvalue of $(\delta R - I)^T \cdot (\delta R - I)$:

$$(\delta R - I)^T \cdot (\delta R - I) = \begin{bmatrix} (\cos\delta Ø-1)^2+\sin^2\delta Ø & 0 \\ 0 & (\cos\delta Ø-1)^2+\sin^2\delta Ø \end{bmatrix} \tag{4.3.1.8}$$

The diagonals of this matrix *are* its eigenvalues, which simplify to $4\cdot\sin^2\left(\frac{\delta Ø}{2}\right)$, and the norm is thus

$$\|\delta R - I\| = \left|2\cdot\sin\frac{\delta Ø}{2}\right| \tag{4.3.1.9}$$

and the error vector bound is

$$\|\delta\mathbf{u}\| = \left|2\cdot\sin\frac{\delta Ø}{2}\right| \cdot \|\mathbf{v}\| \tag{4.3.1.10}$$

This, too, makes geometric sense— the length of a vector between two vectors of length ||**v**|| with an angle δØ between them is exactly $\left|2\cdot\sin\frac{\delta Ø}{2}\right|\cdot\|\mathbf{v}\|$. Note that this formula is valid for all angles, not just when δØ is small. In fact, when δØ is small, $2\cdot\sin\frac{\delta Ø}{2} \approx \delta Ø$, as expected. Since Equation 4.3.1.10 applies for all angles, we can apply this result for any operator of the form R - I.

4.3.1.3. The Norm of R - I in 3D

In three dimensions, the expression for a general rotation matrix does not lend itself as readily to the eigenvalue approach used above. However, we can look at the geometry and use a convenient coordinate system in order to derive the norm of R - I in 3D.

As before, we can express the error in a vector that is rotated incorrectly in terms of δR - I:

$$\delta\mathbf{u} = R' \cdot \mathbf{v} - R \cdot \mathbf{v} = (\delta R - I) \cdot R \cdot \mathbf{v} \qquad (4.3.1.11)$$

Since we are interested in the effects of the error caused by δR, we can define

$$\mathbf{w} = R \cdot \mathbf{v} \qquad (4.3.1.12)$$

to simplify the analysis. This does not affect the magnitude of δ**u** because ||R|| = 1. Thus, we now have

$$\delta\mathbf{u} = (\delta R - I) \cdot \mathbf{w} \qquad (4.3.1.13)$$

Without loss of generality, we can choose our coordinate system so that the rotation by δØ is about the Z axis (although **w** is not constrained to the X-Y plane), and therefore δR has the form

$$\delta R = \begin{bmatrix} \cos\delta Ø & -\sin\delta Ø & 0 \\ \sin\delta Ø & \cos\delta Ø & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.3.1.14)$$

It is easy to show that δR - I when formulated in this way has the same norm as shown above:

$$(\delta R - I)^T \cdot (\delta R - I) = \begin{bmatrix} (\cos\delta Ø-1)^2+\sin^2\delta Ø & 0 & 0 \\ 0 & (\cos\delta Ø-1)^2+\sin^2\delta Ø & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad (4.3.1.15)$$

which has the same eigenvalues as shown above, so

$$||\delta R - I|| = \left| 2 \cdot \sin\frac{\delta Ø}{2} \right| \qquad (4.3.1.16)$$

That the 3D and 2D cases should have the same norm comes from Euler's result that any sequence of rotations in 3D can be described by a single rotation about some axis. This final rotation rotates only the components of a vector that are in the plane of rotation (i.e., the plane perpendicular to the axis of rotation). Vectors lying in this plane are rotated by δØ, since this is the same as 2D rotation; vectors lying on the axis of rotation are not rotated at all, since their projection onto the plane of rotation is the null vector. Thus, the maximum value of $||\delta\mathbf{u}||$ corresponds to rotation of a vector in the plane of rotation of δR which is identical to the 2D case; any departure by the vector from the plane of rotation results in less rotation and therefore smaller values of $||\delta\mathbf{u}||$.

4.3.2. Bounding Angles for Compositions of Rotations

In the error-model analysis, we will often encounter rotations (represented as matrices or quaternions) that are themselves products of two or more rotations. In such cases, we will be interested in bounding or calculating the net rotation performed by the composite rotation matrix or quaternion. In the following subsections I will derive expressions describing the net angle of rotation in two relevant cases.

4.3.2.1. Bounding the Angle for Two Rotations

When a vector **v** is rotated first by a rotation matrix R_1 and then by a rotation matrix R_2, the resultant vector is given by

$$\mathbf{u} = R_2 \cdot R_1 \cdot \mathbf{v} = R_T \cdot \mathbf{v} \qquad (4.3.2.1)$$

In this case, the angle between **u** and **v** will determine the magnitude of the vector **u - v** which goes from **v** to **u**. Thus, if we can bound the net rotation performed by R_T, we can bound the magnitude of the difference vector **u - v**.

In two dimensions, it is trivial to show that a rotation by $Ø_1$ followed by a rotation by $Ø_2$ is equal to a rotation by $Ø_1 + Ø_2$; i.e., the angles just add algebraically. The situation in 3D is more complex since the axes of rotation are not generally coincident, in which case the planes of rotation do not coincide. For this reason, I will use a different representation for rotations in order to bound the net rotation performed by this composition of two rotations.

As explained in Appendix A, we can express Equation 4.3.2.1 in terms of unit quaternions as

$$\mathbf{u} = \mathbf{q}_2 \cdot \mathbf{q}_1 \cdot \mathbf{v} \cdot \mathbf{q}_1^{-1} \cdot \mathbf{q}_2^{-1} = \mathbf{q}_T \cdot \mathbf{v} \cdot \mathbf{q}_T^{-1} \qquad (4.3.2.2)$$

Unit quaternions represent a rotation as a vector (**n**) describing the axis of rotation and a scalar (w) that specifies the amount of rotation about that axis:

$$\mathbf{q} = [w, \mathbf{n}], \qquad \text{with } w = \cos\frac{Ø}{2}, \ \ ||\mathbf{n}|| = \sin\frac{Ø}{2} \qquad (4.3.2.3)$$

We can exploit this representation in order to bound the angle on the product of two quaternions (and therefore of any two rotations).

Expressing $\mathbf{q}_T$ in terms of $\mathbf{q}_1$ and $\mathbf{q}_2$ (as explained in Appendix A), we have

$$\mathbf{q}_T = \mathbf{q}_1 \cdot \mathbf{q}_2 = [(w_1 w_2 - \mathbf{n}_1 \cdot \mathbf{n}_2),\ (w_1 \cdot \mathbf{n}_2 + w_2 \cdot \mathbf{n}_1 + \mathbf{n}_1 \times \mathbf{n}_2)] \qquad (4.3.2.4)$$

Since the product of two unit quaternions is also a unit quaternion [Shoemake 89], we can infer the angle of the resulting quaternion $\mathbf{q}_T$ from the following relation:

$$w_T = \cos\frac{Ø_T}{2} = w_1 w_2 - \mathbf{n}_1 \cdot \mathbf{n}_2 \qquad (4.3.2.5)$$

Using the definition in Equation 4.3.2.3 and the definition of the vector dot product, we can write Equation 4.3.2.5 as follows:

$$w_T = \cos\frac{Ø_T}{2} = \cos\frac{Ø_1}{2} \cdot \cos\frac{Ø_2}{2} - \left|\sin\frac{Ø_1}{2}\right| \cdot \left|\sin\frac{Ø_2}{2}\right| \cdot \cos Ø_{12} \qquad (4.3.2.6)$$

where $Ø_{12}$ is the angle between the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$. Solving for $Ø_T$ yields

$$Ø_T = 2\cos^{-1}\left[\cos\frac{Ø_1}{2} \cdot \cos\frac{Ø_2}{2} - \left|\sin\frac{Ø_1}{2}\right| \cdot \left|\sin\frac{Ø_2}{2}\right| \cdot \cos Ø_{12}\right] \qquad (4.3.2.7)$$

We can then find the extreme values of $Ø_T$ by differentiating with respect to $Ø_{12}$ (since $Ø_1$ and $Ø_2$ are assumed to be given and are therefore constant):

$$\frac{dØ_T}{dØ_{12}} = \frac{-\left|\sin\frac{Ø_1}{2}\right| \cdot \left|\sin\frac{Ø_2}{2}\right| \cdot \sin Ø_{12}}{\sqrt{1 - \left[\cos\frac{Ø_1}{2} \cdot \cos\frac{Ø_2}{2} - \left|\sin\frac{Ø_1}{2}\right| \cdot \left|\sin\frac{Ø_2}{2}\right| \cdot \cos Ø_{12}\right]^2}} \qquad (4.3.2.8)$$

Setting this derivative to zero and solving for $Ø_{12}$ shows that $Ø_T$ takes on extreme values for $Ø_{12}$ equal to 0˚ and 180˚ (corresponding to $\sin Ø_{12} = 0$), which means that the extreme values for the composition of two rotations occur when the planes of rotation of $\mathbf{q}_1$ and $\mathbf{q}_2$ are coincident. Depending on the directions of rotation within the plane, the absolute values of two angles will either add or subtract, as in the 2D case. If $Ø_1$ and $Ø_2$ are each between 0˚ and 180˚, then the sum of their absolute values gives the largest net rotation and

$$Ø_T \le |Ø_1| + |Ø_2| \qquad (4.3.2.9)$$

Otherwise, we can use the more general expression

$$Ø_T \le \max(|Ø_1| \pm |Ø_2|) \qquad \textit{(modulo } 2\pi\textit{)} \qquad (4.3.2.10)$$

In the derivations that follow I will use the simpler expression (Equation 4.3.2.9) and assume (without loss of generality) that all angles are expressed within the range 0˚ - 180˚.

4.3.2.2. Bounding the Angle for the Case $\mathbf{R_T = R_1 \cdot R_2 \cdot R_1^{-1}}$

In the analysis that follows, we will often encounter a set of rotations of the form

$$R_T = R_1 \cdot R_2 \cdot R_1^{-1} \qquad (4.3.2.11)$$

Transformations of this form are generally known as *similarity transformations.* As in the previous subsection, our goal is to bound the net rotation of R_T given the magnitude of the rotations for R_1 and R_2. As before, we can use the properties of quaternions to derive the result we seek. In terms of quaternions, we seek the angle specified by the scalar part of the unit quaternion $\mathbf{q}_T$ where

$$\mathbf{q}_T = \mathbf{q}_1 \cdot \mathbf{q}_2 \cdot \mathbf{q}_1^{-1} \qquad (4.3.2.12)$$

Because $\mathbf{q}_1$ and $\mathbf{q}_2$ are unit quaternions, $\mathbf{q}_T$ will also be a unit quaternion since the product of two unit quaternions is itself a unit quaternion. Thus, if we calculate the scalar portion of $\mathbf{q}_T$ in terms of $\mathbf{q}_1$ and $\mathbf{q}_2$, we should easily be able to infer the net angle of rotation.

From the definitions in Appendix A, we have

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = [(w_1\, w_2 - \mathbf{n}_1 \cdot \mathbf{n}_2),\ (w_1 \cdot \mathbf{n}_2 + w_2 \cdot \mathbf{n}_1 + \mathbf{n}_1 \times \mathbf{n}_2)] = [w_{12}, \mathbf{n}_{12}] \qquad (4.3.2.13)$$

and

$$\mathbf{q}_T = \mathbf{q}_1 \cdot \mathbf{q}_2 \cdot \mathbf{q}_1^{-1} =$$
$$[(w_{12}\, w_1 - \mathbf{n}_{12} \cdot (-\mathbf{n}_1)),\ (w_{12} \cdot (-\mathbf{n}_1) + w_1 \cdot \mathbf{n}_{12} + \mathbf{n}_{12} \times (-\mathbf{n}_1))] \qquad (4.3.2.14)$$

Expanding the scalar portion of the result, w_T, in terms of known quantities yields

$$w_T = (w_1 \cdot w_2 - \mathbf{n}_1 \cdot \mathbf{n}_2) \cdot w_1 - (w_1\, \mathbf{n}_2 + w_2\, \mathbf{n}_1 + \mathbf{n}_1 \times \mathbf{n}_2) \cdot (-\mathbf{n}_1) \qquad (4.3.2.15)$$

which reduces to

$$w_T = w_1^2 \cdot w_2 - w_1 \mathbf{n}_1 \cdot \mathbf{n}_2 + w_1 \mathbf{n}_1 \cdot \mathbf{n}_2 + w_2 \mathbf{n}_1 \cdot \mathbf{n}_1 + (\mathbf{n}_1 \times \mathbf{n}_2) \cdot \mathbf{n}_1 \qquad (4.3.2.16)$$

The last term is zero by the definition of cross and dot products, and the second and third terms cancel to yield

$$w_T = w_2 \cdot (w_1^2 + \|\mathbf{n}_1\|^2) \qquad (4.3.2.17)$$

which reduces to

$$w_T = \cos\frac{\emptyset_2}{2} \cdot \left(\cos^2\frac{\emptyset_1}{2} + \sin^2\frac{\emptyset_1}{2}\right) = \cos\frac{\emptyset_2}{2} \qquad (4.3.2.18)$$

Thus, the net angle of rotation of the sequence $\mathbf{q}_1 \cdot \mathbf{q}_2 \cdot \mathbf{q}_1^{-1}$ is equal to the angle specified by the inner rotation, $\mathbf{q}_2$. Therefore, any time we need to bound the net angle of rotation of a series of rotations of the form $R_1 \cdot R_2 \cdot R_1^{-1}$ we can use the angular bound for R_2 in its place. Note that in contrast to the previous derivation, the net angular term here is an equality, rather than an upper bound.

4.4. Bounding Error for Rigid-Body Transformations

Rigid-body transformations consist of only rotations and translations, and therefore do not alter the shape of the transformed objects, hence the term “rigid-body”. Since the STHMD system performs several such transformations at each frame and since these transformations are based on measurements which contain error, we are interested in the way errors are propagated by compositions of such transforms. The first section will treat the errors in the transformations themselves, and the second section will examine the effects of these errors on a transformed vector.

4.4.1. Error in Rigid-Body Transforms

The basic question is, given a transformation T_{A_B} and a transformation T_{B_C}, both with some amount of rotational and translational error, what is the error in the composition of these two transformations? In the analysis that follows, we will assume that each of the transformations is measured independently of the others[7]; that is, each measurement is unaffected by errors made in the other measurements. Given this assumption, the analysis must focus on how the errors made in one transformation are propagated by subsequent transformations. Analyzing the error propagation in the composition of two transforms will give us a general result that can be applied to multiple compositions, since the error in the first transform can be treated as the total error of some prior composition of transforms, and the error in the result can be then used as an input to a subsequent composition. In this way, we can form a result for an arbitrary sequence of rigid-body transforms, assuming only that we have bounds on the error in the angle of rotation and the translational error-vector magnitude.

Using the nomenclature described earlier in this chapter, we can write the composition as

$$T_{A\_C} = T_{A\_B} \cdot T_{B\_C} \tag{4.4.1.1}$$

The figure below depicts the ideal case: A point P’s location in the C coordinate system is given by the vector $\mathbf{v}_{C\_P}$. To express the location of P in the A coordinate system, we transform it using the composition above:

$$\mathbf{v}_{A\_P} = T_{A\_B} \cdot T_{B\_C} \cdot \mathbf{v}_{C\_P} \tag{4.4.1.2}$$

[7] This is the case for the STHMD system: For example, errors made in measuring the HMD relative to the tracker do not affect the error in measuring the locations of the screen images of the HMD.

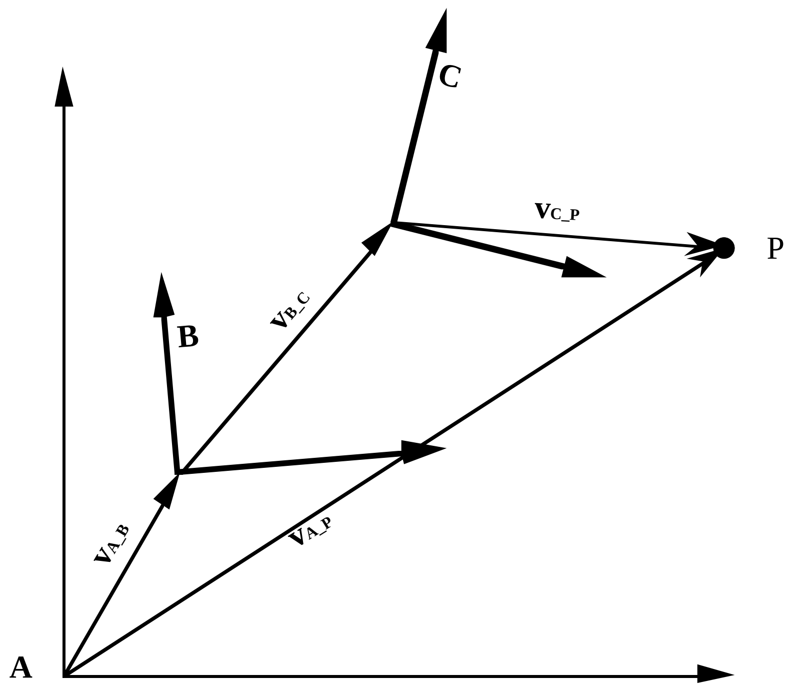


Figure 4.4.1.1. Ideal case for composition of two transforms

However, if there are errors in T_{A_B}, then B’s position and orientation with respect to A will be known only approximately. Similarly, the errors in T_{B_C} will give only an approximate notion of C’s relationship to B. Moreover, computing C’s relationship to A will involve the errors in both transformations. Geometrically, we can think of this in the following way (see the figure below): We measure B’s position and orientation with respect to A to yield B', the measured B coordinate system. This gives $T_{A\_B'}$. Assuming that the measurement of T_{B_C} is made independently of this measurement of $T_{A\_B'}$ and is therefore made relative to the true coordinate system B, we will have another measured transform, $T_{B\_C'}$. Because we do not know where the true B coordinate system is, this result is applied at B' because that is the coordinate system expressed by $T_{A\_B'}$. Thus, the two measured transformations (indicated as dashed-line vectors in the figure) are composed to yield the apparent coordinate system C", which is different from both C and C' due to the propagated error from $T_{A\_B'}$. Note that the true C coordinate system is disconnected, because its location and orientation are not known exactly.

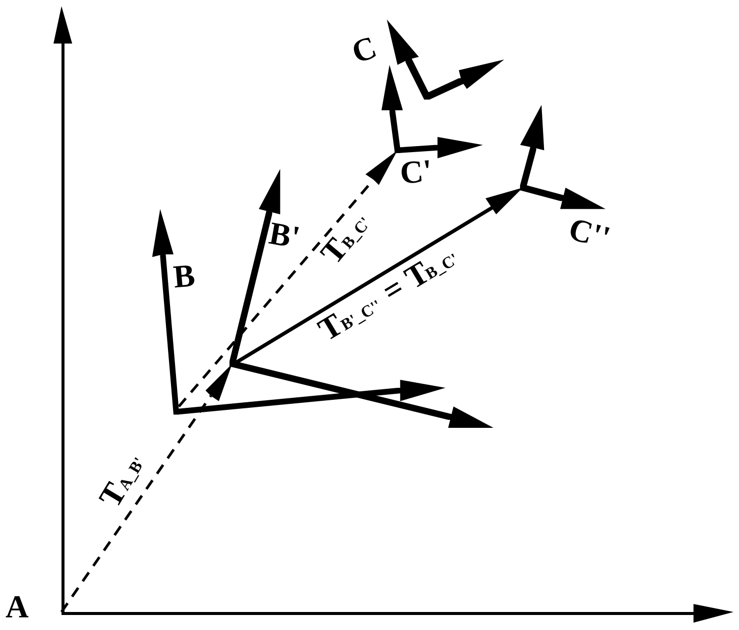


Figure 4.4.1.2. Composition of transformations with errors

Note that the transformation $T_{B'\_C''}$ in the figure is just $T_{B\_C'}$ applied in the B' coordinate system and the two are thus identical.

Symbolically, we have

$$T_{A\_C''} = T_{A\_B'} \cdot T_{B'\_C''} \tag{4.4.1.3}$$

Because of the equalities listed above this is equivalent to

$$T_{A\_C''} = T_{A\_B'} \cdot T_{B\_C'} \tag{4.4.1.4}$$

(Note that in this case the subscripts no longer cancel exactly.)

The transformation which represents the error in the system is denoted by $T_{C''\_C}$ (or its inverse), which represents where the true C coordinate system is with respect to the erroneous CS, C''. Expressing this transformation in terms of known transforms, we have

$$T_{C''\_C} = T_{C''\_B'} \cdot T_{B'\_B} \cdot T_{B\_C'} \cdot T_{C'\_C} \tag{4.4.1.5}$$

By expanding this expression, we can derive a bound on the angular and translational errors in the resulting transform in terms of known error bounds on the measured transforms. Since $T_{C''\_C}$ is a rigid-body transform, it can be written as the combination of a 3x3 rotation matrix $R_{C''\_C}$ and a vector $\mathbf{v}_{C''\_C}$ acting on a vector $\mathbf{v}$ as follows:

$$T_{C''\_C} \cdot \mathbf{v} = R_{C''\_C} \cdot \mathbf{v} + \mathbf{v}_{C''\_C} \tag{4.4.1.6}$$

That is, the vector **v** is first rotated by $R_{C''\_C}$ and then translated by the vector addition with $\mathbf{v}_{C''\_C}$.

Expanding as before yields

$$\begin{aligned} T_{C''\_C} \cdot \mathbf{v} = \; & R_{C''\_B'} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot R_{C'\_C} \cdot \mathbf{v} \; + \\ & R_{C''\_B'} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot \mathbf{v}_{C'\_C} \; + \\ & R_{C''\_B'} \cdot R_{B'\_B} \cdot \mathbf{v}_{B\_C'} \; + \\ & R_{C''\_B'} \cdot \mathbf{v}_{B'\_B} \; + \\ & \mathbf{v}_{C''\_B'} \end{aligned} \tag{4.4.1.7}$$

The first line clearly represents the rotation component of the composite transformation, while the last four lines of the equation represent the various translation components. Our goal is to bound the angle of rotation represented by the first line and the magnitude of the translation vector represented by the last four terms.

Because of the equivalence of $R_{C''\_B'}$ and $R_{C'\_B}$, the rotational term reduces to

$$R_{C''\_C} = R_{C'\_B} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot R_{C'\_C} \tag{4.4.1.8}$$

The first three rotations constitute the similarity rotation discussed in Section 4.3.2.2, so that the angle represented by the transformation $R_{C'\_B} \cdot R_{B'\_B} \cdot R_{B\_C'}$ is just that of the inner rotation, which we will call $Ø_{B'\_B}$. What remains, then, is the composition of this rotation with the rotation $R_{C'\_C}$, whose angle is represented by $Ø_{C'\_C}$. But in Section 4.3.2.1 we showed that the upper bound on the composition of two rotations is just the sum of their absolute values (for angles between 0˚ and 180˚). Thus, the angular error bound for the transformation $T_{C''\_C}$ is just

$$|Ø_{C''\_C}| \leq |Ø_{B'\_B}| + |Ø_{C'\_C}| \tag{4.4.1.9}$$

In other words, the angular errors just add in the worst case, which is to be expected since this is the behavior in two dimensions.

The bound on the translational error is somewhat more complex. The exact expression for the translational error is given by the last four lines of Equation 4.4.1.7:

$$\begin{aligned} \mathbf{v}_{C''\_C} = \; & R_{C'\_B} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot \mathbf{v}_{C'\_C} \; + \\ & R_{C'\_B} \cdot \mathbf{v}_{B'\_B} \; + \\ & R_{C'\_B} \cdot R_{B'\_B} \cdot \mathbf{v}_{B\_C'} \; + \; \mathbf{v}_{C'\_B} \end{aligned} \tag{4.4.1.10}$$

(where the ordering of the terms has been changed slightly and $R_{C'\_B}$ and $\mathbf{v}_{C'\_B}$ have been substituted for $R_{C''\_B'}$ and $\mathbf{v}_{C''\_B'}$). We can simplify the last line by expressing $\mathbf{v}_{C'\_B}$ in terms of $\mathbf{v}_{B\_C'}$. To do so, note that

$$\mathbf{v}_{C'\_B} = -(R_{C'\_B} \cdot \mathbf{v}_{B\_C'}) \qquad (4.4.1.11)$$

i.e., if we take the vector from B to C', rotate it into the C' coordinate system, and then invert it, it goes from C' to B as before. The last line of 4.4.1.10 then becomes

$$R_{C'\_B} \cdot R_{B'\_B} \cdot \mathbf{v}_{B\_C'} - R_{C'\_B} \cdot \mathbf{v}_{B\_C'} \qquad (4.4.1.12)$$

which can be simplified to yield

$$R_{C'\_B} \cdot (R_{B'\_B} - I) \cdot \mathbf{v}_{B\_C'} \qquad (4.4.1.13)$$

We have

$$\begin{aligned}\mathbf{v}_{C''\_C} = \; & R_{C'\_B} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot \mathbf{v}_{C'\_C} + \\ & R_{C'\_B} \cdot \mathbf{v}_{B'\_B} + \\ & R_{C'\_B} \cdot (R_{B'\_B} - I) \cdot \mathbf{v}_{B\_C'}\end{aligned} \qquad (4.4.1.14)$$

The error bound is then

$$\begin{aligned}\|\mathbf{v}_{C''\_C}\| = \; & \|R_{C'\_B} \cdot R_{B'\_B} \cdot R_{B\_C'} \cdot \mathbf{v}_{C'\_C} + \\ & R_{C'\_B} \cdot \mathbf{v}_{B'\_B} + \\ & R_{C'\_B} \cdot (R_{B'\_B} - I) \cdot \mathbf{v}_{B\_C'}\|\end{aligned} \qquad (4.4.1.15)$$

From the triangle inequality (Equation 4.3c), we know that the sum of the norms is greater than or equal to the norm of the sum:

$$\begin{aligned}\|\mathbf{v}_{C''\_C}\| \leq \; & \|R_{C''\_B'} \cdot R_{B'\_B} \cdot R_{B\_C'}\| \cdot \|\mathbf{v}_{C'\_C}\| + \\ & \|R_{C''\_B'}\| \cdot \|\mathbf{v}_{B'\_B}\| + \\ & \|R_{C'\_B} \cdot (R_{B'\_B} - I)\| \cdot \|\mathbf{v}_{B\_C'}\|\end{aligned} \qquad (4.4.1.16)$$

Moreover, if we assume that the orientation of the error vectors $\mathbf{v}_{B'\_B}$ and $\mathbf{v}_{C'\_C}$ in this equation are uncorrelated and that the vector $\mathbf{v}_{B\_C'}$ is also uncorrelated with the error vectors, then we can do no better than the above expression; i.e., it is not overly pessimistic since we have no reason to suspect that the vectors will consistently cancel each other out to any degree.

From Section 4.3.1.1 we know that the norm of a rotation matrix (or a combination of rotation matrices, since the combination is itself a rotation matrix) is equal to 1, and from the calculation of the norm of (R - I) in Section 4.3.1.3, we can simplify the error bound to

$$\|\mathbf{v}_{C''\_C}\| \leq \|\mathbf{v}_{C'\_C}\| + \|\mathbf{v}_{B'\_B}\| + \left|2\cdot\sin\frac{\text{Ø}_{B'\_B}}{2}\right| \cdot \|\mathbf{v}_{B\_C'}\| \qquad (4.4.1.17)$$

This is the net translational error; recall that the net rotational error is given by

$$|\text{Ø}_{C''\_C}| \leq |\text{Ø}_{B'\_B}| + |\text{Ø}_{C'\_C}| \qquad (4.4.1.9)$$

Thus, the rotational errors just add (in the worst case), and the translational error is just the sum of the translational errors plus a term involving prior angular errors and the measured distance between the B and C coordinate systems.

Because we assumed the existence of errors in the transformation leading up to this one, we can generalize the result for multiple transformations. A transformation from C to some other coordinate system D would have a rotational error of

$$|\text{Ø}_{D''\_D}| \leq |\text{Ø}_{B'\_B}| + |\text{Ø}_{C'\_C}| + |\text{Ø}_{D'\_D}| \qquad (4.4.1.18)$$

and a translational error of

$$\|\mathbf{v}_{D''\_D}\| \leq \|\mathbf{v}_{B'\_B}\| + \|\mathbf{v}_{C'\_C}\| + \left|2\cdot\sin\frac{\text{Ø}_{B'\_B}}{2}\right| \cdot \|\mathbf{v}_{B\_C'}\| + \|\mathbf{v}_{D'\_D}\| + \left|2\cdot\sin\frac{\text{Ø}_{B'\_B}+\text{Ø}_{C'\_C}}{2}\right| \cdot \|\mathbf{v}_{C\_D'}\| \qquad (4.4.1.19)$$

Note that the translational errors calculated for the previous transform just add in (including the term with $\text{Ø}_{B'\_B}$) with the translation error from this transform, plus a new term involving all of the prior angular errors and the translation between the last two CSs involved. The general formulation for N transformations (labelled 1..N) between *N+1* coordinate systems (labelled 0..N) is

Rotation:

$$|\text{Ø}_{N''\_N}| \leq \sum_{i=1}^{N} |\text{Ø}_{i'\_i}| \qquad (4.4.1.20)$$

Translation:

$$\|\mathbf{v}_{N''\_N}\| \leq \sum_{i=1}^{N} \|\mathbf{v}_{i'\_i}\| + \sum_{i=1}^{N-1} 2\cdot\left|\sin\left(\frac{\sum_{j=1}^{i}\text{Ø}_{j'\_j}}{2}\right)\right| \cdot \|\mathbf{v}_{i\_i+1'}\| \qquad (4.4.1.21)$$

This equation assumes that the 0th coordinate system's location and orientation are known exactly by definition; if this is not true, we can always define an additional CS for which this is true and increment N accordingly.

The uses and implications of this result will be discussed in the next chapter.

4.4.2. Error in Transforming a Vector

If we transform a vector which contains error by a transformation which contains error, how much error does the resulting vector contain? I.e., given a transformation $T_{A\_B'}$ between A and B' (the measured location/orientation of B) and a vector $\mathbf{v}_{B\_P'}$ (from the actual B coordinate system to the measured point P'), how we can derive a vector which expresses the location of P' in the A coordinate system? Because the transformation contains error, the resulting vector from A will not go to P' but will end up at P'', as shown in the figure below.

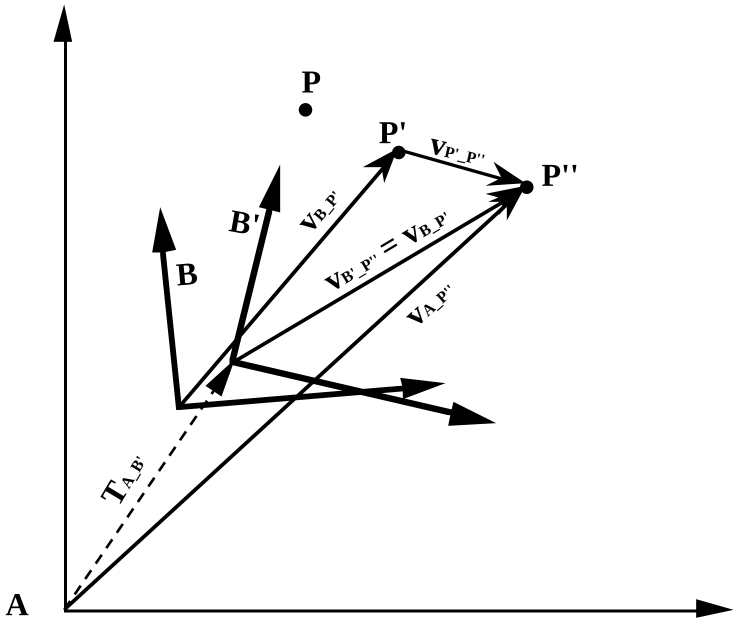


Figure 4.4.2.1. Expressing a point in two coordinate systems

That is, we measure P relative to the real B coordinate system to yield the measured vector $\mathbf{v}_{B\_P'}$; we transform this measured vector by the measured transformation $T_{A\_B'}$ to yield the vector $\mathbf{v}_{A\_P''}$:

$$\mathbf{v}_{A\_P''} = T_{A\_B'} \cdot \mathbf{v}_{B\_P'} = T_{A\_B'} \cdot \mathbf{v}_{B'\_P''} \tag{4.4.2.1}$$

Note the similarity to the derivation in the previous section. In fact, we can bound the magnitude of the error vector $\mathbf{v}_{P\_P''}$ by treating P as the origin of yet another coordinate system whose orientation is not of interest. That is, we have already treated the case above where we consider the error in the transformation

$$T_{A\_C''} = T_{A\_B'} \cdot T_{B\_C'} = T_{A\_B'} \cdot T_{B'\_C''} \qquad (4.4.2.2)$$

If we substitute P for C, P' for C', and P" for C", let $R_{B'\_C''} = I$ and $\mathbf{v}_{B'\_C''} = \mathbf{v}_{B'\_P''}$ and examine the translational error, we get

$$\|\mathbf{v}_{P''\_P}\| \leq \|\mathbf{v}_{P'\_P}\| + \|\mathbf{v}_{B'\_B}\| + 2\cdot\left|\sin\frac{\varnothing_{B'\_B}}{2}\right|\cdot\|\mathbf{v}_{B\_P'}\| \qquad (4.4.2.3)$$

That is, the error vector between P and P" is bounded by the translational error in measuring P initially, plus the translational error in the transformation $T_{A\_B'}$, plus the rotation term seen for transformation compositions seen earlier.

Finally, having given this bound on $\mathbf{v}_{P\_P''}$, it is important to show that it is not overly pessimistic; i.e., that this expression is an equality in the worst case. Assuming that the two translational error terms and the rotational errors are uncorrelated, then the following figure demonstrates the worst-case behavior.

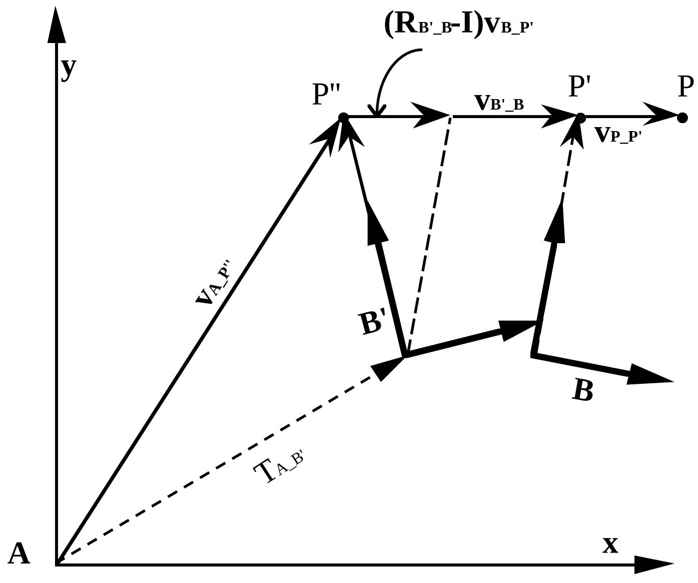


Figure 4.4.2.2. Worst-case behavior for transforming a vector

In this figure, the initial measurement error vector $\mathbf{v}_{P'\_P}$ is in the +X direction, as are the translational and rotational errors due to $T_{A\_B'}$. The total error vector magnitude is just the sum of the magnitudes of each of these vectors. Thus, although the worst-case behavior may occur infrequently, it can occur.

4.5. Computer Graphics Background

This section covers some of the mathematics used in rendering the images displayed in the see-through head-mounted display (STHMD). We will first discuss how the perspective projection is done and will then move on to a treatment of how two projected points are combined by the eyes into a 3D point.

4.5.1. Planar Perspective Projection

As shown in Chapter 4, the STHMD creates an illusion of a point in three space by drawing the left- and right-eye projections of the point onto the left and right screens, whose images are viewed through the STHMD optics. These screen images are therefore the projection planes onto which we draw the two-dimensional left- and right-eye views. In this section, we will examine the mathematics of the planar perspective projection that converts a 3D description of a point into a 2D projection.

The figure below shows the situation for the projection.

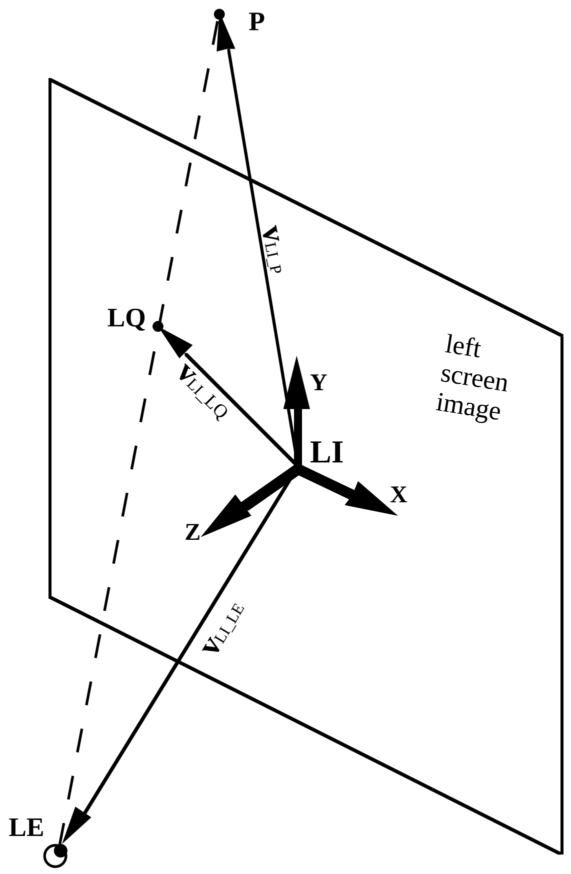


Figure 4.5.1.1. Projection of point onto plane

Here we use the Left Image CS (LI) as the reference CS and project the point P onto the X-Y plane of LI at LQ, based on the location of the left eyepoint LE. The origin of the LI coordinate system is the center of the left screen's virtual image and the orientation is X = right, Y = up, Z = toward the eye (the right screen is analogous). The vectors to each point are indicated in the figure. Note that this is different from the usual coordinate system used in computer graphics, where the origin is at the eye point and the screen is defined relative to the eye. This coordinate system was chosen so that modeling eye movement could be done in a natural way; i.e., when the eye moves relative to the screen image, the origin does not change.

We will use the following shorthand notation for the components of the vectors to make the equations that follow more readable: $\mathbf{v}_{LI\_P} = (x, y, z)$, $\mathbf{v}_{LI\_LQ} = (x_{LQ}, y_{LQ}, z_{LQ})$, $\mathbf{v}_{LI\_LE} = (x_{LE}, y_{LE}, z_{LE})$ (equations for the right eye and screen will use "R" instead of "L").

If we represent the vector $\mathbf{v}_{LI\_P}$ in homogenous coordinates, the projection operation can be represented in matrix form as follows[8]:

$$T_{Proj} \cdot \mathbf{v}_{LI\_P} = \mathbf{v}_{LI\_LQ} = \begin{bmatrix} 1 & 0 & -\frac{x_{LE}}{z_{LE}} & 0 \\ 0 & 1 & -\frac{y_{LE}}{z_{LE}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{z_{LE}} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x - \frac{x_{LE}\, z}{z_{LE}} \\ y - \frac{y_{LE}\, z}{z_{LE}} \\ 0 \\ -\frac{z - z_{LE}}{z_{LE}} \end{bmatrix} \tag{4.5.1.1}$$

After dividing by the last coordinate, we have

$$\mathbf{v}_{LI\_LQ} = \begin{bmatrix} \frac{x\, z_{LE} - x_{LE}\, z}{z_{LE} - z} \\ \frac{y\, z_{LE} - y_{LE}\, z}{z_{LE} - z} \\ 0 \\ 1 \end{bmatrix} \tag{4.5.1.2}$$

This vector expresses the 2D location of the projected point LQ in Left Image space. The expression for the right eye projected point is analogous. The units for this vector are still the same as those used in World space.

To convert this description of the point into device coordinates, we perform a 2D scaling to convert from World-space units to pixels. In matrix form, we have

$$T_{LD\_LI} \cdot \mathbf{v}_{LI\_LQ} = \begin{bmatrix} \frac{p_x}{w_x} & 0 & 0 & 0 \\ 0 & \frac{p_y}{w_y} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{x\, z_{LE} - x_{LE}\, z}{z_{LE} - z} \\ \frac{y\, z_{LE} - y_{LE}\, z}{z_{LE} - z} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{p_x}{w_x} \cdot \left[\frac{x\, z_{LE} - x_{LE}\, z}{z_{LE} - z} \right] \\ \frac{p_y}{w_y} \cdot \left[\frac{y\, z_{LE} - y_{LE}\, z}{z_{LE} - z} \right] \\ 0 \\ 1 \end{bmatrix} \tag{4.5.1.3}$$

where p_x and p_y are the size of the frame buffer image in pixels. For simplicity, I assume that the center of the frame buffer is at (0, 0) and that the center frame buffer pixel maps to the center of the display screen.

8 The derivation for this matrix is given in [Foley et al. 90], p. 256-58.

4.5.2. Combining Two Projected Points into a 3D Point

Now that we have derived an expression for the planar projections of a 3D point for the left eye and right eye, we turn to the problem of how the eyes combine the two images of the point to arrive back at a percept of a 3D point.

As we saw in Figure 3.2.2, the perceived point is at the intersection of the projectors from each eye through the projected points. The following figure gives a view on the situation from above (with the +Y axis out of the page).

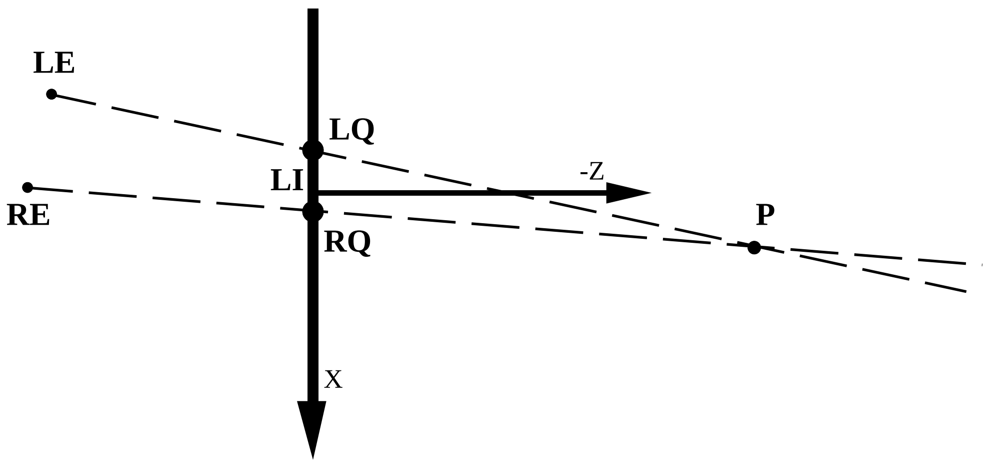


Figure 4.5.2.1. Combining projected points

In the figure, the Left-Image CS is still the reference CS; both eyepoints and projected points are expressed in this CS. In what follows, I will make the following assumptions:

- The eyes can be modeled as single points, corresponding to the pinhole camera model. (This is not to say that the eyes are modeled as immobile relative to the screens, but only that we can treat some point within the eye as the pinhole of the pinhole camera.)
- The left and right projection planes (the screen images) are coplanar. (When the projection planes deviate from this model, we will project the points in the actual planes back onto the modeled planes and treat the difference as an error in LQ or RQ. See Section 5.7 for more on this.)
- The left and right projectors are coplanar. This guarantees that the projectors will intersect except when the virtual point is at infinity. To be sure, it is possible to have projectors which are skew and therefore do not intersect, but I assume for this

analysis that this does not occur. This assumption seems reasonable, since a system displaying images which are vertically misaligned will be very hard to use and will typically have to be adjusted until the images are vertically aligned.

Given these assumptions, we can now analyze the geometry in the X-Z plane to determine the intersection point's X and Z coordinates as a function of the eye and projected-point positions. These results will then be used to derive the Y coordinate of the 3D point.

From the previous section, we have

$$x_{LQ} = \frac{x\, z_{LE} - x_{LE}\, z}{z_{LE} - z} \tag{4.5.2.1}$$

and

$$x_{RQ} = \frac{x\, z_{RE} - x_{RE}\, z}{z_{RE} - z} \tag{4.5.2.2}$$

If we solve each for *z,* set the two equations equal, and solve for *x*, we get

$$x = \frac{x_{RQ}\, z_{RE}\,(x_{LQ} - x_{LE}) - x_{LQ}\, z_{LE}\,(x_{RQ} - x_{RE})}{z_{RE}\,(x_{LQ} - x_{LE}) - z_{LE}\,(x_{RQ} - x_{RE})} \tag{4.5.2.3}$$

Via a similar process, we find *z* to be

$$z = \frac{z_{LE}\, z_{RE}\,(x_{LQ} - x_{RQ})}{z_{RE}\,(x_{LQ} - x_{LE}) - z_{LE}\,(x_{RQ} - x_{RE})} \tag{4.5.2.4}$$

To find the Y coordinate of the perceived point, we begin with the expression for the projected Y coordinate for the left eye derived in the previous section:

$$y_{LQ} = \frac{y\, z_{LE} + y_{LE}\, z}{z_{LE} - z} \tag{4.5.2.5}$$

If we solve for *y* and use Equation 4.5.2.4 for *z*, we get

$$y = y_{LQ} + \frac{z_{RE}\,(x_{LQ} - x_{RQ})(y_{LE} - y_{LQ})}{z_{RE}\,(x_{LQ} - x_{LE}) - z_{LE}\,(x_{RQ} - x_{RE})} \tag{4.5.2.6}$$

We can use a similar derivation for the right eye to express y in terms of y_{RQ} and y_{RE}.

Note that in the simple case where $z_{LE} = z_{RE} = d$, $x_{LE} = 0$, and $x_{RE} = IPD$ (corresponding to the line joining the eyes parallel to the the projection plane with the left eye on the Z axis, and the right eye a distance IPD to the right), these equations reduce to

$$x = \frac{IPD\, x_{LQ}}{x_{LQ} - (x_{RQ} - IPD)} \tag{4.5.2.7}$$

$$y = \frac{IPD\, y_{LQ}}{x_{LQ} - (x_{RQ} - IPD)} \tag{4.5.2.8}$$

$$z = \frac{d\,(x_{LQ} - x_{RQ})}{x_{LQ} - (x_{RQ} - IPD)} \qquad (4.5.2.9)$$

5. Error Model Derivation

This chapter explains the derivation of the error model for the x-ray vision system. The bulk of the chapter is a step-by-step analysis of each transformation, the errors generated, and the errors propagated by that transformation. This analysis will use a fairly theoretical, top-down approach, with fairly abstract models for each component of the system. In order to keep the line of reasoning clear, this chapter will not discuss the implications of the error expressions derived here in any depth. Chapter 6 will analyze the main expressions derived in this chapter in greater detail.

5.1. Registration Error Metrics

Before discussing the model for predicting the registration error for a STHMD system, we should examine some different metrics for registration error. If two points that are supposed to be coincident are separated by some distance, then one can describe the degree of separation or misregistration with a 3D error vector from one point to the other. *Linear registration error* is defined here to be the magnitude, or length, of this error vector. This is often a useful metric for registration error, since it gives a concrete measure of how far a virtual point may deviate in any direction from its real counterpart. On the other hand, the analysis of errors associated with stereoscopic displays introduces a number of problems with using only the linear registration error as a metric.

The first problem is caused by the nature of stereoscopic displays. With such displays, it is quite possible to have small input errors that induce theoretically huge output errors. To illustrate this, consider the two cases in the figure below.

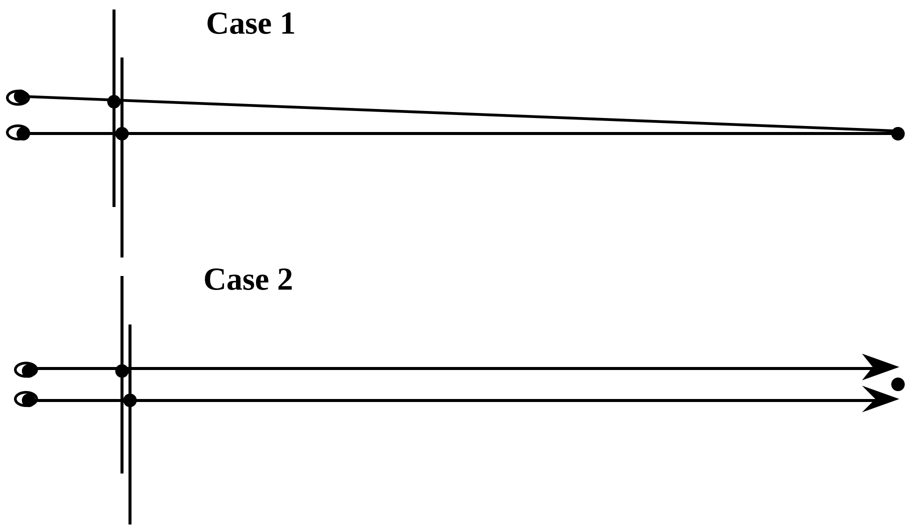


Figure 5.1.1. Small changes in IPD $\rightarrow$ large changes in depth

In the figure, a small change in interpupillary distance (IPD) from Case 1 to Case 2 caused the projectors from the eyes through the projected points to become parallel, thus inducing an infinite depth error, since the virtual point is theoretically infinitely distant. A numerical example shows how easily this can happen: A STHMD system with a screen image distance of 0.5 meters attempting to display an on-axis point at a depth of 7.5 meters for an IPD of 64 millimeters will be perceived as infinitely distant when viewed by someone with an IPD of 60 millimeters (both IPDs are well within the 95th percentile according to [Woodson & Conover 64]).

To characterize this situation as having infinite registration error seems overly pessimistic and not very useful, since the projectors do pass near the point and may appear to converge at the point when coupled with other depth cues, such as head-motion parallax[9]. Another problem is that, in theory, to find the linear registration error we must analyze the situation for both eyes before we will know the point's depth, and therefore its location. But certainly we can close one eye and still make some judgment of how bad the alignment is, even if we cannot judge the depth error very well.

---

[9] [Reading 83] notes, "under circumstances in which the absolute depth cue conflicts with binocular disparity information, not only does the stereopsis depth match shift, but also it suffers a loss of sensitivity" (p. 289). Thus, the presence of other, correctly rendered depth cues may well serve to minimize the error due to incorrect stereopsis and convergence information.

All of this leads to the conclusion that registration errors in depth are somehow different from registration errors that cause a clear visual separation between the real and virtual points, particularly when a stereoscopic display is involved. For this reason, I propose to use an additional set of metrics for registration error. The metrics are based on a decomposition of the linear registration error metric into *lateral error* and *depth error,* as shown below.

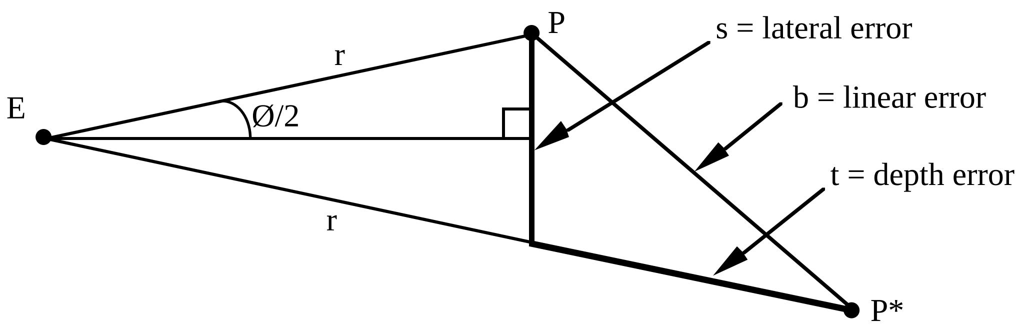


Figure 5.1.2. Lateral/depth registration error

Lateral error is the segment (with length *s*) which is normal to the line of sight (for a given eye), and depth error is the segment (of length *t*) aligned with the line of sight. Note that the lateral and depth components are not orthogonal, so we cannot use the familiar Pythagorean theorem for relating the lateral and depth errors to the linear error (except as an approximation when Ø is small). If the angle between the lateral and depth segments can be derived for a situation, we can use the law of cosines to describe the relationship of the three segments' lengths.

The specification of the lateral error leads to yet another, related metric: the *angular registration error*. This is just the visual angle subtended by the lateral error (and the segment PP*) at a given eyepoint, and will prove quite useful for characterizing certain types of errors. Clearly, all of these measures depend on the geometry defining the segment PP*, so a complete specification will depend on the situation being discussed.

If we know the vectors from the eye to P and P*, the size of the depth error is just the absolute value of the difference of the vector magnitudes:

$$t = \left| \|\mathbf{v}_{E\_P*}\| - \|\mathbf{v}_{E\_P}\| \right| \tag{5.1.1}$$

If we have derived the angle Ø for a given situation, then the relationship between the size of the lateral error and the angular error is given by

$$\sin\frac{Ø}{2} = \frac{s}{2r} \qquad (5.1.2)$$

In cases where we want to convert from linear to angular error, we will often assume that the real and virtual points are equidistant from the eye so that the lateral error is a maximum (and depth error is zero), leading to

$$s = 2\, r \sin\frac{Ø}{2} \qquad (5.1.3)$$

In summary, then, the linear registration error metric can be decomposed into lateral and depth components, which separates the visually important lateral error from the less perceivable and more error-prone depth error. Angular error can be used to describe the error as a visual angle, which will be useful in some cases. Moreover, we may use the angular error as a measure of the linear error in cases where we are prepared to either ignore depth error or assume that it is zero. I will use this decomposition for viewing and display errors, and the linear registration error for the analysis of rigid-body transformations (whose errors are typically bounded and are best expressed in Cartesian coordinates).

5.2. STHMD System Model Overview

5.2.1. Coordinate Systems for STHMD System

A STHMD system necessarily contains a number of different coordinate systems (CSs), reflecting the spatial locations and orientations of the various system components. As we will see, the errors in the system come from inaccurate modeling of the transformations between these coordinate systems. We therefore begin with a discussion of the various coordinate systems in the STHMD system and their relationships to each other.

As detailed in [Robinett & Holloway 92], the coordinate systems used in HMD software can be represented with a *coordinate system graph.* The graph for the ideal, error-free[10] see-through HMD system analyzed herein is given in the figure below.

[10] The next section addresses the issue of errors in these transforms; in this section I present only the ideal, error-free versions of the transforms and CSs.

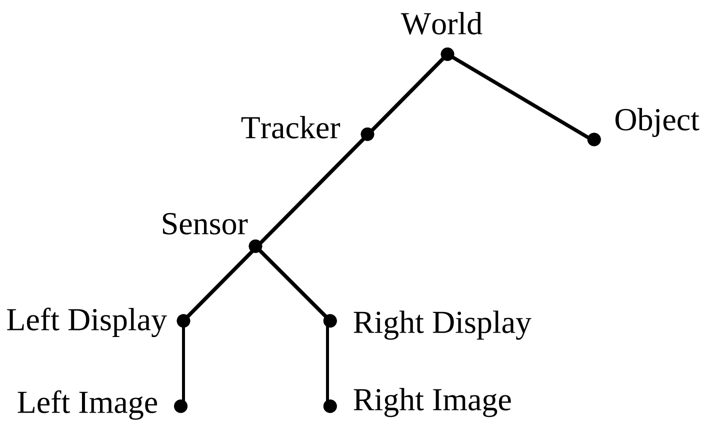


Figure 5.2.1.1. Coordinate systems for a HMD system

The nodes in the graph represent coordinate systems and the edges represent the transformations between the CSs.

The coordinate systems are defined as follows:

Object: Each object in the virtual environment can have its own user-defined CS in which its primitives are defined. For example, the CT scanner used in our surgery planning application reports the scanned anatomy in terms of its own coordinate system (the virtual anatomy must then be transformed to align it with the real anatomy in World space, as described in Section 5.3.4).

World: This is sometimes also called the Laboratory CS and is essentially the real-world reference CS, and can be defined to be anywhere within the laboratory that is convenient. Our World CS has the origin on the floor in the center of the tracker's working volume (for convenience), with orientation: X = East, Y = North, Z = up.

Tracker: This is the CS in which the tracker reports the position and orientation of the Sensor that is mounted on the STHMD. The origin is typically in the center of the transmitter (for magnetic trackers); the orientation varies according to how the transmitter is mounted.

Sensor: The origin of this CS is the origin of the tracker sensor, which is mounted on the STHMD. The orientation is also determined by how the sensor is mounted.

Left, Right Display Device: These are the coordinate systems for the display devices themselves (i.e., the LCDs or CRTs). The position and orientation of the devices is only relevant insofar as it affects the images of the devices. The orientation is the same as the Image CSs, with the origin in the middle of the screen[11].

Left, Right Image: The origin of the Image CS is defined to be the center of the virtual image of the screen; the orientation is defined to be: X = right, Y = up, Z = toward the user. The left and right eyepoints are defined via vectors expressed in the corresponding Image CS.

The following list describes the rigid-body transformations between each CS and where their values come from:

T_{W_O} = World-Object transform: Gives the position and orientation of the virtual anatomy (*e.g.*, from a CT scan) in the World CS. This transform is the output of the alignment phase described in Section 5.3.4.

$T_{W\_T\_}$ = World-Tracker transform: Gives the position and orientation of the Tracker CS within the World. Measured.

T_{T_S} = Tracker-Sensor transform: This is the output of the tracker; gives the position and orientation of the tracker's sensor (which is mounted on the STHMD) relative to the Tracker CS.

T_{S_LI}, T_{S_RI} = Sensor-{Left,Right}-Image transform: Gives position and orientation of left or right screen image relative to Sensor CS. Derived via a calibration procedure.

[This list contains only the rigid-body transformations between the CSs listed above; the non-rigid-body transforms will be discussed later.]

At this point, it should be mentioned that the coordinate systems used here differ from those used in conventional computer graphics (such as is described in [Foley et al. 90]) and that derived in [Robinett & Holloway 92] due to the need here to reflect the physical characteristics of a real STHMD system, which are often abstracted away in most systems (for good reason). In particular, because we must treat the eyes as movable objects, they are no longer well suited as coordinate-system origins and will be treated simply as points representing the centers of projection (real and modeled). The traditional Eye CS has been replaced by the Image CS, which will be used to handle both projected and unprojected

---

[11] This is another departure from traditional computer graphics, but an origin in the center of the screen will simplify and clarify the derivations that follow.

points. In addition, the treatment of head movement relative to the STHMD itself forces us to abandon the Head CS described in [Robinett & Holloway 92] in favor of the comparatively stable Sensor CS.

A transform between any two CSs can be derived by composing the transforms between the two nodes. For example, the transform that takes a vector in World space and expresses it relative to Left-Image space is derived as follows:

$$T_{LI\_W} = T_{LI\_S} \cdot T_{S\_T} \cdot T_{T\_W} \qquad (5.2.1.1)$$

(Note how the subscripts cancel in the product to yield the final subscript.[12])

The registration errors in the x-ray vision are caused by errors in the transformations which are applied to the landmark points as they go through the system. The goal of this chapter is to determine the theoretical error bound for the displayed points as a function of the individual errors made in each stage of the pipeline.

5.2.2. Partitioning the Problem

The system has so many parameters that some high-level organization is essential. In particular, it makes sense to group together those transformations which are interdependent or similar and to treat them together, and to separate those parts of the problem which are independent of each other.

Toward this end, I will partition the problem as follows:

1. The tasks of imaging the anatomy with the CT scanner, polygonalizing the resulting dataset, and aligning that set of polygons in the World CS will all be handled together. Although these tasks are quite different in nature, they can all be viewed as part of the task of getting the virtual object into the system, which is quite separate from the tasks associated with displaying and viewing it. Errors made in any of these tasks will be referred to as errors in locating the virtual point in the World CS. In other words, if we imagine a particular landmark point on the patient, the goal of this stage is to image and polygonalize the anatomy defining it and then align that point and its neighbors with their real-anatomy counterparts in the World CS. Thus, failures in this phase of the pipeline will cause the virtual landmark to deviate from the real-world landmark independently of subsequent operations. I will refer to this category as *acquisition/alignment error*.

[12] We will see later that when errors are introduced, the subscripts no longer cancel exactly.

2. The STHMD will be treated as a black box whose position and orientation are specified via the Sensor CS. The World-Tracker and Tracker-Sensor transformations are rigid-body transformations associated with head tracking and will be treated together under the category of *head-tracking error*.

3. Within the STHMD, the errors will be divided into two types: viewing error and display error. *Viewing error* is the deviation of the eyepoints from their modeled locations. *Display error* is anything that goes wrong in the rendering of the images into the frame buffer, the scanout of those images onto the display devices, and the formation of the virtual images of the display devices by the optics. The image coordinate systems (LI and RI) are within the black box. Even though their transformations relative to the Sensor CS are rigid-body transformations, errors in them induce non-rigid-body errors and will therefore be treated with the other display errors.

The figure below depicts these components graphically.

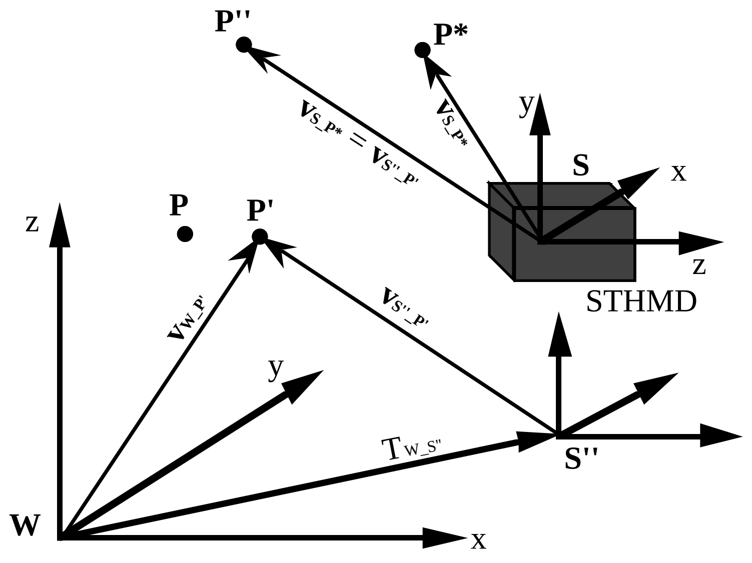


Figure 5.2.2.1. Black-box model of STHMD system

In the figure, the point P represents the landmark on the real patient, which is where we would like the apparent point P* to appear. P' is the CT-imaged/polygonalized/aligned point from Step 1 above, which will in general deviate from P. The vector $\mathbf{v}_{W\_P'}$ gives the location of P' in World space. The arrow labeled $T_{W\_S''}$ is the composite transformation described in Step 2 and represents the composition of the rigid-body transformations $T_{W\_T'}$ and $T_{T\_S'}$ which yield the transform between World space (W) and apparent Sensor space

(S") (this type of error was discussed in detail in Section 4.4.1). Due to errors in the component transforms, S" and S (the actual Sensor CS) do not coincide. Finally, the black box with the Sensor CS embedded in it represents the STHMD as described in Step 3. Its task is to create two projected images that, when viewed by the user's two eyes, give a perception of a point exactly at the location specified by the input vector $\mathbf{v}_{S''\_P'}$. Because of the errors in $T_{W\_S''}$, the best the display can hope to do is to display the point at P", but because of display and viewing errors, the final perceived point ends up at P^* instead.

In summary, then, there are three general categories of error, corresponding to the three steps listed previously:

1. Acquisition/alignment error in locating the point P in W, a process which yields the point P'.
2. Tracking error in measuring the transformations between World space and Sensor space, a process which yields the transformation $T_{W\_S''}$, which is then inverted and used to express P' in S" ($\mathbf{v}_{S''\_P'}$). When this vector is applied in the actual S coordinate system, the result is the point P".
3. Display/viewing error in the point whose location is specified by $\mathbf{v}_{S''\_P'}$ due to errors in modeling the eyepoints and errors within the display itself, which yields the final apparent point P^*.

The partitioning just described will be reflected in the following sections; that is, the next section will discuss errors made in locating P in World space, Section 5.4 will analyze the rigid-body transformation errors associated with locating and orienting the Sensor CS relative to World space, and Sections 5.5-7 will examine errors made within the black box representing the STHMD. This analysis essentially traces a virtual point's path through the system and analyzes the error generated and propagated at each step from beginning to end.

5.3. Acquisition/Alignment Errors

5.3.1. Introduction

This section discusses the error sources encountered in entering the virtual anatomy data into the system, apart from all the problems one encounters in trying to correctly view that dataset. The three principal operations are the imaging of the real anatomy with some type of medical imaging device (such as a CT scanner), polygonalization of the resulting dataset, and the alignment of the polygonal model within the World CS. An overview of this process is shown in the following figure.

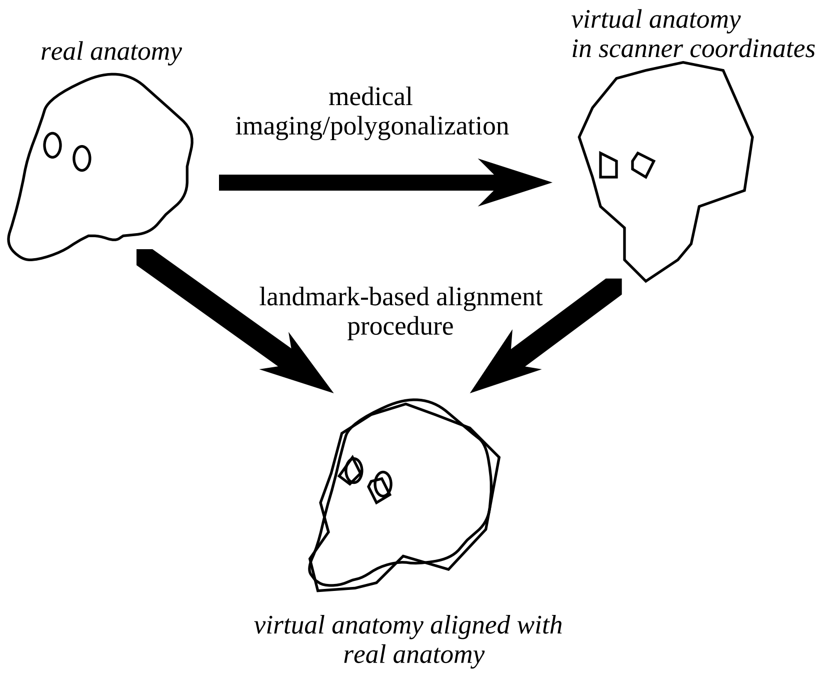

real anatomy
virtual anatomy in scanner coordinates
medical imaging/polygonalization
landmark-based alignment procedure
virtual anatomy aligned with real anatomy

Figure 5.3.1.1. Overview of acquisition/alignment process

In the figure, the real anatomy is imaged and then polygonalized to yield the virtual anatomy (or virtual dataset) in CT scanner coordinates. The virtual dataset is then aligned with the real anatomy in World space with a landmark-based alignment procedure.

The three processes of medical imaging, polygonalization, and alignment are interrelated and are often treated together in the literature. The error sources in these processes are so numerous and complex that a complete model describing them is beyond the scope of this thesis. Moreover, any such model would be very dependent on the medical imaging technology used, the method used for visualizing the resulting dataset (e.g., surface vs. volume rendering), and the alignment algorithm used. For all these reasons, these errors will be treated together as a black box for the error model derived here. This section will describe the procedures and the errors associated with them, and Section 6.2 will give an idea of what numerical bounds are reasonable for this part of the pipeline via simulation of the alignment process and citation of results from the literature.

5.3.2. Computed Tomography (CT)

Computed tomography, or CT, is a non-invasive imaging technique that allows surgeons to "see" inside the patient. The basic idea is to pass x-rays through the patient at different positions and angles, detect the amount of attenuation for each, and reconstruct the underlying anatomy based on this data. The following figure shows a simple configuration.

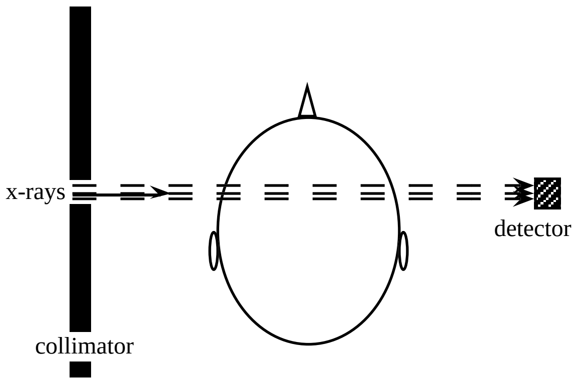


Figure 5.3.2.1. Basic CT operation

In the figure, x-rays emerge from the source, pass through the patient, and are stopped in the detector where their intensity is measured. The detector reports the *ray sum*, or total attenuation for a given ray. This is determined by the amount of absorption and scatter in the patient (both of which depend on the type of tissue or matter encountered by the beam). The collimator assures that the rays are essentially parallel on exit so that the beam does not diverge (and thereby partially miss the detector).

An algorithm known as *filtered back-projection* is commonly used to reconstruct the slice from the ray sums. The term *back-projection* comes from the way the ray sums are projected back along the ray and averaged into each voxel along it, effectively smearing the detector reading back along the ray that produced it. When this is done for all angles, the original image is reconstructed. The filtering done is to negate starlike blurring caused by the projection and back projection and to minimize starlike artifacts caused by using a finite number of ray angles to sample the volume, and to suppress noise and enhance edge features.

The errors in real CT images come from a number of sources; the main ones are described below [Lee & Rao, 87].

- *Patient motion:* Even though modern systems can take a scan in less than 5-10 seconds, a patient can move considerably in this amount of time. This causes blur and streaking, especially in the presence of high-density structures.
- *Beam-hardening artifacts:* Reconstruction of the slice from the ray sums is a process of assigning attenuation coefficients to each pixel. However, the attenuation

coefficient is dependent on beam energy, and the beam is actually polychromatic (containing photons with a spectrum of energies). As the beam passes through the patient, lower-energy photons are preferentially absorbed (thereby *hardening* the beam), so that the beam passing through a voxel at one angle will often have a different energy spectrum from another beam at a different angle. This leads to inconsistencies in the attenuation coefficients as calculated in different views, and the result is streaks in the final image.

- *Partial-volume effect:* Because of the finite voxel (volume element) size, voxels containing multiple types of tissue (or air and tissue, etc.) will essentially be an average value of the tissue types contained within it. This leads to blurring of object boundaries and loss of detail in fine structures. This is particularly relevant for the identification of landmark points, which are often defined in terms of some fine structure (such as a foramen, which is a small perforation in a bone or membrane) which may be blurred or non-existent in the final image. The extent of this error is determined by the voxel size.
- *Aliasing:* Related to the partial-volume effect is the problem of aliasing. Aliasing appears when we use a finite number of samples to represent a continuous signal (here, the anatomy) and causes high-frequency features to appear as low-frequency features (i.e., false structures or artifacts) in the final image. This is also a problem in computer graphics and is discussed in more detail in Section 5.7.8.1. False structures caused by aliasing might be mistaken for real structures, leading to incorrect landmark identification. Similar to partial-volume errors, aliasing is reduced by using smaller (contiguous) voxels.
- *Scatter:* Because photons are both scattered and absorbed, some of the radiation that reaches a detector will have come along a different path from the straight-line radiation passing through the anatomy. This leads to inconsistencies which show up as streaks.
- *Equipment errors:* This category includes mechanical misalignment of the x-ray source or detector, variation in the beam intensity during the scan, variation between detectors, and detector nonlinearities. The most common effect of these is streaks in the reconstructed image.

There are other error sources in addition to those just listed; the interested reader is referred to [Lee & Rao, 87]; the point here is that the scanning process is imperfect and that the errors can show up as blur or more importantly, as false structure in the imaged volume. Section 6.2 will give more details on the magnitude of these errors.

5.3.3. Polygonalization

The output of the CT reconstruction is a 3D volume of CT numbers which must be rendered by the graphics engine in some fashion. This process is known as *volume visualization*, and there are numerous methods used. A common method is to generate a set of polygons that represent the surfaces of interest in the dataset, such as the outer surface of the skull. The resulting set of polygons can be rendered using conventional computer graphics hardware. Other methods, such as volume rendering and binary voxel techniques, are also used and are discussed in [Fuchs et al. 89]. Since polygonalization is commonly used in the medical community and since polygonal representations are the most common form for other (non-medical) applications, I will assume that the CT volume is to be converted to a set of polygons and will discuss some of the errors in this process.

Polygonalization algorithms divide the 3D volume into a set of polygons (usually triangles) using some criterion such as a density threshold. An algorithm known as *marching cubes* [Lorensen & Cline 87] is a classic example; it creates triangle models of constant density surfaces from the 3D volume and the triangles are rendered using traditional methods. A sketch of the algorithm (from [Lorensen 90]) follows.

1. Create a cube from 8 adjacent voxel values
2. Classify each vertex of the cube as being inside (value >= threshold) or outside (value < threshold)
3. Build an 8-bit index for each cube based on the inside/outside classification for each vertex
4. Use the index to access a list of cases for edge crossings
5. Interpolate cube vertices to determine crossing points for this case
6. Calculate and interpolate normals for triangle based on gradient of density data

The upshot is that each cube in the dataset (defined by 8 adjacent voxel values) has some number of triangles fit to it based on the 8 vertex values. This gives a best-fit surface for the data within the cube; doing this for all cubes in the volume yields a large set of polygons which, in theory, represent the surface of interest.

The problem with polygonalization is that the binary classification of the data often yields artifacts for small or poorly-defined structures. The two most common artifacts are false surfaces/structures and false holes, which can be caused by noise, partial-volume effects, or other imaging artifacts as presented in the previous section.

5.3.4. Real/Virtual Landmark Alignment Errors

Once the anatomy has been imaged and polygonalized, the next task is to register the resulting dataset (the *virtual dataset* or *virtual anatomy*) with the real anatomy in World space for display with the STHMD. A common method for 3D registration is to choose a number of landmarks (LMs) on the real anatomy and the virtual dataset and compute the transformation that minimizes the distance between corresponding landmark pairs. The landmarks for the virtual dataset (the *virtual landmarks*) may be identified offline (on a 2D monitor, for example) and the landmarks on the real anatomy (the *real landmarks*) will normally be identified by digitizing points on the real anatomy with a 3D stylus of some sort.

The real landmarks have at least three sources of error:

1. **Soft-tissue thickness:** Because the real skull is covered with soft tissue and is not accessible directly, the digitized landmarks will normally be at some distance from the surface of the skull. Depending on the patient and the landmark, the tissue thickness error ranges from 0 (for digitizing a landmark on a tooth, for example) to 15 mm or more at various points on the jaw [Bartlett et al. 91]. However, because the corresponding point in the CT volume will also be digitized and since the CT data can also be used to detect the skin surface, it should be possible to compute a correction based on the distance along the normal from the bone-surface (virtual) landmark to the landmark on the skin. Nevertheless, the presence of the soft tissue and its deformability will make it more difficult to identify the landmark relative to its bony counterpart.

2. **Digitizer error:** As we shall see in the sections that follow, trackers all have some amount of measurement error, so even if the stylus tip could be placed exactly on each landmark, the reported value would be off by some amount.

3. **Human error:** As noted in [Herman 91], there can be significant variability in landmark locations as chosen by different people, or even by the same person at different times. (We will discuss this in more detail in Section 6.2.)

The virtual landmarks also have at least three sources of error:

1. **Medical imaging error:** As discussed in Section 5.3.2.

2. **Polygonalization error:** As discussed in Section 5.3.3.

3. **Human error:** As above.

Note that because the virtual anatomy is represented by a set of polygons, the virtual landmarks can be chosen on a 2D monitor at any scale and can be constrained to lie on the polygonal surface. Thus, digitizer error for these landmarks should be negligible.

The real and virtual landmarks (containing these errors) are then used as input to the alignment algorithm, which finds a least-squares fit of the virtual LMs to the real LMs. The next question is how these errors are propagated by the alignment algorithm. Because of the complexity of the algorithms used for this process, I will not attempt to derive an analytical error model for this stage. Rather, Section 6.2 presents some empirical data on the error-propagation behavior of one representative algorithm for typical input error distributions.

5.4. Head-Tracking Errors

The next error source is measurement error in the transformations that define where the STHMD is in World space. Errors in the two component transformations T_{W_T} and T_{T_S} will be discussed next.

5.4.1. World-Tracker Transformation

It is common practice to have a user- or system-defined World CS as a reference. An example of when we need a World CS is when we have a special digitizer (such as a camera measurement system) for precisely locating points in W, but which is not suitable for head tracking. We then have W as the reference CS and T is expressed relative to it.

In such cases, we use the transform T_{W_T} to express where the tracker is in World space. The values for this transform come from measurement of where the tracker's origin and orientation are within the room in which it is located. As always, measurement means that some amount of error is made, and the question to be answered is how much error and how it is propagated.

In the cases where the World CS is user-defined, an obvious first step at eliminating error would be to define it to be identical to the Tracker CS, which would make T_{W_T} the identity transform. Even so, there are reasons to retain the World CS (and therefore this transform). The World CS provides a stable, well-defined reference for all other measurements and calibration procedures. In contrast, the origin and orientation of the

Tracker CS is often known only by inference from the tracker's measurements[13], which contain some amount of noise; this means that the reference CS may seem to move as a function of tracker error. I will therefore include this transform in the model with the understanding that certain systems may be able to omit it.

In this section and the next section, we will use the derivation done in Section 4.4 for the error in compositions of rigid-body transformations. Here, the World CS is defined within the laboratory arbitrarily and thus corresponds to the '0' coordinate system of Section 4.4.1. Thus, this transformation is the first in the series and therefore contains no error from prior stages. Setting N = 1 into Equations 4.4.1.20 and 4.4.1.21 gives the following error bounds:

Rotation: $|Ø_{T''\_T}| \leq |Ø_{T'\_T}|$

Translation: $\|\mathbf{v}_{T''\_T}\| \leq \|\mathbf{v}_{T'\_T}\|$

That is, since there is no prior error to propagate, T'' = T', and the error in this transformation is just whatever error was made in the measurement of Tracker space relative to World space. Since the error here is strictly measurement error, it does not vary with time and is comparatively straightforward to estimate, depending on the measurement method and tracking technology used. As for the other transformations, numerical error bounds and sensitivity analysis for this transformation will be treated in the next chapter.

5.4.2. Tracker-Sensor Transformation

In this stage, the position and orientation of the tracker sensor mounted on the HMD is calculated by the tracker and sent to the image-generation system. This is represented by the transformation $T_{T\_S'}$, which gives the measured position and orientation of the Sensor coordinate system with respect to the Tracker CS. This transform is composed with $T_{W\_T'}$ to yield $T_{W\_S''}$, which gives the apparent position and orientation of the Sensor in World space.

In contrast to the other transformations in the pipeline, $T_{T\_S'}$ is measured and updated each frame. While any of the other transformations might change with time due to slop in the system (e.g., head slippage inside the HMD causing the eyepoints to move relative to the screen images), these are relatively small variations and are not usually measured. The

[13] For example, the origin of the Tracker CS for magnetic trackers is typically in the middle of the transmitter and therefore unavailable as a reference for external measurements.

errors associated with these transformations are described here by a single error bound denoting the maximum expected deviation from the modeled transform. In contrast, the changes in the Tracker-Sensor transform are quite large and must be measured.

Any measurement of a non-repeating, time-varying phenomenon is valid (at best) at the instant the measurement is made, and then becomes out of date with the passage of time until the next measurement. The age of the data is thus a measure of its accuracy. Any delay between the time the measurement is made and the time that measurement is manifested by the system contributes to the age and therefore the inaccuracy of that measurement.

The Tracker-Sensor transform is measured and then used to generate the next frame's image. The manifestation of the measurement, then, is when the image calculated with the tracker data is finally visible to the user. As a result, all delays from the tracker measurement until the associated image is visible to the user contribute to the age of the tracker data. The older the tracker data is, the more likely that the displayed image will be misaligned with the real world.

This delay between a user's head movement and the display of images manifesting that movement will be called the *tracker-to-display latency*. This latency is defined as the sum of all of the delays between the tracker measurement and the final image display; it is a measure of how old the tracker data used for the image is. Although many of the delays that contribute to the tracker-to-display latency are not specifically associated with the tracker, they each contribute to the discrepancy between the real and reported head position and orientation at display time, and therefore are listed with the other sources of error for this transformation.

The following list enumerates the static and dynamic tracker errors, and then the delays which affect the age of the tracker data.

Tracker Measurement Errors

- *Static field distortion:* For an immobile sensor, we can divide the measurement errors into two types: repeatable and nonrepeatable. That is, 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. We will therefore assume that the gross field distortion has been calibrated out of the system and will lump any residual error in with the jitter term, which is discussed next.

- *Jitter:* This category of error is used for all the non-repeatable errors made by the tracker for an immobile sensor. Some amount of noise in the sensor inputs is to be expected with measurement systems, and this input noise leads to jitter, or noise in the outputs. By our definition, this type of error is not repeatable and therefore not correctable 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 and the tracker's origin and may become very large near the edge of the tracker's working volume. The sum of tracker jitter and residual static field distortion will be called *static tracker error* in the derivations that follow.

- *Dynamic tracker error:* Some trackers' errors increase for objects in motion. This is caused by the finite period of time it takes to collect data in order to derive the position/orientation information. For magnetic trackers, there is a finite period of time for sensing the field induced in the receiver coils by the transmitter coils; in addition, the three transmitter coils are pulsed sequentially (to avoid interference), which increases the measurement period. The readings are then combined in order to derive the position and orientation of the sensor. The trackers assume that the sensor is in approximately the same place for all three measurements, an assumption which is violated for rapid head motions. The result is dynamic tracker error, and its magnitude clearly depends on the amount the sensor moves between measurements.

Delay-Induced Error

- *Tracker delay:* This is the time required for gathering data, making calculations in order to derive position and orientation from the sensed data, and transmitting the result to the host.

- *Host-computer delay:* The host computer is typically charged with reading the tracker data and sending it on to the graphics engine. This may involve servicing serial-port interrupts to fetch the data, decoding it, converting it into a different representation (e.g., from quaternion and vector to matrix), scaling into the appropriate units (e.g., inches to meters), etc., all of which adds delay. In addition, the host for the UNC system is a Unix system, which has other operating-system tasks that it must periodically perform (e.g., garbage collection, updating the clock, etc.) which add to the delay as well.

- *Image-generation delay:* After the tracker data is received, the graphics computer must incorporate this latest report into the viewing matrix for calculating the next image. This, too, takes a finite amount of time, which introduces further delay.
- *Video sync delay:* Once the image is rendered into the frame buffer, it still must be scanned out to the display device. For raster displays, the display controller must wait until the last frame has been scanned out before scanning out the next one. Thus, even if the image were computed instantaneously with up-to-date tracker data, it might be an entire frame before that image started being scanned out to the display. This can be avoided by careful synchronization between the image-generation system and the scan-out hardware, but such sync is not done in many systems.
- *Frame delay:* Most raster devices (such as CRTs) paint the image one pixel at a time, from scan-line to scan-line on down to the bottom of the screen. For a 60 Hz non-interlaced display, this amounts to roughly 17 ms between the display of the upper left pixel and the lower right pixel. If this is not compensated for, the information toward the bottom of the screen will be at least 10-17 ms old when it is drawn on the display.
- *Internal display delay:* Some display devices add additional delays due to processing within the display device itself. For example, [Mine 93] reports that the LCDs in an HMD in use at UNC added an additional field time (about 17ms) of delay. This could be due to the display having a different resolution from the input signal and having to resample the input before it can display the current frame.

The dynamic tracker error and the delay-induced errors are functions of the speed of head movement by the user. This is clearly application-dependent: One would expect fighter pilots to move much more rapidly than surgeons, for example. Estimates of static and dynamic errors, delays, and typical user head velocities are discussed in the next chapter.

The error-model derivation for this section will begin by lumping all of the errors listed above together into an angular error and a translational error. We can then further divide each of these into static, dynamic, and delay-induced errors.

Setting $N = 2$ into Equations 4.4.1.20 and 4.4.1.21 gives the following error bounds for $T_{S''\_S}$:

$$\textit{Rotation:} \quad |Ø_{S''\_S}| \leq |Ø_{T'\_T}| + |Ø_{S'\_S}| \tag{5.4.2.1}$$

$$\textit{Translation:} \quad \|\mathbf{v}_{S''\_S}\| \leq \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{S'\_S}\| + \left|2\cdot\sin\frac{Ø_{T'\_T}}{2}\right| \cdot \|\mathbf{v}_{T\_S'}\| \tag{5.4.2.2}$$

Note that the angular errors just add, as expected; the translation terms add, but also have an additional term involving the rotational error from the previous transformation ($T_{W\_T'}$) and the measured tracker-to-sensor distance. Thus, any angular errors made in the previous transformation are scaled by a vector magnitude which may be on the order of a meter or more, depending on the application. The next chapter will examine this in more detail.

Now that we have a high-level expression for the error in the World-Sensor transformation, it would be useful to relate this expression to the error sources listed above. In particular, we would like to relate the delay terms above to the angular and translational errors which are inputs to the error bounds calculated above. In the worst case, the net delay from the above sources is just the sum of the individual delays, and so for now we can group all of the delays into a single variable, Δt, where

$$\Delta t = \Delta t_{tracker} + \Delta t_{im\text{-}gen} + \Delta t_{host} + \Delta t_{sync} + \Delta t_{frame} + \Delta t_{display} \qquad (5.4.2.3)$$

which corresponds to the worst-case delay for the last pixel drawn on the screen for a given frame.

If we use a simple first-order model for head motion, we have

$$\textit{Rotation:} \quad |\text{Ø}_{delay}| = |\dot{\text{Ø}}_{head}\Delta t| \qquad (5.4.2.4)$$
$$\textit{Translation:} \quad \|\mathbf{v}_{delay}\| = \|\dot{\mathbf{v}}_{head}\Delta t\| \qquad (5.4.2.5)$$

where $\dot{\text{Ø}}_{head}$ and $\dot{\mathbf{v}}_{head}$ are the angular and linear velocity of the user's head, and Ø_{delay} and $\mathbf{v}_{delay}$ are the net rotation and translation as a result of the delay (ignoring higher order terms).

As discussed above, dynamic error is also a function of head motion; i.e., the more the head has moved between measurements that were assumed to be at the same location, the more dynamic error there is. Since this is tracker-dependent, we will just list it as Ø_{t_dyn} and $\mathbf{v}_{t\_dyn}$ with the understanding that both of these are functions of head movement and must be measured for each tracker under varying types of head motion.

If we use Ø_{t_stat} and $\mathbf{v}_{t\_stat}$ to denote the static tracker angular and translation errors in $T_{S'\_S}$, the total error bound for the Tracker-Sensor transform is

$$\textit{Rotation:} \quad |\text{Ø}_{S'\_S}| \leq |\text{Ø}_{t\_stat}| + |\text{Ø}_{t\_dyn}| + |\dot{\text{Ø}}_{head}\Delta t| \qquad (5.4.2.6)$$
$$\textit{Translation:} \quad \|\mathbf{v}_{S'\_S}\| \leq \|\mathbf{v}_{t\_stat}\| + \|\mathbf{v}_{t\_dyn}\| + \|\dot{\mathbf{v}}_{head}\Delta t\| \qquad (5.4.2.7)$$

The total error term for the transformations composed so far is thus

$$\textit{Rotation:} \quad |\text{Ø}_{S''\_S}| \leq |\text{Ø}_{T'\_T}| + |\text{Ø}_{t\_stat}| + |\text{Ø}_{t\_dyn}| + |\dot{\text{Ø}}_{head}\Delta t| \qquad (5.4.2.8)$$

$$\textit{Translation:}\quad \|\mathbf{v}_{S''\_S}\| \leq \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{t\_stat}\| + \|\mathbf{v}_{t\_dyn}\| + \|\dot{\mathbf{v}}_{head}\Delta t\| + \left|2\cdot\sin\frac{\text{Ø}_{T'\_T}}{2}\right|\cdot\|\mathbf{v}_{T\_S'}\| \quad (5.4.2.9)$$

It should be clear that for any significant amount of delay, or any significant angular or linear velocity, the delay-induced errors can be quite large.

5.4.3. Registration Error in Expressing P in Sensor Space

Now that we have derived the error in measuring P in World space and the error in the transformations from World space to Sensor space, we can derive an expression for the total registration error up to this point (this would be the total error if no errors were made in the display or viewing of the point).

In Section 4.4.2 we bounded the error for transforming a vector when both the input vector and the transformation contained errors:

$$\|\mathbf{v}_{P''\_P}\| \leq \|\mathbf{v}_{P'\_P}\| + \|\mathbf{v}_{B'\_B}\| + 2\cdot\left|\sin\frac{\text{Ø}_{B'\_B}}{2}\right|\cdot\|\mathbf{v}_{B\_P'}\| \quad (4.4.2.3)$$

If we use $\|\mathbf{v}_{P'\_P}\|$ as the error bound for acquisition/alignment errors and substitute the head-tracking error terms derived in the last section for the rotation and translation terms in Equation 4.4.2.3, we get a bound on the error vector magnitude for all of the errors considered so far of

$$b = \|\mathbf{v}_{P'\_P}\| + \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{S'\_S}\| + 2\cdot\left|\sin\frac{\text{Ø}_{T'\_T}}{2}\right|\cdot\|\mathbf{v}_{T\_S'}\| + 2\cdot\left|\sin\frac{|\text{Ø}_{T'\_T}| + |\text{Ø}_{S'\_S}|}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \quad (5.4.3.1)$$

The error bound for head-tracking errors alone is just

$$b_{head\_track} = \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{S'\_S}\| + 2\cdot\left|\sin\frac{\text{Ø}_{T'\_T}}{2}\right|\cdot\|\mathbf{v}_{T\_S'}\| + 2\cdot\left|\sin\frac{|\text{Ø}_{T'\_T}| + |\text{Ø}_{S'\_S}|}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \quad (5.4.3.2)$$

In Chapter 6 we examine the implications of this equation for a real system.

5.5. Overview of Display and Viewing Errors

Now that we have treated the errors caused by the imperfect localization of the virtual point in the world and the errors caused by our imprecise knowledge of where the STHMD is in the world, we turn to the issue of what goes wrong inside the STHMD itself. As we have just seen, the rigid-body transformation errors lead to an uncertainty in the position and orientation of the Sensor CS, which means that even a perfect display (barring some sort of feedback mechanism) can do no better than to perfectly display the point P", which is defined by the vector $\mathbf{v}_{S''\_P'}$ applied in S.

Viewing error and display error are interrelated, and so we must begin by discussing the two together before breaking off into separate analyses of each. Consider the case of error-free projection for a single eye as depicted below.

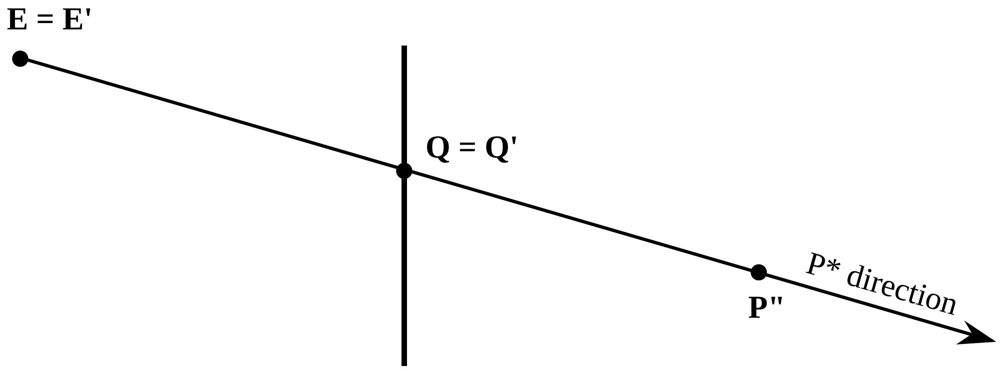


Figure 5.5.1. Error-free projection for one eye

In this case, the eyepoint E coincides with the modeled or measured eyepoint E', the displayed projected point Q coincides with its modeled counterpart, Q', and the ray from E through Q passes through the point P", as desired. Because there are errors in both the modeled eyepoint and in the display system, E will deviate from E' and Q from Q'. In the presence of such errors, the ray EQ will not generally pass through P", leading to further registration error. The goal here is to derive a general expression for bounding the error due to viewing errors (E' ≠ E) and display errors (Q' ≠ Q).

Given the measured eye location E' and a modeled projected point Q' (the projection of P") and bounds on the error vector magnitudes for each of $||\mathbf{v}_{E'\_E}||$ and $||\mathbf{v}_{Q'\_Q}||$ (we will derive analytical bounds for these in the following sections), we can model the situation as depicted in the figure below.

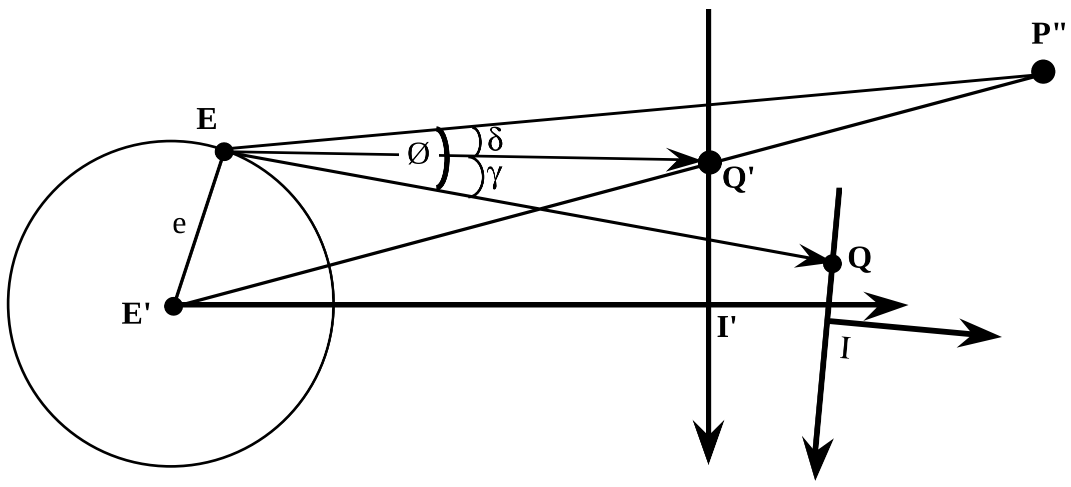


Figure 5.5.2. Errors in modeling eye and projected point positions

Here, the measured image coordinate system I' (*i.e.*, LI' or RI') is used as the reference CS, since this allows us to vary both the actual eyepoint (E) and the actual image CS (I) with respect to a convenient reference CS. The error up to this stage is encapsulated in the difference between P and P"; the error due to viewing and display error will be encapsulated in the vector between P" and P*.

The circle around E' represents the sphere of uncertainty of radius $e = \|\mathbf{v}_{E'\_E}\|$ surrounding the modeled eyepoint. Q' is the modeled projected point and Q is the actual displayed point. The angular registration error due to both display and viewing errors is specified by the angle $Ø = \angle QEP''$. Ø clearly must be measured in the plane defined by the points Q, E and P" as shown in the figure; we can therefore treat this as a 2D problem. The worst case clearly occurs when E' and Q' are also in the plane QEP", since this maximizes the distance within that plane between E and E' and between Q and Q'.

Ø has been divided into the sum of two angles, δ and γ. δ is the angular error due to the eye deviating from its modeled location, and γ is the error due to both eyepoint deviation and display errors. In other words, the viewing error is how much error there would be even if there were no display error at all (Q = Q'), and the display error is the error induced by the projected point deviating from its modeled location, complicated by the eyepoint not being at its modeled location. Thus, the viewing error (characterized by δ) is not affected by $\|\mathbf{v}_{Q'\_Q}\|$, whereas γ *is* affected by the viewing error. This is because γ is the angle that the line segment QQ' subtends from E, which clearly changes as E's location varies. δ, however, is just the angle Q'EP", which does not vary with $\|\mathbf{v}_{Q'\_Q}\|$ at all. Therefore, we can begin the analysis by studying δ alone, and then use these results in the analysis of the display errors.

5.6. Viewing Error

5.6.1. Overview

Viewing error is defined as registration error caused by the modeled eyepoints not corresponding to the actual eyepoints. Up to this point, we have used the term *eyepoint* without rigorously defining it, since the previous discussions involving it did not require a precise definition. There has been some debate on the issue: [Deering 92] uses the first nodal point of the eye[14] as the center of projection for his system, while [Rolland et al. 93] argue that the center of the *entrance pupil*[15] is the correct point to use. This analysis will use the center of the entrance pupil, which is approximately 11 millimeters forward of the center of rotation for the eye (vs. 6mm for the first nodal point). For the detailed arguments for using the center of the entrance pupil as the center of projection, refer to [Rolland et al. 93].

Because the images displayed by the system are calculated for a given viewpoint, any deviation of the eyepoints from the modeled locations will introduce some amount of error. There are a number of reasons for the eyepoints to be offset with respect to their modeled locations:

- *Calibration error:* The locations of the user's eyes are normally determined through a calibration procedure. No method is perfect, and thus there will be some amount of measurement error associated with each method.
- *Eye movement not tracked:* As the eye rotates in its socket, the eyepoint moves because it is not coincident with the center of rotation for the eye. If there is no eye tracking in the system (as is usually the case), the eye movement will induce errors. Even if there is eye tracking, it will not be perfect, and so there will be some residual error even in these systems.

---

14 This is the intersection of the plane of unit angular magnification with the optical axis; a ray passing through the first nodal point will emerge at the same angle from second nodal point.

15 The entrance pupil is the image of the aperture stop as seen from object space; in this case, it is the image of the pupil seen from outside the eye.

- *Slippage of HMD on user’s head:* Each time that a users dons the HMD, there will be some variation in the positioning of the device on the head, and therefore some variation in the eye position relative to the screens’ virtual images. In addition, the HMD may slip on the head as the user looks around the environment, since it is not rigidly fixed to the head.

Numerical bounds for these error sources will be derived in the next chapter. The question addressed in this section is, How do discrepencies in real eyepoint locations vs. their modeled locations affect the registration error? Because of the problems with unbounded linear and depth errors in stereoscopic displays described in Section 5.1, I will use the decomposition error metrics discussed in that section for many of the derivations here. I will begin by treating the angular and lateral errors for one eye, and then will analyze the case for binocular viewing which will include an analysis of depth error. In all sections of the viewing-error analysis, the point P^* will be understood to be the apparent point due to viewing error *only*.

5.6.2. Angular Viewing Error for One Eye

The figure below shows the angular registration error due to viewing error for one eye. I have exaggerated the size of the error sphere around E' in order to make the geometry clearer and have chosen the I' coordinate system such that the Q'EP" plane falls in the X-Z plane in order to simplify the analysis.

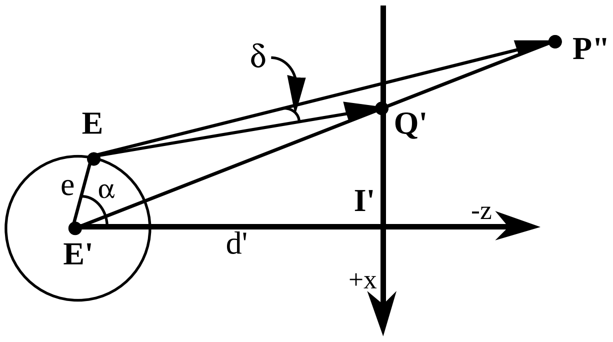


Figure 5.6.2.1. Viewing error diagram

The point P" = (x", y", z") is projected onto the screen image at Q' for the modeled eyepoint E'. However, the eyepoint is actually at E = (x_e, y_e, z_e), leading to an angular registration error defined by ∠Q'EP", or δ. *d'* is the perpendicular distance from the modeled eyepoint to the modeled screen image, and *e* is the radius of the error sphere around E' and is equal to $||\mathbf{v}_{E'\_E}||$.

The angle α specifies where E is on the periphery of the sphere. This clearly affects δ, which varies from 0 when E is along the line E'Q' to some maximum value yet to be determined. As shown in the figure, we can derive the angle δ by considering the vectors $\mathbf{v}_{E\_Q'}$ and $\mathbf{v}_{E\_P''}$ and using the definition of the dot product to calculate the angle between them:

$$\mathbf{v}_{E\_P''} \cdot \mathbf{v}_{E\_Q'} = ||\mathbf{v}_{E\_P''}|| \cdot ||\mathbf{v}_{E\_Q'}|| \cos\delta \qquad (5.6.2.1)$$

The expressions for the two vectors can be derived by first expressing all points relative to I' (with E as a function of α) and then expressing the vectors as differences of points:

$$\mathbf{v}_{E\_P''} = P'' - E = (x'' - x_e, z'' - z_e) \qquad (5.6.2.2)$$

$$\mathbf{v}_{E\_Q'} = Q' - E = (x_{I'\_Q'} - x_e, 0 - z_e) = \left(\frac{d'\ x''}{d' - z''} - x_e, -z_e\right) \qquad (5.6.2.3)$$

where

$$x_e = -e\sin\alpha \qquad (5.6.2.4)$$
$$z_e = d' - e\cos\alpha \qquad (5.6.2.5)$$

(Note that the expression for $x_{I'\_Q'}$ comes from the perspective projection derivation in Section 4.5.1.)

This leads to

$$\cos\delta = \frac{\mathbf{v}_{E\_P''} \cdot \mathbf{v}_{E\_Q'}}{||\mathbf{v}_{E\_Q'}|| \cdot ||\mathbf{v}_{E\_P''}||} = \frac{(x''-x_e)\left(\frac{d'x}{d'-z''} - x_e\right) - z_e(z''-z_e)}{\sqrt{\left[(x''-x_e)^2 + (z''-z_e)^2\right] \cdot \left[\left(\frac{d'x}{d'-z''} - x_e\right)^2 + z_e^2\right]}} \qquad (5.6.2.6)$$

This can be simplified by taking the sine of both sides and using the fact that

$$\sin(\operatorname{acos}A) = \sqrt{1 - A^2} \qquad (5.6.2.7)$$

After some manipulation, the two expressions can be combined to get

$$\delta = \tan^{-1}\left(\left|\frac{z_e(x''-x_e) + (z''-z_e)\left(\frac{d'x}{d'-z''} - x_e\right)}{-z_e(z''-z_e) + (x''-x_e)\left(\frac{d'x}{d'-z''} - x_e\right)}\right|\right) \qquad (5.6.2.8)$$

which is valid for the range $0 \le \delta \le 90^\circ$.

Ideally, one would expand x_e and z_e in terms of α and differentiate with respect to α to find the maximum value of δ as the eyepoint moves along the periphery of the sphere. Unfortunately, the complexity of the fully expanded expression for δ makes this task non-trivial, even with a tool such as Mathematica. In lieu of an analytical solution, we can examine the behavior of this function for some typical parameter values and see that the situation is neither overly complex nor mathematically intractable.

If we vary both α and *x″* while holding the other parameters constant, we get a plot of δ as a function of both parameters. For this analysis, I have varied *x″* in such a way that the viewing-direction angle θ from E' to P" changes uniformly[16]. The figure below depicts the situation.

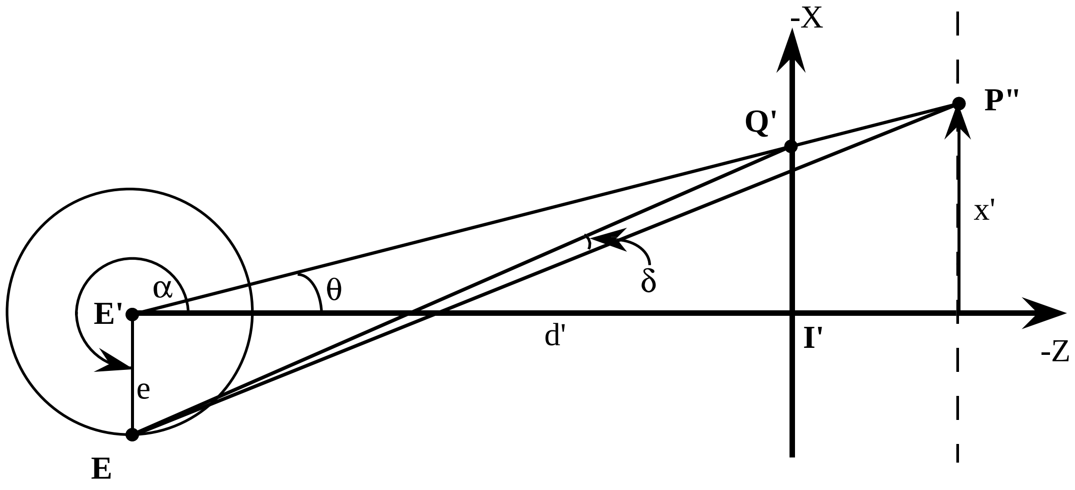


Figure 5.6.2.2. Viewing error

The graph below shows δ as a function of θ and α.

[16] This is done by making *x'* a function of θ as follows: $x' = (d - z')\cdot\tan\theta$.

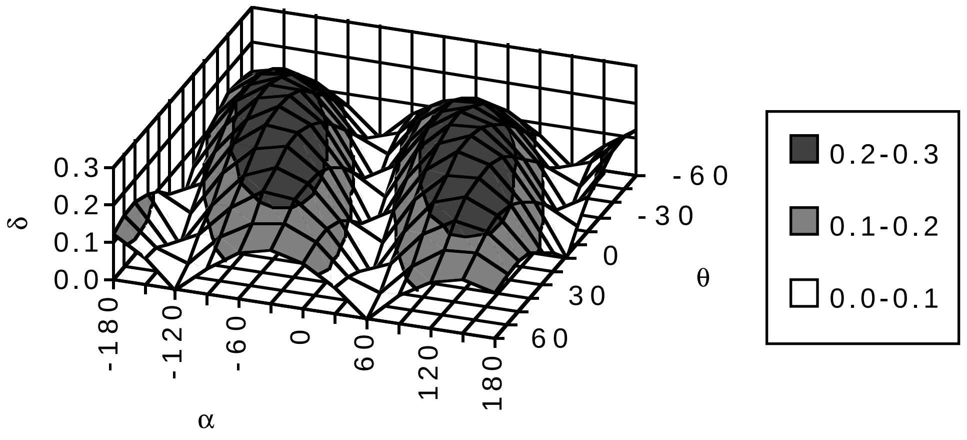


Figure 5.6.2.3. Angular viewing error as a function of θ and α; all angles in degrees

In the graph above, *z"* is fixed at 100 mm (100 mm in front of the screen image), *e* is set at 10 mm, and *x"* is varied such that θ changes uniformly from 0˚ to 60˚ (corresponding to a 120˚ horizontal field of view, which is beyond that of many current commercial systems).

The first observation is that there are two peaks at (θ = 0˚, α ≈ -90˚) and (θ = 0˚, α ≈ +90˚). Thus, for these parameter values at least, the worst viewing error occurs for on-axis points with eye movement that is at right angles to the viewing direction. The valleys in between the peaks correspond to places in which E lies along the projector E'P", in which case δ = 0.

Further manipulation of the parameters that were fixed for the plot above yields the following results:

- The worst-case angular viewing error is always for on-axis points (θ = 0˚; x" = 0). While the number of interrelated parameters makes a proof of this non-trivial, an intuitive argument may lend some insight: For off-axis points, the segment Q'P" subtends the largest angle (i.e., δ is a maximum) when the segment is closest to the circle of uncertainty, which occurs for on-axis points. As the P" and Q' move off-axis, the distance to the segment increases and the angle it subtends decreases. (This intuitive proof is supported by calculation of δ for all reasonable values of the involved parameters: the worst case is always for on-axis points.)

- As long as *e* is small with respect to *d'* and the distance E'P" (= *d'-z"*), the peaks occur at roughly 90˚ and 270˚ in α. That is, for on-axis points, the worst-case viewing error is for eye movement perpendicular to the viewing direction. This is intuitive in that one would expect the maximum error to correspond to an eye position that gives the most parallax, which is a movement perpendicular to the viewing direction.
- When *e* is no longer small with respect to *d'* and E'P", (i.e., about 10% of *d'*), the peaks shift away from each other along the α axis. For example, when e = 100, d' = 500, and z" = 100, the peaks occur at α ≈ 60˚ and 300˚. This, however, is a grosser viewing error than we would ever expect in a real system.

Because the worst-case viewing error occurs for on-axis points, we can try to obtain a simpler expression for δ by setting x" = 0 in Equation 5.6.2.8. Moreover, if we assume that *e* is small with respect to *d'* and the distance E'P" (which is the case in most systems), we can set α = 90˚ and further simplify the expression. This corresponds to a point directly in front of the eye; if both displayed points are directly in front of each eye, the point should be perceived as straight ahead and infinitely far away. While this is not the case in arm's-length applications such as surgical planning, as long as *d'* is large relative to *x*, the on-axis case is a reasonable approximation. For example, for a point at the typical working distance for this application of 500mm, the convergence angle for each eye is about 3.5˚, as shown in the figure below.

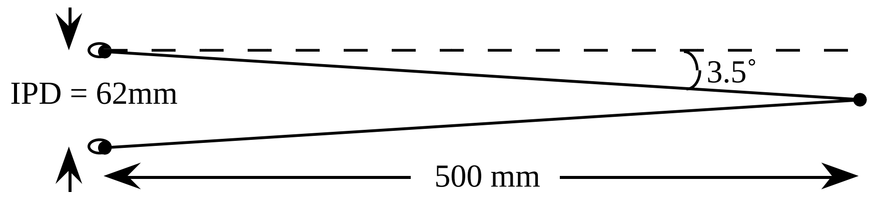


Figure 5.6.2.4. Convergence angle for arm's-length point

A viewing angle of 3.5˚ is still well within the peak region in Figure 5.6.2.3, and is therefore close enough to the worst-case angle of 0˚ that it is not overly pessimistic for bounding the viewing error for an arm's-length application.

Thus, substituting α = 0 and x" = 0 into Equation 5.6.2.8 yields

$$\delta \approx \tan^{-1}\left(\left| \frac{e \cdot z''}{e^2 + d'^2 - d'z''} \right| \right) \qquad (5.6.2.9)$$

We may also be interested in this function's behavior for arbitrarily distant points. As *z″* approaches minus infinity, we have

$$\lim_{z'' \to -\infty} \delta \approx \tan^{-1}\left(\frac{e}{d'} \right) \qquad (5.6.2.10)$$

corresponding to the situation depicted below.

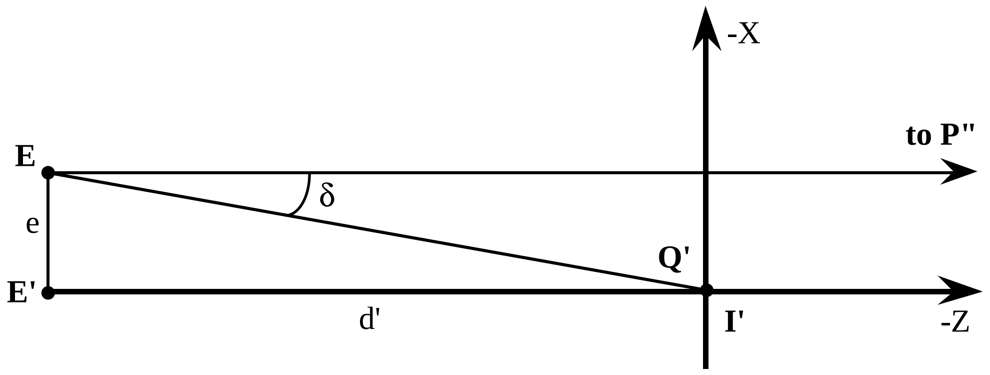


Figure 5.6.2.5. Viewing error for infinitely distant on-axis point

However, since the geometry of this case is somewhat simpler than the general case, we can see that the exact expression follows from the following figure.

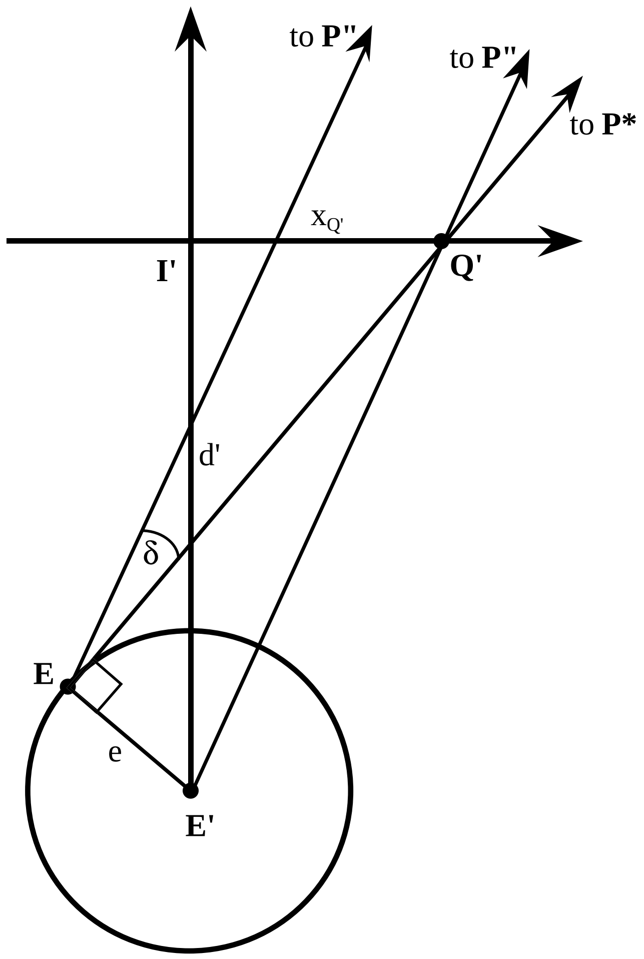


Figure 5.6.2.6. Worst-case viewing error for distant points

That is, δ is a maximum when the distance Q'E' is the shortest, which is for $x_{Q'} = 0$ (Q' is at origin). In this case the segment I'E' is the hypotenuse of length *d'*, and we have

$$\lim_{z'' \to -\infty} \delta = \sin^{-1}\left(\frac{e}{d'}\right) \tag{5.6.2.10}$$

as a more precise upper bound on δ for arbitrarily distant points.

5.6.3. Lateral Viewing Error for One Eye

We can now look at the lateral error implied by these expressions for δ using

$$s_{view} = 2r\cdot\sin\frac{\delta}{2} \tag{5.1.3}$$

The full expression is too complex to yield much insight, so we can move directly to the worst-case expression (with $\alpha = 90$, $x'' = 0$). After some simplification, we have

$$s_{view} = 2r\cdot\sin\frac{\delta}{2} = \sqrt{2e^2 + 2(d'-z'')^2} \cdot \sqrt{1 - \frac{e^2 + d'(d'-z'')}{\sqrt{(e^2 + d'^2)(e^2 + (d'-z'')^2)}}} \qquad (5.6.3.1)$$

which is still rather complex. For small angles, we can use $2\sin\frac{\delta}{2} \approx \sin\delta$ and then use Equations 5.6.2.6 and 5.6.2.7 to convert the express for $\cos\delta$ into one for $\sin\delta$, which eventually simplifies to

$$s_{view} \approx \left| \frac{e\cdot z''}{\sqrt{e^2 + d'^2}} \right| \qquad (5.6.3.2)$$

Furthermore, if *e* is small with respect to *d'*, we can ignore the e^2 term and simplify this further to

$$s_{view} \approx e\cdot\frac{|z''|}{d'} \qquad (5.6.3.3)$$

Thus, the lateral error increases linearly with the distance *e* and with the screen-image-to-point distance. That is, for points not in the plane of projection, the lateral error will vary linearly with the distance to the projection plane, with a scaling factor equal to e/d'. This is in agreement with a relationship derived by [Rolland et al. 95] for lateral eye movement. This also says that, for the cases where the registration error becomes very large (as in the case of nearly parallel projectors), the lateral error is bounded by this expression and all the rest of the error is depth error. We can now move on to a discussion of the binocular case which will enable us to verify this assertion.

5.6.4. Binocular Viewing Error

While the analysis of angular and lateral error just completed can be done for a single eye, the model for depth error must include both eyes. Because the expression for depth error in its exact form is rather complex, this section will give a general treatment of viewing error in Cartesian coordinates in order to shed more light on the process. The next section will use the results from this section to give exact and approximate estimates of depth error.

The figure below shows the general case for binocular viewing.

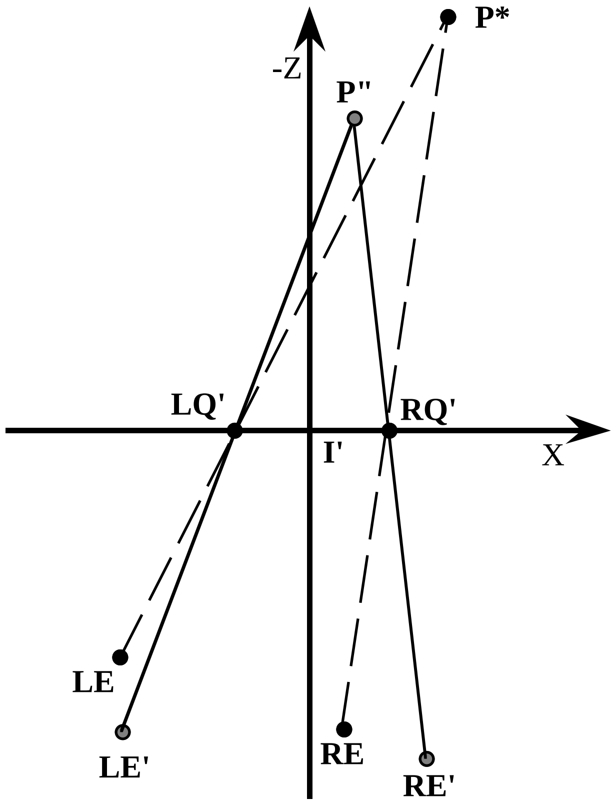


Figure 5.6.4.1. General case for binocular viewing error

As shown in the figure, P* is determined by the intersection of the projectors LE-LQ' and RE-RQ'. For simplicity, I have now redefined I' to be a combined image coordinate system with its origin midway between the modeled eyepoints.

The goal of this section is to derive the location of P* as a function of the real and modeled eyepoints and the coordinates of P".

To calculate P*, I begin by computing the projected points LQ' and RQ' using P" and LE' and RE'. From Section 4.5.1,

$$x_{LQ'} = \frac{x'' z_{LE'} - x_{LE''} z''}{z_{LE'} - z''} \tag{5.6.4.1}$$

$$y_{LQ'} = \frac{y'' z_{LE'} - y_{LE''} z''}{z_{LE'} - z''} \tag{5.6.4.2}$$

where all coordinates are measured relative to I'.

Section 4.5.2 has equations for the coordinates of the apparent point P* as a function of its projections and two eyepoints. In this case, the eyepoints are LE and RE rather than their modeled counterparts, LE' and RE', which is what leads to the discrepency between P'' and P*. Thus, I will annotate the equations from Section 4.5.2 to make them match the figure above:

$$x^* = \frac{x_{RQ'}\, z_{RE}\, (x_{LQ'} - x_{LE}) - x_{LQ'}\, z_{LE}\, (x_{RQ'} - x_{RE})}{z_{RE}\, (x_{LQ'} - x_{LE}) - z_{LE}\, (x_{RQ'} - x_{RE})} \tag{5.6.4.3}$$

$$y^* = y_{LQ'} + \frac{z_{RE}\, (x_{LQ'} - x_{RQ'})(y_{LE} - y_{LQ'})}{z_{RE}\, (x_{LQ'} - x_{LE}) - z_{LE}\, (x_{RQ'} - x_{RE})} \tag{5.6.4.4}$$

$$z^* = \frac{z_{LE}\, z_{RE}\, (x_{LQ'} - x_{RQ'})}{z_{RE}\, (x_{LQ'} - x_{LE}) - z_{LE}\, (x_{RQ'} - x_{RE})} \tag{5.6.4.5}$$

Substituting the expressions for $x_{LQ'}$ and $y_{LQ'}$ into this set of equations gives us the coordinates of P* in terms of the coordinates of P'' and the real and modeled eyepoints. While this set of equations is complete and exact, it is not particularly useful due to its complexity. In order to get any insight into the nature of viewing depth error I will have to make some simplifying assumptions.

A common assumption in HMD systems is that the line joining the eyepoints (hereafter called the *interpupillary line*) is parallel to the X axis of *I'*; *i.e.*,

$$z_{LE'} = z_{RE'} = d' \tag{5.6.4.6}$$

Since *I'* is centered on the modeled eyepoints,

$$x_{LE'} = -i', \quad x_{RE'} = i' \tag{5.6.4.7a}$$

and

$$y_{LE'} = y_{RE'} = 0 \tag{5.6.4.7b}$$

where *i'* is one-half of the modeled interpupillary distance. Furthermore, if we assume that the eyes remain parallel to the screen image as they move about (*i.e.*, the user does not cock his head within the HMD relative to the screen images), we have

$$y_{LE} = y_{RE} \tag{5.6.4.8a}$$

$$z_{LE} = z_{RE} = d \tag{5.6.4.8b}$$

The X coordinates of the real eyepoints can be expressed in terms of the real IPD:

$$x_{RE} - x_{LE} = 2i \tag{5.6.4.9}$$

where *i* is one-half of the real IPD.

Substituting these expressions into the expressions for x*, y*, and z*, we have

$$x^* = \frac{x'' + \frac{z''}{d'}\frac{i'}{i}\left(\frac{x_{LE}+x_{RE}}{2}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.10}$$

$$y^* = \frac{y'' + \frac{z''}{d'}\frac{i'}{i}y_{LE}}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.11}$$

$$z^* = \frac{z''\left(\frac{d}{d'}\right)\left(\frac{i'}{i}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.12}$$

This allows us to examine the magnitude of the error vector between P* and P". This is just

$$b_{view} = \|\mathbf{v}_{P''\_P*}\| = \|\mathbf{v}_{I\_P*} - \mathbf{v}_{I\_P''}\| = \sqrt{(x^*-x'')^2 + (y^*-y'')^2 + (z^*-z'')^2} \tag{5.6.4.13}$$

where b_{view} is a shorthand notation for the net registration error due to viewing error. After some algebraic manipulation we have

$$b_{view} = \frac{\sqrt{\left(\frac{i'}{i}\right)^2(x_m^2+y_{LE}^2+(d-\frac{i'}{i}d')^2) + \left(\frac{i'}{i}-1\right)^2(x''^2+y''^2+z''^2) - 2\frac{i'}{i}\left(\frac{i'}{i}-1\right)(x_m x''+y_{LE}y''+z''(d-\frac{i'}{i}d'))}}{\frac{d'}{z''} + \left(\frac{i'}{i} - 1\right)}$$

(5.6.4.14)

where $x_m = \left(\frac{x_{LE}+x_{RE}}{2}\right)$ is the X coordinate of the midpoint of the line segment between LE and RE.

This can be expressed in terms of vector magnitudes as follows:

$$b_{view} = \frac{\frac{|z''|}{d'}\sqrt{\left(\frac{i'}{i}\right)^2\|\mathbf{v}_\varepsilon\|^2 + \left(\frac{i'}{i} - 1\right)^2\|\mathbf{v}_{I\_P''}\|^2 - 2\frac{i'}{i}\left(\frac{i'}{i} - 1\right)(\mathbf{v}_\varepsilon \cdot \mathbf{v}_{I\_P''})}}{1 + \frac{|z''|}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.15}$$

where $\mathbf{v}_{I\_P''} = (x'', y'', z'')$ and $\mathbf{v}_\varepsilon = (x_m, y_{LE}, d - \frac{i'}{i}d')$. $\mathbf{v}_\varepsilon$ is a new vector that gives a measure of the deviation of the real eyepoints from the modeled eyepoints. Specifically, x_m indicates how far the eyes' midpoint has moved; *i.e.*, $x_m \neq 0$ when both eyes have shifted to one side or the other. Similarly y_{LE} is a measure of how much the eyes have shifted vertically from their modeled locations, since the y coordinates of the modeled eyepoints are equal to zero. The Z component is more complex because it accounts for both fore-aft movement (d ≠ d') *and* errors in modeling the IPD (i ≠ i'). All of these terms are zero when the eyes are in their modeled locations.

Using the law of cosines, this can be re-expressed as the magnitude of the difference of two vectors multiplied by scalars:

$$b_{view} = \frac{\frac{|z''|}{d'} \left\| \left(\frac{i'}{i}\right)\mathbf{v}_\varepsilon - \left(\frac{i'}{i} - 1\right)\mathbf{v}_{I\_P''} \right\|}{1 + \frac{|z''|}{d'}\left(\frac{i'}{i} - 1\right)} \qquad (5.6.4.16)$$

All of these expressions are valid as long as the interpupillary line is parallel to the X axis for both the modeled and real eyepoints. Thus, the real IPD can be different from the modeled IPD, and the real interpupillary line can be translated (but not rotated) relative to the modeled interpupillary line. This model can therefore describe errors in modeling the IPD, errors in fore-aft, left-right, and up-down placement of the user's eyes, which covers what I expect are the most common deviations from the modeled eyepoints. (Of course, those deviations not covered in this model can be examined by the full model above.)

Clearly, the denominator in any of these expressions can become zero for appropriate values of i' and i, leading to theoretically infinite viewing errors. It is this behavior that motivated the development of the lateral/depth error metric in the first place.

If there is only IPD error, $x_m = 0$, $y_{LE} = 0$, d = d', and

$$b_{view} = \frac{\sqrt{x''^2 + y''^2 + (d' - z'')^2}}{1 + \frac{d'}{|z''|\left(\frac{i'}{i} - 1\right)}} \qquad (5.6.4.16b)$$

The numerator is the distance from the midpoint between the eyes to P''; the denominator goes to infinity to give zero error when i' = i, or can go to zero for suitable values of *i', i, d',* and *z''*.

It is also interesting to examine what happens when the IPD is modeled correctly; i.e., i = i', but the eyepoints are translated as a *rigid pair* relative to the modeled eyepoints. This is what happens when the head shifts inside the STHMD or when one translates one's head when viewing a stereo image on a fixed display. In this case,

$$b_{view} = \frac{|z''|}{d'} \|\mathbf{v}_\varepsilon\| \qquad (5.6.4.17)$$

where

$$\|\mathbf{v}_{\varepsilon}\| = \sqrt{x_m^2 + y_{LE}^2 + (d-d')^2} \qquad (5.6.4.18)$$

which is exactly the distance between the real and modeled eyepoints (since the IPD is correct and fixed, the only error is a 3D translation which is the same for both eyes). If we restrict the eye movement to a sphere of uncertainty as before, *e* can be used as the input error bound to get

$$b_{view} = \frac{|z''|}{d'} e \qquad (5.6.4.19)$$

(this is similar to an expression derived by [Rolland et al. 95] for rigid-pair eye motion).

This expression indicates that for rigid-pair displacements of magnitude *e*, the error is only a function of the amount of displacement, the screen-to-point distance, and the modeled distance to the projection plane. The error in this case is fairly well behaved in comparison to the more general expression that allowed IPD error. While this error scales with z", it is only infinite for infinitely distant points. Note also that this expression is the same as the approximate bound for monocular lateral viewing error in Section 5.6.3.

While the error estimates derived here in Cartesian coordinates are exact and useful in some cases, we have seen that any expression involving IPD error can yield infinite error bounds, which are not very useful. Since the $\|\mathbf{v}_{P''\_P*}\|$ error bound combines the lateral and depth error and since we know that the lateral error is typically well-behaved, we expect that the sensitive behavior seen here will also show up in the expressions for depth error, to be derived next.

5.6.5. Depth Viewing Error

As discussed in Section 5.1, the depth error for a given eye is just absolute value of the difference in the distances from the eyepoint to the real and apparent points:

$$t_{view} = \big| \|\mathbf{v}_{E\_P*}\| - \|\mathbf{v}_{E\_P}\| \big| \qquad (5.1.1)$$

For the left eye,

$$t_{LE} = \big| \|\mathbf{v}_{LE\_P*}\| - \|\mathbf{v}_{LE\_P''}\| \big| \qquad (5.6.5.1)$$

which expands to

$$t_{LE} = \left| \sqrt{(x^*-x_{LE})^2 + (y^*-y_{LE})^2 + (z^*-z_{LE})^2} - \sqrt{(x''-x_{LE})^2 + (y''-y_{LE})^2 + (z''-z_{LE})^2} \right| \qquad (5.6.5.2)$$

Expanding x*, y*, and z* in terms of their full definitions (Equations 5.6.4.3-5) offers little hope of insight, so we can make the same limiting assumptions as in the previous section in order to obtain a more useful expression for t_{LE}. Plugging in Equations 5.6.4.10-12 and simplifying yields

$$t_{LE} = \left| \frac{\sqrt{\left[x''+\frac{z''}{d'}i'+\left(\frac{z''}{d'}-1\right)x_{LE}\right]^2 + \left[y''-\left(\frac{z''}{d'}-1\right)y_{LE}\right]^2 + \left[d\left(\frac{z''}{d'}-1\right)\right]^2}}{\left|1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)\right|} - \sqrt{(x''-x_{LE})^2 + (y''-y_{LE})^2 + (z''-d)^2} \right| \quad (5.6.5.3)$$

Although this equation is still rather complex, it does indicate that the depth error can grow without bound as the denominator of the first term approaches zero due to errors in modeling the IPD. If the IPD is modeled correctly (i = i'), then this simplifies slightly to

$$t_{LE} = \left| \sqrt{\left[x''-x_m+\frac{z''}{d'}x_m\right]^2 + \left[y''-y_{LE}+\frac{z''}{d'}y_{LE}\right]^2 + \left[\frac{d}{d'}(z''-d')^2\right]} - \sqrt{(x''-x_{LE})^2 + (y''-y_{LE})^2 + (z''-d)^2} \right| \quad (5.6.5.4)$$

As we saw in Section 5.6.4, because the overall error b_{view} is bound by $\frac{|z''|}{d'}$ e, we know that the depth error in this case is no greater than the total error and is thus bounded by the same expression.

5.7. Display Error

The next part of within-HMD error analysis (and the final part of the error model) examines the registration error made in displaying the virtual points. *Display* is defined here as the process of turning the coordinates of the point P" into a luminous image at the point Q within the 3D field of view of the user (Q is the actual displayed point and Q' is the modeled displayed point). Display error, then, is the registration error caused by errors made in the entire process of displaying the point, which covers the rendering of the model of the virtual object into the frame buffers, the scanout of the two images onto the screens in the STHMD, misalignments of the screen images, and aberrations in the virtual images of the screens. The model developed here is not meant to be a complete explanation of all aspects of the display process. In particular, rendering subtasks (such as lighting, clipping, and visible surface determination) that do not affect the location of a displayed point have been omitted.

5.7.1. Overview of Display Error

There are two parts to the analysis of display error: 1. What are the individual error sources that contribute to the net display error? 2. How do these error sources affect the net registration error?

The individual error sources can be divided into two categories: 1. Errors that displace the displayed point within the plane of the actual displayed image, and 2. Errors that move the displayed image relative to the modeled image plane. The figure below shows this graphically. Because there is interaction between viewing error and display error, the derivations that follow will assume that some amount of viewing error is present.

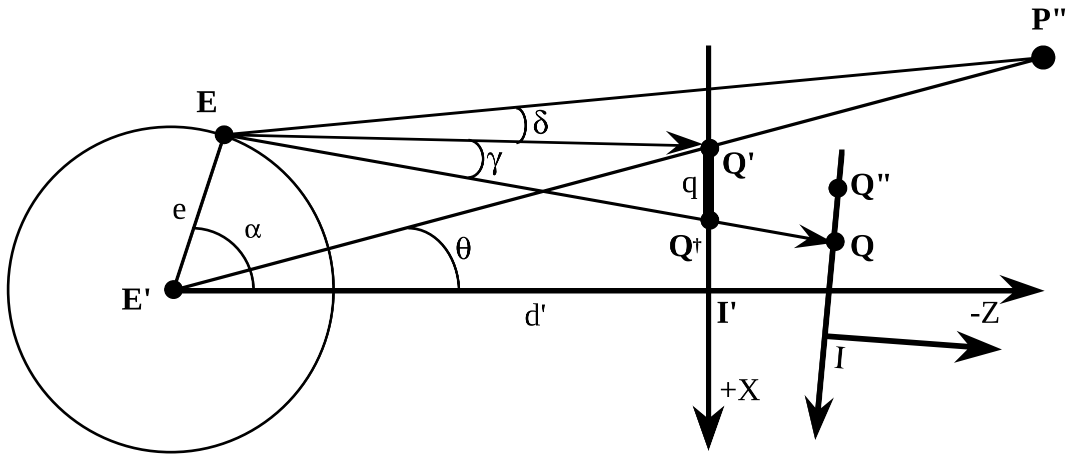


Figure 5.7.1.1. Display error

I' is the modeled Image CS and is the reference for all derivations. *I* is the actual Image CS and deviates from I' due to calibration and manufacturing errors. Based on the modeled projection plane and eyepoint, we project P" onto the X-Y plane of I' to get the vector $\mathbf{v}_{I'\_Q'}$, which defines Q'. We then tell the system to display $\mathbf{v}_{I'\_Q'}$ in the actual Image CS (I). If this were done perfectly, we would get the point Q", but because of within-image warping errors, the final displayed point ends up at Q. Q† is the projection of Q onto the modeled projection plane for the actual eyepoint E. The angular display error is specified by $\gamma = \angle Q'EQ$. Since the ray EQ† specifies the same projector as EQ, we can simplify the analysis by specifying all of the display error in the modeled projection plane via the vector $\mathbf{v}_{Q'\_Q^\dagger}$.

Because $\|\mathbf{v}_{Q'\_Q^\dagger}\|$ is such an important quantity, I define

$$q \equiv \|\mathbf{v}_{Q'\_Q^\dagger}\| \tag{5.7.1.1}$$

as a convenient shorthand notation. The following sections will analyze the net registration error (angular, lateral, etc.) for a given net display error of size *q*, and then will analyze the error sources that contribute to *q* and how they interact.

5.7.2. Angular Display Error

As was done for viewing error, I will begin by examining the monocular angular display error and will consider the binocular case later. Using an approach similar to the dot-product derivation used for viewing error, we have

$$\mathbf{v}_{E\_Q'} \cdot \mathbf{v}_{E\_Q\dagger} = \|\mathbf{v}_{E\_Q'}\| \cdot \|\mathbf{v}_{E\_Q\dagger}\| \cos\gamma \tag{5.7.2.1}$$

where

$$\mathbf{v}_{E\_Q'} = \mathbf{v}_{I'\_Q'} - \mathbf{v}_{I'\_E} = (x_{Q'} - x_e,\ z_{Q'} - z_e) \tag{5.7.2.2}$$

$$\mathbf{v}_{E\_Q\dagger} = \mathbf{v}_{I'\_Q\dagger} - \mathbf{v}_{I'\_E} = (x_{Q\dagger} - x_e,\ z_{Q\dagger} - z_e) \tag{5.7.2.3}$$

$$x_e = -e\sin\alpha \qquad z_e = d' - e\cos\alpha \tag{5.7.2.4}$$

$$x_{Q'} = \frac{x''}{1-\frac{z''}{d'}} \ ; \qquad z_{Q'} = 0 \tag{5.7.2.5}$$

$$x_{Q\dagger} = x_{Q'} + x_{Q'\_Q\dagger} \ ; \quad z_{Q\dagger} = 0 \tag{5.7.2.6}$$

where I have dropped the *I'* from the *x* and *z* component subscripts for simplicity. The Y components of $\mathbf{v}_{I'\_Q'}$ and $\mathbf{v}_{I'\_Q\dagger}$ are zero because I have chosen the coordinate system so that Q'EQ† lies in the X-Z plane (as was done for viewing error). In this case,

$$x_{Q'\_Q\dagger} = \pm q \tag{5.7.2.7}$$

which leads to

$$x_{Q\dagger} = x_{Q'} + x_{Q'\_Q\dagger} = \frac{x''}{1-\frac{z''}{d'}} \pm q \tag{5.7.2.8}$$

Combining these equations and simplifying in the same way as was done for viewing error, we arrive at

$$\gamma = \tan^{-1}\left(\left| \frac{q\, z_e}{z_e^2 + (x_{Q'} - x_e)^2 \pm q(x_{Q'} - x_e)} \right| \right) \tag{5.7.2.9}$$

which is valid for the range $0 \le \gamma \le 90^\circ$.

Searching analytically for the maximum of this function over all α and θ (the direction to P") fared no better than the search for the viewing-error angle's maximum. As we did for viewing error, we can plot γ as a function of α and θ to examine the function's behavior for typical parameter values. The graph below shows the plot for a typical case where z" = 500 mm, e = 10 mm, q = 10 mm.

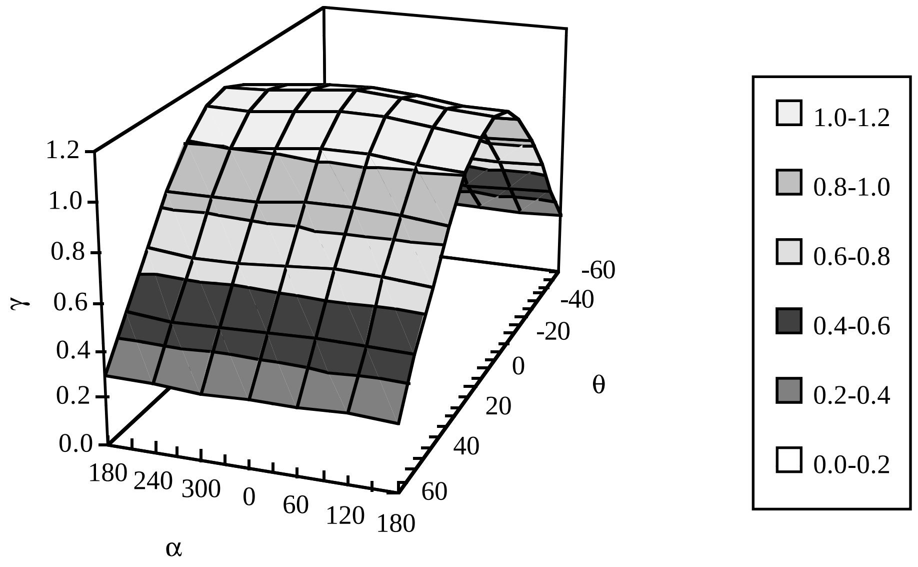


Figure 5.7.2.1. Angular display error γ as a function of θ and α; all angles in degrees.

Clearly, the maximum display error in this case occurs when θ is near 0; *i.e.*, for on-axis or near-on-axis points. α seems to have little effect in this case, although the high point of the surface is near α = 0. If the errors are exaggerated somewhat, we see this behavior more clearly. The next plot is for the situation depicted in Figure 5.7.1.1, in which the errors were deliberately increased to show the geometric relationships.

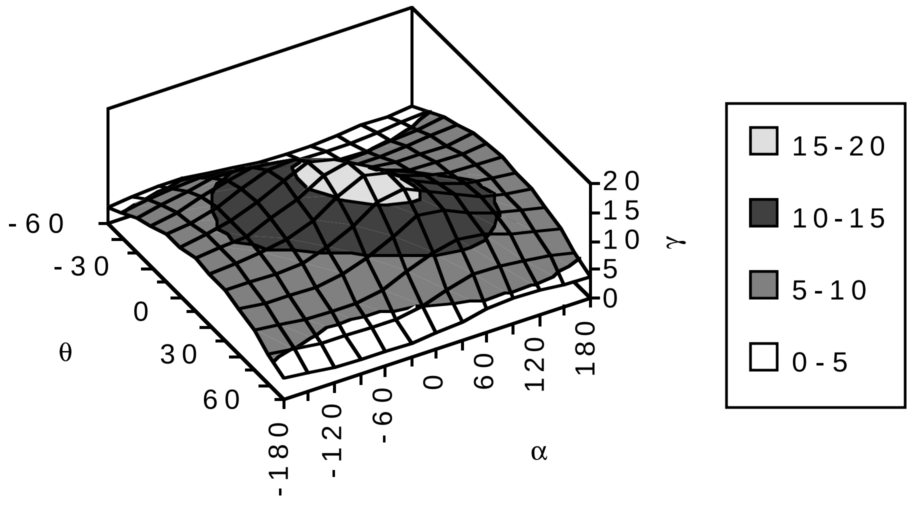


Figure 5.7.2.2. Angular display error for exaggerated case

In this case, e = 22, q = 15, and d' = 71. This is more error than one would ever expect to see in a real system, but it shows clearly how γ reaches its peak for both α and θ near zero. This is a fairly intuitive result: γ is at its maximum value when the eyepoint is closest to the midpoint of the segment Q'Q†, which occurs for nearly-on-axis points[17] with α = 0. This remains true as we vary *q*, *d'*, and *e* through typical values. When α = 0, we see that $x_e = 0$ and $z_e = d' - e$; when θ = 0, $x_{Q'} = 0$, and Equation 5.7.2.9 reduces to

$$\gamma_{wc} = \tan^{-1}\left|\frac{q}{d'-e}\right| \qquad (5.7.2.10)$$

The graphs above also reveal something about the interaction between viewing error and display error. The fact that the first plot of γ (Figure 5.7.2.1) does not vary significantly with α is an indication that when *e* is small with respect to *d'*, the effect of eye movement on display error is minimal. When this is the case (as it usually is for this application)[18], the sphere of uncertainty around E' is far enough from the screen image that it looks like a point and can be approximated as such. (This is not the same as having zero viewing error;

[17] Specifically, the angle γ is a maximum when the bisector of Q'Q† passes through E. This distinction is not important for small angles, however.

[18] An analysis of the variation of γ with *e* showed that display error varies only 8-9% for *e* values as high as 2cm, which is more than we expect to see in this application.

the result here is that the variation of the display error due to the viewing error is often negligible even in the presence of non-negligible viewing error.)

When *e* is small compared to *d'*, Equation 5.7.2.9 reduces to

$$\gamma = \tan^{-1}\left(\left|\frac{q\,d'}{d'^2 + x_{Q'}^2 \pm q x_{Q'}}\right|\right) \tag{5.7.2.11}$$

For on-axis points, $\theta = x_{Q'} = 0$, and

$$\gamma = \tan^{-1}\left|\frac{q}{d'}\right| \tag{5.7.2.12}$$

which is the same as the worst-case expression for γ when $e \ll d'$.

Equations 5.7.2.10-12 show that for small angles (where $\tan\gamma \approx \gamma$), the angular error increases linearly as *q* increases. One might expect that increasing *d'* would scale down this error, but as we shall see, larger screen-image distances typically imply greater optical magnification, which will tend to increase *q* as well.

Finally, it should be clear from Figure 5.7.1.1 that the angular viewing error just adds to the angular display error in the worst case; i.e.,

$$|\varnothing_{wc}| = |\delta_{wc}| + |\gamma_{wc}| \tag{5.7.2.13}$$

5.7.3. Lateral Display Error

Now that I have derived a number of expressions for the angular display error, I turn to the problem of deriving an expression for the lateral display error. As in the discussion of angular display error, one must take into account the effect of viewing error on the display error, as depicted below.

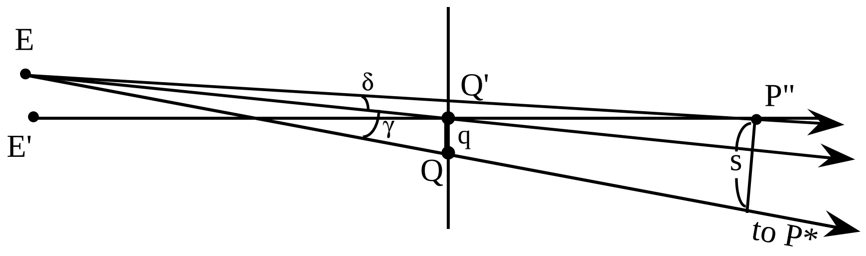


Figure 5.7.3.1. Lateral error due to viewing and display error

The lateral error here due to viewing and display error is given by

$$s_{v\&d} = 2r\sin\left(\frac{\delta+\gamma}{2}\right) \qquad (5.7.3.1)$$

Expanding the full expressions for δ and γ in terms of arctangent yields an exact but very complex function. A simpler approximation to this full expression can be derived by assuming that δ and γ are small so that

$$2r\sin\left(\frac{\delta+\gamma}{2}\right) \approx r(\delta + \gamma) \approx r\sin\delta + r\sin\gamma \qquad (5.7.3.2)$$

If we take the worst-case expression for δ where x" = 0 and α = 90˚, and assume that *e << d'*, we can use Equation 5.6.3.3 for s_{view} in place of r sinδ. In this case, the fact that α is not at its worst-case value for γ will have a negligible effect on the display error. The full expression for sinγ is

$$\sin\gamma = \frac{q\,z_e}{\sqrt{\left[(x_{Q'}-x_e)^2 + z_e{}^2\right]\left[(x_{Q'}-x_e\pm q)^2 + z_e{}^2\right]}} \qquad (5.7.3.3)$$

If we use x" = 0, α = 90˚, the exact expression for *r* is

$$r = \sqrt{e^2 + (d'-z")^2} \qquad (5.7.3.4)$$

Since *e*^{2} *<< (d'-z")*^{2} and if *q << d'*, then

$$r\sin\gamma \approx q\cdot\left(1 - \frac{z"}{d'}\right) \qquad (5.7.3.5)$$

Combining this with the approximate worst-case lateral viewing error, we get

$$s_{v\&d} \approx e\cdot\frac{|z"|}{d'} + q\cdot\left(1 - \frac{z"}{d'}\right) \qquad (5.7.3.6)$$

Thus, at least for small errors, the amount of lateral error is linear in both *e* and *q* and increases with *z"* for points on the far side of the screen image. The lateral display error actually decreases for points on the near side of the screen image, whereas the lateral viewing error increases in both directions as we move away from the projection plane.

In the case of display error alone, the expression is

$$s_{disp} \approx q\cdot\left(1 - \frac{z"}{d'}\right) \qquad (5.7.3.6)$$

5.7.4. Binocular Display Error

Just as I did for viewing error, I will analyze the binocular case for display error in order to derive the depth display error, as well as general Cartesian expressions for display error.

As in the previous two sections, one must take any viewing error into account, since this has some effect on the display error.

This derivation is a natural continuation of the previous two sections and the section on viewing error. The figure below shows the case for two eyes.

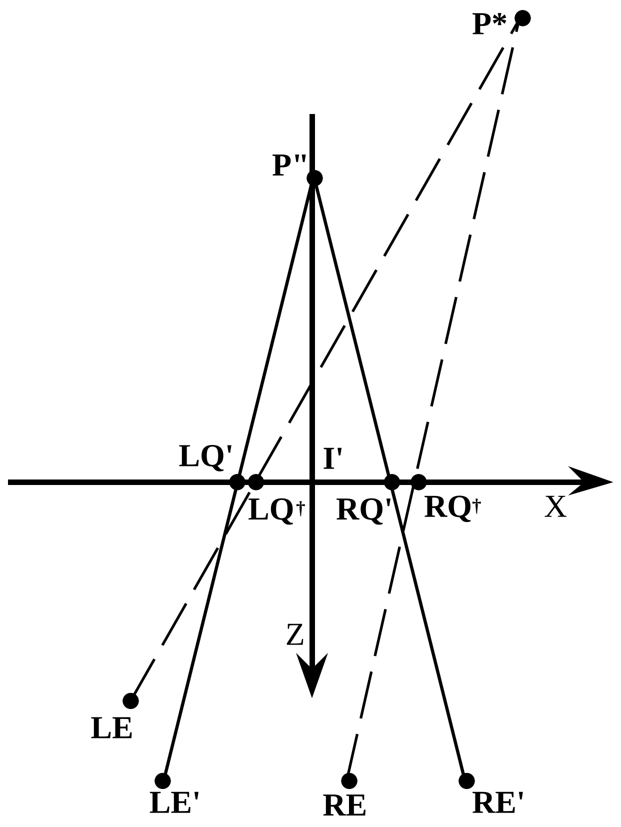


Figure 5.7.4.1. Binocular viewing/display error

Here, neither the eyes nor the projected points end up in their modeled locations, leading to the perceived point P*, which deviates from P" due to both types of error. In a similar fashion to what was done for viewing error, I define a unified coordinate system (I'), and then define the coordinates of P* in terms of the real eyepoints and the projected points. The difference is that here, LQ ≠ LQ' and RQ ≠ RQ', so we cannot use the modeled projected points any more. Instead we must use the projections of LQ and RQ onto the X-Y plane of *I'*, LQ† and RQ†. This gives the following definitions for the true projected points:

$$x_{LQ^\dagger} = x_{LQ'} + x_{Q'\_Q^\dagger} = x_{LQ'} + q_{LX} = \frac{x'' z_{LE'} - x_{LE''} z''}{z_{LE'} - z''} + q_{LX} \qquad (5.7.4.1)$$

$$y_{LQ\dagger} = y_{LQ'} + y_{Q'\_Q\dagger} = y_{LQ'} + q_{LY} = \frac{y''\, z_{LE'} - y_{LE'} z''}{z_{LE'} - z''} + q_{LY} \tag{5.7.4.2}$$

where q_{LX} and q_{LY} are the components of the 2D error vector $\mathbf{v}_{LQ'\_LQ\dagger}$ in the plane of projection (the equations for the right eye are analogous). We can now use the equations for x*, y*, and z* used in Section 5.6.4, except that we will now use LQ† and RQ† instead of LQ' and RQ':

$$x^* = \frac{x_{RQ\dagger}\, z_{RE}\,(x_{LQ\dagger} - x_{LE}) - x_{LQ\dagger}\, z_{LE}\,(x_{RQ\dagger} - x_{RE})}{z_{RE}\,(x_{LQ\dagger} - x_{LE}) - z_{LE}\,(x_{RQ\dagger} - x_{RE})} \tag{5.7.4.5}$$

$$y^* = y_{LQ\dagger} + \frac{z_{RE}\,(x_{LQ\dagger} - x_{RQ\dagger})(y_{LE} - y_{LQ\dagger})}{z_{RE}\,(x_{LQ\dagger} - x_{LE}) - z_{LE}\,(x_{RQ\dagger} - x_{RE})} \tag{5.7.4.6}$$

$$z^* = \frac{z_{LE}\, z_{RE}\,(x_{LQ\dagger} - x_{RQ\dagger})}{z_{RE}\,(x_{LQ\dagger} - x_{LE}) - z_{LE}\,(x_{RQ\dagger} - x_{RE})} \tag{5.7.4.7}$$

As with viewing error, to get a complete and accurate specification for P*, we need only substitute the expressions for $x_{LQ\dagger}$ and $y_{LQ\dagger}$ into the above three equations. And as before, because the resulting equations are so complex, it behooves us to make some simplifying assumptions in order to get a better understanding of the behavior of the combined viewing/display error.

We can use the same set of restrictions used for viewing error, namely,

$$x_{LE'} = -i',\ x_{RE'} = i' \quad \text{and} \quad x_{RE} - x_{LE} = 2i \tag{5.7.4.8}$$

$$y_{LE'} = y_{RE'} = 0 \quad \text{and} \quad y_{LE} = y_{RE} \tag{5.7.4.9}$$

$$z_{LE'} = z_{RE'} = d' \quad \text{and} \quad z_{LE} = z_{RE} = d \tag{5.7.4.10}$$

That is, we examine the case in which we have defined our coordinate system so that the modeled eyepoints are centered on it and in which the interpupillary line remains parallel to the screen. In this case, we have

$$x^* = \frac{x'' + \frac{i'z''}{i\,d'}\, x_m + \left(\frac{x_{RE}\, q_{LX} - x_{LE}\, q_{RX}}{2i}\right)\left(1 - \frac{z''}{d'}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right) + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.11}$$

$$y^* = \frac{y'' + \frac{i'z''}{i\,d'}\, y_{LE} + \left(q_{LY} + \frac{\Delta q_X}{2i} y_{LE}\right)\left(1 - \frac{z''}{d'}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right) + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.12}$$

$$z^* = \frac{z''\frac{i'd}{i\,d'} + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right) + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.13}$$

where

$$x_m = \frac{x_{LE} + x_{RE}}{2} \tag{5.7.4.14}$$

$$\Delta q_X = q_{LX} - q_{RX} \tag{5.7.4.15}$$

Since viewing error has been accounted for, the expressions for x*, y*, and z* reduce to those for viewing error (Eqns 5.6.4.10-12) when the *q* terms go to zero, as one would expect.

In the case of pure display error (no viewing error), i = i', $x_m = 0$, $x_{RE} = -x_{LE} = i$, $y_{LE} = 0$, d = d', and Equations 5.7.4.11-13 reduce to

$$x^* = \frac{x'' + q_{mx}\left(1 - \frac{z''}{d'}\right)}{1 + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.16}$$

$$y^* = \frac{y'' + q_{LY}\left(1 - \frac{z''}{d'}\right)}{1 + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.17}$$

$$z^* = \frac{z'' + \frac{\Delta q_X}{2i}(d' - z'')}{1 + \frac{\Delta q_X}{2i}\left(1 - \frac{z''}{d'}\right)} \tag{5.7.4.18}$$

where

$$q_{mx} = \frac{q_{LX} + q_{RX}}{2} \tag{5.7.4.19}$$

Finally, if we use these equations to compute the vector $\mathbf{v}_{P''\_P*} = \mathbf{v}_{I\_P*} - \mathbf{v}_{I\_P''}$, we can express the overall registration error magnitude due to pure display error:

$$x^* - x'' = \frac{q_{mx} - x''\frac{\Delta q_X}{2i}}{\frac{d'}{d' - z''} + \frac{\Delta q_X}{2i}} \tag{5.7.4.20}$$

$$y^* - y'' = \frac{q_{LY} - y''\frac{\Delta q_X}{2i}}{\frac{d'}{d' - z''} + \frac{\Delta q_X}{2i}} \tag{5.7.4.21}$$

$$z^* - z'' = \frac{d'\frac{\Delta q_X}{2i} - z''\frac{\Delta q_X}{2i}}{\frac{d'}{d' - z''} + \frac{\Delta q_X}{2i}} \tag{5.7.4.22}$$

which can be expressed as a difference of two vectors:

$$\mathbf{v}_{P''\_P*} = \frac{1}{\frac{d'}{d'-z''} + \frac{\Delta q_x}{2i}} \left(\mathbf{v}_{de} - \frac{\Delta q_x}{2i}\mathbf{v}_{I\_P''}\right) \tag{5.7.4.23}$$

where the new vector $\mathbf{v}_{de}$ is defined as

$$\mathbf{v}_{de} = \left(q_{mx},\ q_{LY},\ d'\frac{\Delta q_x}{2i}\right) \tag{5.7.4.24}$$

This vector gives an indication of how far the projected points have deviated from their modeled locations (subject to the assumptions already stated) and is similar to the vector $\mathbf{v}_e$ defined for viewing error and gives a particularly compact expression for pure display error. The magnitude of the registration error vector is given by the law of cosines:

$$b_{disp} = \|\mathbf{v}_{P''\_P*}\| = \left|\frac{1}{\frac{d'}{d'-z''} + \frac{\Delta q_x}{2i}}\right| \sqrt{\|\mathbf{v}_{de}\|^2 + \left(\frac{\Delta q_x}{2i}\right)^2 \|\mathbf{v}_{I\_P''}\|^2 - 2\,\frac{\Delta q_x}{2i}\mathbf{v}_{de}\cdot\mathbf{v}_{I\_P''}} \tag{5.7.4.25}$$

5.7.5. Display Depth Error

As with viewing error, the depth error for a given eye is just the absolute value of the difference in the distances from the eyepoint to the real and apparent points. For the left eye,

$$t_{LE} = \left|\,\|\mathbf{v}_{LE\_P*}\| - \|\mathbf{v}_{LE\_P''}\|\,\right| \tag{5.7.5.1}$$

which expands to

$$t_{LE} = \left|\sqrt{(x^*-x_{LE})^2 + (y^*-y_{LE})^2 + (z^*-z_{LE})^2} - \sqrt{(x''-x_{LE})^2 + (y''-y_{LE})^2 + (z''-z_{LE})^2}\,\right| \tag{5.7.5.2}$$

As before, the exact expression for the general case is too complex to be of much use for understanding the behavior of the display depth error. The expression for the case of pure display error treated at the end of the previous section gives some insight:

$$t_{LE} = \left|\ \frac{\left|\sqrt{[x''-x_{LE}+ß(q_{mx}-f x_{LE})]^2+[y''-y_{LE} + ß(q_{LY}-f y_{LE})]^2+[z''-z_{LE} + f ß(d'-z_{LE})]^2}\right|}{1 + \frac{\Delta q_x}{2i}\left(1-\frac{z''}{d'}\right)} - \left|\sqrt{(x''-x_{LE})^2 + (y''-y_{LE})^2 + (z''-z_{LE})^2}\,\right|\ \right| \tag{5.7.5.3}$$

where $f = \frac{\Delta q_x}{2i}$ and $ß = 1 - \frac{z''}{d'}$.

The most important conclusion to be drawn from this rather complex expression is that the first term shows the same behavior as in the viewing depth error analysis; namely, the denominator can go to zero when errors in display cause the projectors to become parallel, and the depth error then tends toward infinity.

5.7.6. Overview of Display Error Sources

Up to this point, the display process has been quite abstract, and I have left out the details about the error sources that cause the actual projected point Q to deviate from the modeled projected point Q'. This section will analyze the error sources from the rendering into the frame buffers to the formation of the virtual images for each eye.

As stated in Section 5.7.1, the display error sources can be divided into two categories: within-image-plane errors and deviations of the image plane from its modeled location. Clearly, any translation or rotation of the image plane will affect all points within that plane, so I will begin by examining the effect of such misalignments of the images. Next, I will analyze the screen-space errors associated with the display devices, and finally I will turn to the imaging errors associated with the optics of the STHMD.

5.7.7. Misalignment of the Virtual Images

The location and orientation of the virtual images is expressed via the Sensor-Image transformations $T_{S\_LI'}$ and $T_{S\_RI'}$. These values for these transformations are typically derived via a calibration procedure and therefore contain some amount of measurement error. The goal of this section is to specify how much display error results from a given amount of rigid-body transformation error in specifying the locations of the screen images. For convenience, I will analyze this error relative to the modeled Image CS rather than the Sensor CS, since the relationship between S and I' can be treated as fixed and known.

The figure below shows the situation for one eye.

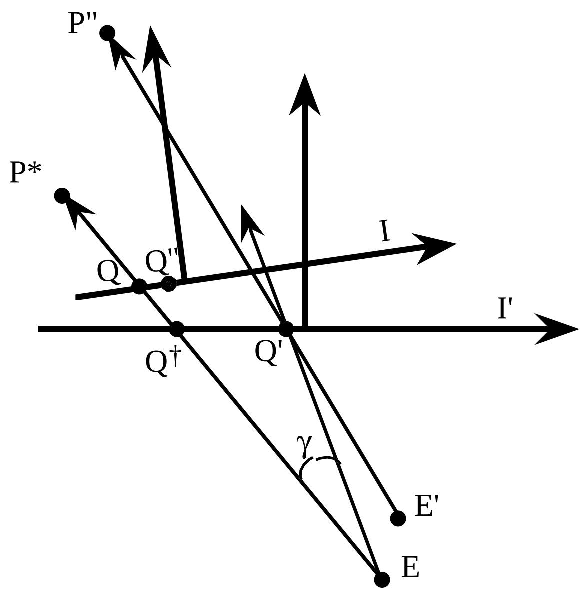


Figure 5.7.7.1. Screen-image misalignment

As we saw in Section 5.7.1, *I'* is the modeled Image CS, and the real Image CS, *I*, is shown translated and rotated relative to *I'*. The point Q' is the modeled projected point in *I'* and is described by the vector $\mathbf{v}_{I'\_Q'}$. If this vector were perfectly displayed in the *I* CS, it would yield the point Q"; however, because of within-image display errors, the final, displayed point ends up at Q. Finally, the projection of Q onto the X-Y plane of *I'* yields the point Q†. Thus, the vector between Q and Q" describes all of the within-projection-plane warpings already described, and the vector between Q' and Q† gives the net display error within the X-Y plane of *I'*. Note that if the misalignment of *I* relative to *I'* happens to be contained to the X-Y plane, Q is its own projection and Q† = Q.

The expression in terms of projection and rigid-body transformation operators is

$$q = \|\mathbf{v}_{Q'\_Q^\dagger}\| = \|\mathbf{v}_{I'\_Q^\dagger} - \mathbf{v}_{I'\_Q'}\| = \|T_{Proj} \cdot (T_{I'\_I} \cdot \mathbf{v}_{I\_Q}) - \mathbf{v}_{I'\_Q'}\| =$$
$$\|T_{Proj} \cdot T_{I'\_I} \cdot (\mathbf{v}_{I\_Q''} + \mathbf{v}_{Q''\_Q}) - \mathbf{v}_{I'\_Q'}\| \qquad (5.7.7.1)$$

where T_{Proj} (described in Chapter 4) gives the projection onto the X-Y plane of I' for the actual eyepoint E, $T_{I'\_I}$ is the rigid-body transformation that describes the displacement of the actual image, $\mathbf{v}_{I'\_Q'}$ (= $\mathbf{v}_{I\_Q''}$) is the modeled image-space vector, and $\mathbf{v}_{Q''\_Q}$ contains all of the within-image errors considered in the sections that follow.

The general expression for *q* is rather complex, so let us take a look at a simpler case in order to gain some insight into the nature of this error. If the misalignment is relatively small, *I* will be close enough to *I'* that the projection operation will have little effect, which simplifies the expression to

$$q \approx \|R_{I'\_I} \cdot (\mathbf{v}_{I'\_Q'} + \mathbf{v}_{Q''\_Q}) + \mathbf{v}_{I'\_I} - \mathbf{v}_{I'\_Q'}\| \tag{5.7.7.2}$$

where I have expressed $T_{I'\_I}$ as a rotation operation followed by a vector addition. This can be simplified to

$$q \approx \|R_{I'\_I} \cdot \mathbf{v}_{Q''\_Q} + \mathbf{v}_{I'\_I} + (R_{I'\_I} - I) \cdot \mathbf{v}_{I'\_Q'}\| \tag{5.7.7.3}$$

We encountered the last term in Chapter 4; it is just the "difference vector" $R\mathbf{v} - \mathbf{v}$ which we already know from Chapter 4 to be bounded by $|2 \sin\frac{Ø_{I'\_I}}{2}| \, \|\mathbf{v}_{I'\_Q'}\|$. The complete expression is therefore

$$q \approx \|\mathbf{v}_{Q''\_Q}\| + \|\mathbf{v}_{I'\_I}\| + 2 \cdot \left|\sin\frac{Ø_{I'\_I}}{2}\right| \cdot \|\mathbf{v}_{I'\_Q'}\| \tag{5.7.7.4}$$

To first order, then, the image misalignment error just adds rigid-body transformation error to the existing error sources, as might be expected. The projection operator will minimize the effects of movements perpendicular to the projection plane (Z translation, X/Y axis rotation), but will have little effect on movement within the image plane (X-Y translation or rotation about Z). Thus, within-image-plane rotations and translations will be more significant than movements perpendicular to the image plane. I next address the within-image display errors that determine $\|\mathbf{v}_{Q''\_Q}\|$, beginning with the screen-space errors.

5.7.8. Screen-Space Errors

This group of error sources consists of any imperfections in the conversion of the geometrical model expressed in Image space into an image on the screen of the display device. These error sources will generally be very technology-specific in their nature and their magnitude. Errors considered in this section are limited frame-buffer resolution, limited resolution of the display device itself (especially a factor with LCDs), refraction by the faceplate glass of a CRT, curvature of the CRT screen, and non-linearities in the CRT drive electronics. Each of these is considered as a potential error source.

In this section, I consider the display of the points prior to imaging by the optics, which will magnify any errors introduced here and will typically further distort the image. For this reason, our test point in display-device coordinates will be called G (or G') in order to differentiate it from the image-plane point Q (or Q'). The significance of the errors examined here can only be judged after they have been imaged by the optics, which I will address in the sections that follow.

5.7.8.1. Errors Due to Aliasing

The primitives making up the model are eventually scan-converted into a set of colored pixels in the frame buffer, which are then scanned out onto the display screen. Because both the frame buffer and the display screen have finite resolution, there is a fundamental problem in rendering a polygon precisely: Each pixel corresponds to a finite area on the screen, which means that the polygon edge will typically cross pixels in such a way that part of the pixel is inside the polygon and the other part is outside, as shown in the following figure.

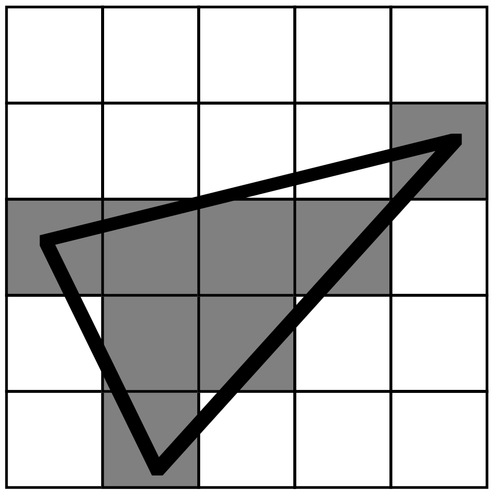


Figure 5.7.8.1.1. Partial coverage of pixels by a polygon

The implication for registration error is that primitives which represent the point P may be drawn shifted relative to their ideal locations. We are interested in any shift of the polygon's vertices, edges, and center of mass, since any of these might be used to judge the location of P. I begin with a brief discussion of sampling theory, and then derive an error bound for aliasing error.

According to sampling theory, a regularly spaced grid of sample points (such as a frame buffer) can only faithfully represent a signal or image whose highest frequency is one half of the sampling frequency [Castleman 79]. Polygons and other sharp-edged primitives contain infinitely high spatial frequencies and thus cannot be properly sampled with any finite grid. The result of point-sampling such primitives is called *aliasing*, in which the high-frequency components of the primitives that cannot be accurately represented show up as artifacts in the rendered image. *Antialiasing* is the name for a set of techniques for reducing these artifacts. Ideal antialiasing blurs the primitives prior to sampling in order to eliminate the high frequencies that the system cannot display correctly. This produces a blurrier but more faithful image.

The second phase of the process of displaying an image is known as *reconstruction.* Reconstruction is the process by which a continuous signal is recreated from the samples. This is done by the display hardware, which scans the sampled values out of the frame buffer and uses them to create an image on the screen. It is the reconstruction filter that determines the shape and size that each sample represents. This part of the process is very technology-specific, and will be treated later in this section. For now, we should consider the samples as having essentially no area, even though we may draw them as squares for simplicity.

Let us begin by examining the case for non-antialiased (point-sampled) primitives. First, it should be clear that we can come up with a tiny, serpentine polygon which the sample points miss almost completely and which is therefore rendered so poorly that the point represented by it is considerably shifted from its modeled location. Thus, in order to arrive at any reasonable error bound, we must make the assumption that our primitives are large with respect to the sampling grid spacing and have simple shapes. In this case, we can look at the edges of the polygon to derive a simple error bound for both edge features and the center of mass of the polygon.

Consider the case of a small, point-sampled square which marks the location of some landmark, as depicted below.

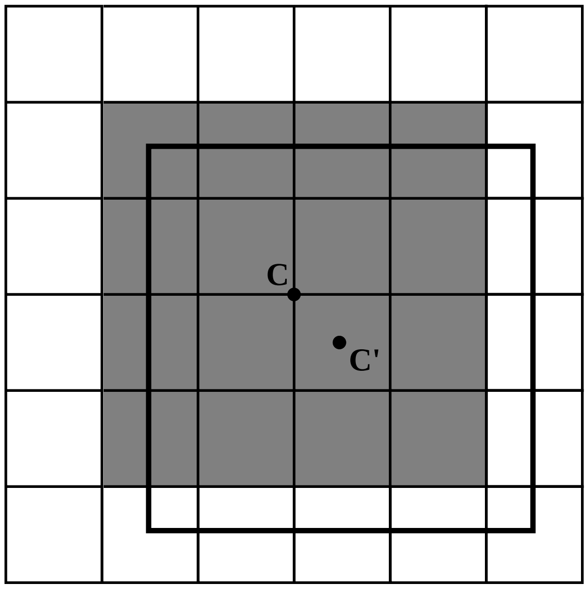


Figure 5.7.8.1.2. Shifting of a rendered marker square

Here, the heavy square is the boundary of the modeled square, and the shaded region is the pixels that are actually turned on using point sampling at pixel centers. (The resolution was deliberately reduced to make the error more obvious.) The point C' is the center of the modeled square, and the point C is the center of the rendered square. The two centers do not coincide because the square is offset slightly with respect to the sampling grid and some of the geometry falls between the sample points. Note that although we have drawn the samples as square pixels, the center of mass for the samples is computed using only the pixel-center locations and not the area around them.

It should be clear that the worst-case error for each pixel on a polygon edge is half the frame-buffer sampling distance; *i.e.,* one half a pixel in either direction. Edges and vertices will therefore be shifted by about a half-pixel in either direction, and in the worst case, all of the edges are shifted in the same manner (as in the figure above) and the center of mass is shifted by this amount as well.

We can now consider the case of antialiased primitives. In ideal antialiasing, the primitives are blurred, sampled, and then the blurred signal is perfectly reconstructed from the samples. The center of mass is not shifted at all in this case, since the blurring does not move the center of mass, and the sampling and reconstruction are done without error. The non-ideal antialiasing used in real systems may shift the center of mass somewhat, but appears to do no worse than point sampling. Thus, for simplicity and because this is a relatively small error source, I will use point-sampled primitives as the worst-case scenario.

Finally, we must consider the process of reconstruction. As mentioned above, the reconstruction process takes the discrete samples in the frame buffer and turns them into a continuous image on the screen. This amounts to convolving the samples in the frame buffer with one or more filters, which has the effect of blending them together into a more or less continuous image. The reconstruction process for CRTs effectively convolves the samples with a one-pixel-wide box filter (the scanout circuitry), followed by a Gaussian (the blur spot formed by the phosphor), all of which blurs the samples together, but should not displace the center of mass of the samples.

The reconstruction process for LCDs is manufacturer-specific, but preliminary experiments at UNC [Mine & Bishop 94] indicate that in addition to the increased blurring caused by the lower resolution, there may also be some resampling. For systems that resample the scanout signal, additional aliasing may be introduced, and the error bound should be increased to one half of the LCD pixel spacing instead of one half of the frame-buffer pixel spacing.

Analytically, then, we can express the aliasing error in display space as the amount by which a point G is displaced from its modeled location at G', the bound for which is

$$\|\mathbf{v}_{G'\_G}\| \leq \sqrt{\left(\frac{h_x}{2}\right)^2 + \left(\frac{h_y}{2}\right)^2} \tag{5.7.8.1.1}$$

where we have defined the pixel dimensions as

$$h_x = \frac{g_x}{p_x} \qquad h_y = \frac{g_y}{p_y} \tag{5.7.8.1.2}$$

where g_x and g_y are the size of display screen in millimeters, and p_x and p_y give the resolution of the frame buffer[19] (or the display screen in the case where the reconstruction process contains further sampling).

In the case of square pixels, $h_x = h_y = h$ and this reduces to

[19] It is worth noting here that the screen does not always display the entire frame-buffer image; that is, some pixels that are scanned out do not appear in the virtual image of the screen for various technology-specific reasons. This leads to scaling and/or translation errors, but is correctable in software. See [Rolland et al. 94] for details.

$$\|\mathbf{v}_{G'\_G}\| \leq \frac{\sqrt{2}}{2} h = \frac{\sqrt{2}}{2} \frac{g_x}{p_x} \qquad (5.7.8.1.3)$$

how much of a shift (in screen space) one can expect.

In the absence of other errors, the *q* term for aliasing is

$$q_{alias} = m \cdot \|\mathbf{v}_{G'\_G}\| = \frac{\sqrt{2}}{2} \cdot m \cdot \frac{g_x}{p_x} \qquad (5.7.8.1.4)$$

where *m* is the linear magnification of the optics.

Finally, it is worth noting that because of the user's head movement, the image of a given feature is constantly shifting around on the screen, and therefore the average error from aliasing should be around zero (or at least very small).

5.7.8.2. Display-Technology-Specific Errors

In addition to the problems caused by aliasing, there are display-technology-specific errors that cause the screen image to deviate from its ideal form. That is, given a specification of an image to be displayed on a screen of dimensions $g_x \times g_y$, one would ideally expect a planar image of exactly that size with evenly spaced pixels. The actual screen image, however, may deviate from this ideal in a number of technology-specific ways.

CRTs are prone to the following problems:

- Beam-deflection distortion: Depending on the type of deflection system used, a CRT may exhibit either pincushion or barrel distortion. The magnetic deflection systems which are prevalent today exhibit pincushion distortion because the current for the given deflection angle is a function of the sine of the angle while the deflection on the faceplate is a function of the tangent of the angle. Values for the distortion are usually given as a percentage of the screen size; 1% is a typical value [Allen 94][Scherr 82].
- Linearity errors: If we move a small vector across the CRT and check its length as a function of its location, we find that it changes as we move across the screen. This is a function of the accuracy of the deflection yoke. The resulting error bounds range from 0.1% to 15% depending on the size of the vector and the system in question [Allen 94][Scherr 82].
- Faceplate curvature: The faceplate of many CRTs is curved, rather than planar as the computer graphics model usually assumes. [Deering 92] explains this error in detail for a workstation screen. Fortunately, the faceplates of CRTs used in HMDs tend to be flat (inside and outside radius of curvature is infinite) due to their small size. The

1" CRT used in the field-sequential color system made by Tektronix is an example [Chi 93]. This error source will not be treated further in this research.

- Faceplate refraction: Because the image is drawn on the *inside* of the CRT, the light passing through the faceplate is refracted on its way through the rest of the optics of the STHMD. This error is also treated in [Deering 92]. For a STHMD, the faceplate glass can be treated as just another optical element, so any displacements of points due to this refraction will be lumped in with the optical distortion term to be discussed in the next section.

The last two problems need no further treatment here; the first two errors are often lumped together into an error term known as *absolute accuracy*.[20] This term gives the amount by which a display error in the CRT can cause a displayed point to deviate from its theoretical location expressed as a percentage of the screen width. Depending on the CRT, this number may be as low as 0.1% or as large as 3%.

If we use η to represent the absolute nonlinearity of the display device, then the maximum error in terms of G and G' (independent of the aliasing error discussed above) is

$$\|\mathbf{v}_{G'\_G}\| \leq \sqrt{(\eta\, g_x)^2 + (\eta\, g_y)^2} = \eta\sqrt{g_x{}^2 + g_y{}^2} \qquad (5.7.8.2.1)$$

In the absence of other errors, the *q* term for display device nonlinearity is

$$q_{\text{nonlin}} = m\cdot\|\mathbf{v}_{G'\_G}\| = m\cdot\eta\sqrt{g_x{}^2 + g_y{}^2} \qquad (5.7.8.2.2)$$

where *m* is the linear magnification of the optics.

5.7.9. Imaging Errors

The familiar thin-lens equations for spherical lenses[21] are valid for rays that intersect the lenses in the system at small heights and angles with respect to the optical axis. This *paraxial* model of optics is only a first-order approximation for real spherical optical systems. When finite apertures are used, rays are no longer constrained to be near the optical axis and we begin to see deviations from the first-order model which are called *aberrations*. There are two main classifications of aberrations: *chromatic aberrations,* which arise from the dependency of the index of refraction on the wavelength of the

---

[20] This is also sometimes called *positional accuracy* or *absolute non-linearity*.

[21] Although aspherical lenses do exist, their cost is usually prohibitive for most applications.

incident light, and *monochromatic aberrations,* which can be seen even with monochromatic light.

The five primary monochromatic aberrations are

1. *Spherical aberration:* Corresponds to a dependence of focal length on aperture for non-paraxial rays. The net effect of this aberration is blur, and therefore it should not be a source of registration error.

2. *Coma:* With this aberration, the effective focal length (and magnification) depends on which part of the lens the rays pass through. The effect is a comet-like image for an off-axis point. Because of its asymmetry, it is often considered to be the worst of all aberrations [Hecht & Zajac 74] and is therefore usually corrected in optical systems (even at the expense of increasing other aberrations). Because it is usually corrected, I will not treat it here.

3 & 4. *Astigmatism* and *field curvature:* These two are closely related and are usually treated together. In systems with these aberrations, the image plane degrades to one or two curved surfaces. These aberrations are discussed in detail in the next section.

5. *Distortion:* Distortion (not to be confused with the CRT distortion discussed earlier) is a warping of the image within the image plane and is discussed in Section 5.7.9.2.

The chromatic aberration of most concern for registration error is *lateral color,* in which the transverse magnification is a function of the light's wavelength. This aberration causes a tri-colored object (such as an image on an RGB display) to be imaged as a sequence of three overlapping colored images of different sizes, which may increase the registration error for full-color systems. This aberration is discussed in Section 5.7.9.3.

In the sections that follow, we will examine the errors in the virtual image itself and thus will use the *I* coordinate system rather than the *I'* CS. All the errors analyzed in these sections can then be analyzed in the *I'* CS by projecting them as discussed in Section 5.7.7.

5.7.9.1. Field Curvature and Astigmatism

In systems with field curvature (and no other aberrations), the image surface curves away from the theoretical image plane, as shown in the figure below. In systems with field curvature *and* astigmatism, the image plane becomes two curved surfaces whose shape depends on the types of lenses used. In both cases, there is a shift in focus for off-axis points such that their image is shifted longitudinally along the *chief ray*. The chief ray is any ray that passes through the center of the aperture stop[22] (also called the *pupil*); in a system with finite apertures, this ray behaves as the central ray of the ray bundle that passes unblocked through the system to form the image of the point. In typical STHMD systems, the aperture stop is the pupil of the eye. This means that the chief ray passes through the center of the entrance pupil (defined in Section 5.6.1), which I have already defined as the center of projection for the computer graphics model. Thus, this aberration moves the image point along the projector from the eyepoint through the projected point but does not cause the projector to move transversally at all. For this reason, field curvature should not affect the registration error at all and will not be treated further.

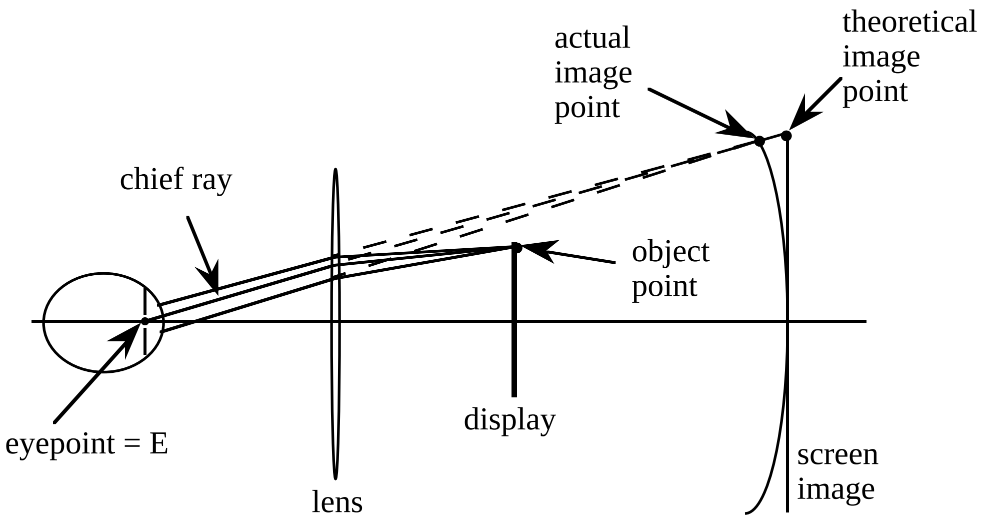


Figure 5.7.9.1.1. Field curvature for a simple system

5.7.9.2.   Optical Distortion

Optical distortion (hereafter refered to as *distortion*) is the most significant optical aberration for most HMD systems and has been analyzed in several other works [Robinett & Rolland 91][Rolland & Hopkins 93][Watson & Hodges 93]. For systems with optical distortion, the magnification for off-axis points is a function of their distance from the optical axis, which moves the images of the points transversally relative to their theoretical image points,

---

[22] This is the optical element that limits how much light reaches the image; it may be a true aperture, such as one finds in a camera or in the human eye, or it may be the edge of a lens.

thereby warping the image. As detailed in [Rolland & Hopkins 93], distortion can be approximated to third order by the following equation:

$$r_Q = m \cdot r_G + k(m\, r_G)^3 \qquad (5.7.9.2.1)$$

where r_Q is just $\|\mathbf{v}_{LI\_LQ}\|$ or $\|\mathbf{v}_{RI\_RQ}\|$ (the radial distance from the optical axis to the point in image space), *m* is the paraxial magnification of the system for a given wavelength, *k* is the third-order coefficient of optical distortion[23] for a given aperture stop, and r_G is just $\|\mathbf{v}_{LD\_LG}\|$ or $\|\mathbf{v}_{RD\_RG}\|$, the radial distance to the point in display-device space (which includes any errors made in the display device). If *k* is positive, the magnification increases for off-axis points, and the aberration is called *pincushion distortion;* if *k* is negative, the magnification decreases, and it is called *barrel distortion.*[24] Pincushion distortion is more common in HMD systems and is pictured below. Note that the corners seem more distorted than the sides due to their greater radial distance.

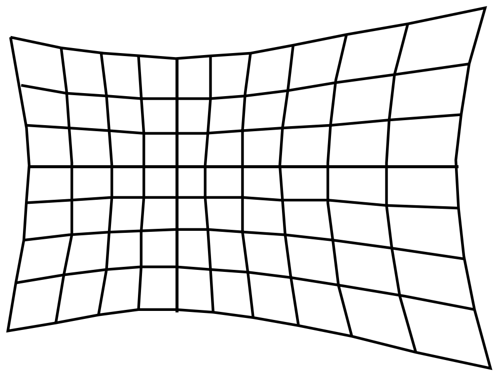


Figure 5.7.9.2.1. Pincushion distortion

Since this error does not vary with time, it can be corrected by pre-warping the image prior to display so that it appears undistorted when viewed through the optics (several approaches for this are given in the above references). Currently, though, this predistortion is so computationally intensive that it may induce more system-latency error than the warping error it corrects. It is therefore useful to examine the effect of uncorrected distortion in order to compare it with other error sources.

---

[23] This will normally be measured for a STHMD system, and will include the refractive effect of the CRT faceplate, as mentioned earlier.

[24] Off-axis optical systems may suffer from a type of distortion known as *keystone* distortion in addition to barrel or pincushion distortion, but this is less common aberration and will not be treated here.

As shown in the figure below, the image point Q' is the paraxial image point for the modeled point G' in screen space. Because of aliasing and display nonlinearities, the actual displayed point is at G. Its paraxial image point is at Q˚, but because of distortion, the final imaged point is at Q.

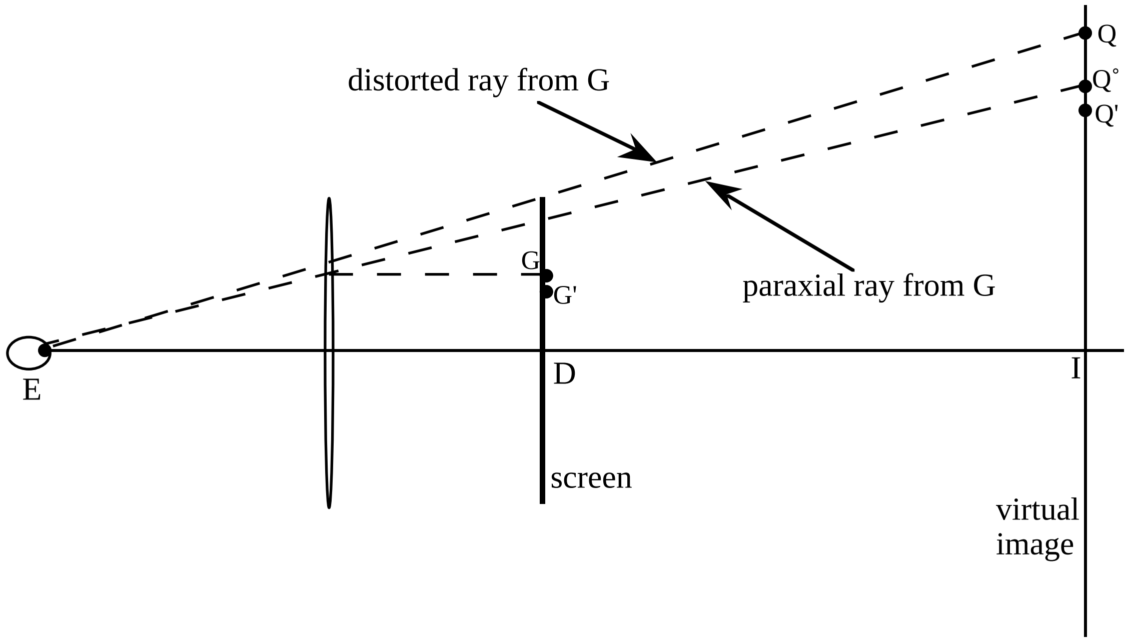


Figure 5.7.9.2.2. Modeled vs. distorted display points

In the analysis that follows, vectors will be subscripted by their endpoint and will be expressed relative to the *I* coordinate system. Furthermore, I will assume that the Z axis of the *I* CS is coincident with the optical axis for simplicity. I now move on to a derivation of the distortion-induced registration error in the presence of the display-space errors already discussed.

As indicated by Equation 5.7.9.2.1, distortion changes the radial distance of the imaged point; that is, it moves the point radially outward from its paraxial location. This can be viewed as a scaling operation on the vector $\mathbf{v}_{Q^\circ}$ since the origin is along the optical axis:

$$\mathbf{v}_Q = a\,\mathbf{v}_{Q^\circ} \tag{5.7.9.2.2}$$

where $\mathbf{v}_{Q^\circ}$ is just a linearly magnified version of $\mathbf{v}_G$:

$$\mathbf{v}_{Q^\circ} = m\,\mathbf{v}_G \tag{5.7.9.2.3}$$

and *a* is a scalar that depends on the length of $\mathbf{v}_{Q^\circ}$. We then have

$$a = \frac{\|\mathbf{v}_Q\|}{\|\mathbf{v}_{Q^\circ}\|} = \frac{r_Q}{r_{Q^\circ}} = \frac{m \cdot r_G + k(m\ r_G)^3}{m \cdot r_G} = 1 + k(m\ r_G)^2 \qquad (5.7.9.2.4)$$

If we express $\mathbf{v}_G$ in terms of the modeled screen-space point G' and the screen-space error vector $\mathbf{v}_{G'\_G}$, we have

$$\mathbf{v}_G = \mathbf{v}_{G'} + \mathbf{v}_{G'\_G} \qquad (5.7.9.2.5)$$

and

$$r_G = \sqrt{(x_{G'}+x_{G'\_G})^2 + (y_{G'}+y_{G'\_G})^2} \qquad (5.7.9.2.6)$$

which lets us express $\mathbf{v}_Q$ as

$$\mathbf{v}_Q = \begin{bmatrix} [1+k\,(m\,r_G)^2]\,m\,(x_{G'}+x_{G'\_G}) \\ [1+k\,(m\,r_G)^2]\,m\,(y_{G'}+y_{G'\_G}) \end{bmatrix} \qquad (5.7.9.2.7)$$

This expression gives the location of the actual displayed point assuming some amount of error in display coordinates and a third-order model for distortion.

The normal computer graphics model without distortion correction is much simpler. The software typically assumes that the display and the optics are linear, so that the modeled projected point within the actual image plane, Q", is just a linear scaling of the point G' by the modeled magnification:

$$\mathbf{v}_{Q''} = m' \cdot \mathbf{v}_{G'} = \begin{bmatrix} m'\,x_{G'} \\ m'\,y_{G'} \end{bmatrix} \qquad (5.7.9.2.8)$$

(*m'* is usually specified indirectly via the field of view parameter in the graphics software).

We can gain some insight into the problem with this model by looking at the graph below.

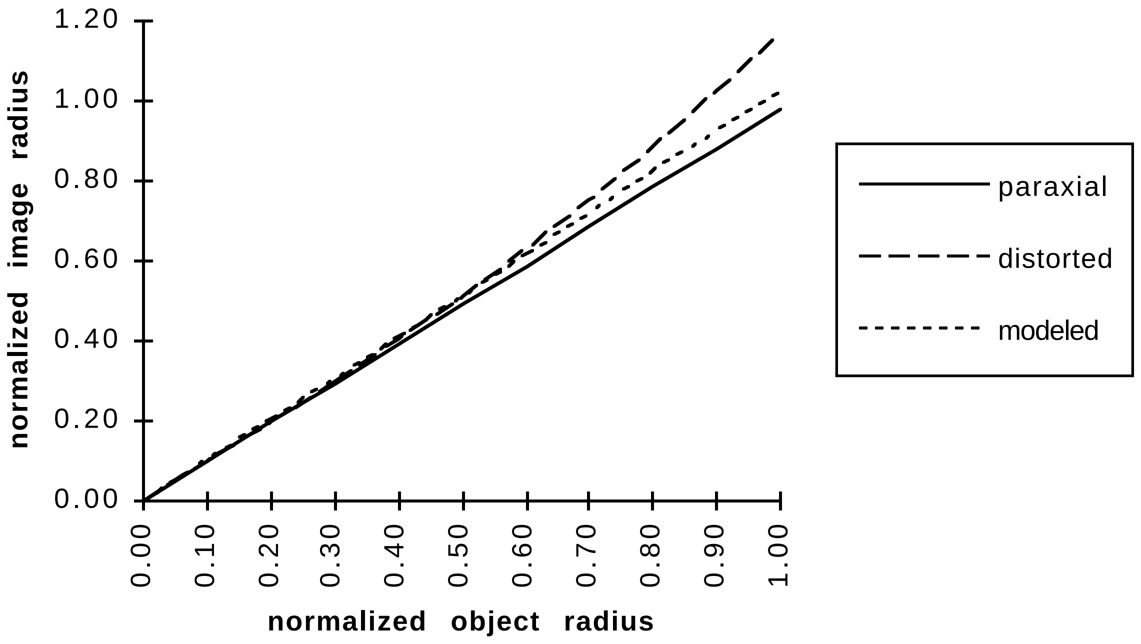


Figure 5.7.9.2.3. Linear vs. cubic model for magnification

This figure plots the normalized radial distance of an imaged point for the paraxial and distorted images. The solid line is the paraxial model, which is clearly linear and has a slope of *m*. The long dashed curve is the distorted model, which is cubic. The short dashed line is a linear approximation to the cubic curve and has a slope of *m'*. Clearly, there is no single correct slope for *m'*: if we set it equal to *m*, the model will match the distorted image in the center and will be off at the edges; if we set it so that it is correct at the edges, it will be wrong everywhere but at the edge and at the exact center.

Thus, a system that does not have the computational power to perform distortion correction will have a certain amount of registration error as a result of using the simpler model. I therefore proceed with the assumption that *m'* will not in general be equal to *m*, and we will now look at the error caused both by using the simpler model and by the non-linearities of the display devices discussed in the previous section.

The error vector in actual image space due to distortion and screen-space errors is therefore

$$\mathbf{v}_{Q''\_Q} = \mathbf{v}_Q - \mathbf{v}_{Q''} = \begin{bmatrix} (m-m')x_{G'} + m\,x_{G'\_G} + k\,m^3 r_G^2 (x_{G'} + x_{G'\_G}) \\ (m-m')y_{G'} + m\,y_{G'\_G} + k\,m^3 r_G^2 (y_{G'} + y_{G'\_G}) \end{bmatrix} \qquad (5.7.9.2.9)$$

If we look at the three terms making up each component of the error vector in Equation 5.7.9.2.9, we see that the error can be divided into

1. scaling error due to the difference in the paraxial magnification and the modeled magnification ($m' \neq m$);

2. error due to magnified screen-space error;

3. error due to not modeling the distortion.

In the absence of other errors, the *q* term for distortion is

$$q_{dist} = k\, m^3\, r_G{}^3 \qquad (5.7.9.2.10)$$

The next section investigates the interaction between lateral color and distortion.

5.7.9.3. Lateral Color

The focal distance of a lens is a function of its index of refraction, as shown by the thin-lens equation:

$$\frac{1}{f} = (n(\lambda) - 1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) \qquad (5.7.9.3.1)$$

The index of refraction for most lenses is itself a function of wavelength, which causes *f* to vary depending on the wavelength of light that passes through the lens. A common example is a tri-color system that focuses the green image correctly, but focuses the blue image in front of the green image (i.e., closer to the lens), and focuses the red image behind the green image. Thus, the red and blue images are out of focus and, depending on the pupil position, will be imaged as larger or smaller than the green image. The axial distance between the red and blue images is known as the *axial chromatic aberration,* and the vertical difference in height of the two images is known as *lateral chromatic aberration*, or *lateral color*. The simple case of a real image and an arbitrary aperture stop is shown below.

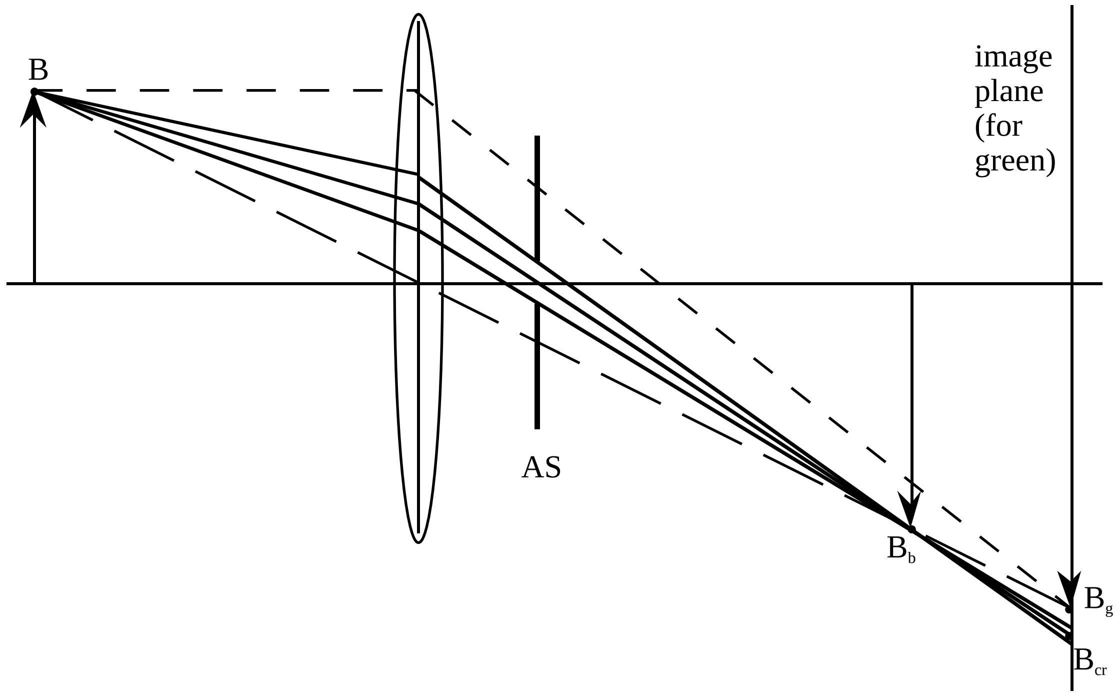


Figure 5.7.9.3.1. Lateral color

In this figure, the object point B is imaged at B_g for green light and at the point B_b for blue light due to the different focal lengths. Because B_b is not in the plane of B_g, when the light passing through B_b strikes the image plane, it will form a blur spot rather than a focused point. The aperture stop (denoted by AS in the figure) determines which bundle of rays that pass through B_b will reach the image plane. The chief ray passes through the center of the stop and gives the location of the center of the blur spot, which is marked B_{cr} in the figure. Therefore, moving the stop along the optical axis (or even off-axis) changes the chief ray, which changes the location of the blur spot. This means that the apparent size of the blue (or red) image will depend on the location of the stop. In a normal STHMD system, the stop is the pupil of the eye, and so the effect of lateral color is a change in magnification that varies with wavelength and with the eye position.

Since we are interested in warpings that move the apparent location of an imaged point, lateral color is only a concern to the extent that the average intensity of the three colored images of a point is displaced from its modeled location. That is, if the green image is accurate and the red and blue images are displaced by equal amounts to either side and have equal intensity, the intensity sum should be roughly equivalent to that of the green image, and no obvious registration error should result. On the other hand, if the red image is further from the green image than the blue image is, there will be a net shift in the center of mass of the three images, leading to registration error. All of this is complicated by the addition of distortion, which affects all three images. I now move on to a quantitative analysis of the shift in image position as a function of wavelength, distortion, and other optical parameters.

In this analysis, I will assume that the optical system has been optimized for some wavelength in the green range, and that red and blue will be imperfectly imaged as a result. Therefore, we can use the image size and distance calculated for green as the reference and compare the other images to it. The figure below shows the situation for a real image; the geometry for virtual images is similar.

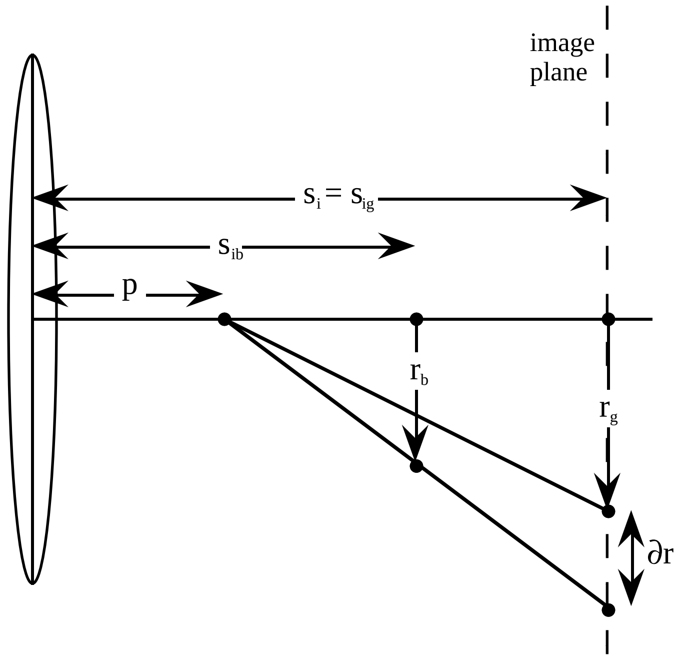


Figure 5.7.9.3.2. Geometric relationships for lateral color

In this figure, r_b is the size of the blue image at s_{ib}, r_g is the size of the green image in the image plane, and ∂r is the difference in size between the two images within the image plane. p gives the distance to the aperture stop. From similar triangles, we can see that

$$\frac{r_b}{s_{ib}-p} = \frac{r_g+\partial r_b}{s_{ig}-p} = \frac{r_g+\partial r_b}{s_i-p} \tag{5.7.9.3.2}$$

where ∂r_b may be positive or negative and the other parameters are positive. The image distance as a function of the focal length for a given wavelength is given by

$$s_{i\lambda} = \frac{s_o f_\lambda}{f_\lambda - s_o} \qquad (5.7.9.3.3)$$

where λ represents red, green, or blue, s_o is the object distance and is strictly positive, and $s_{i\lambda}$ is the image distance and is positive for virtual images (which is the case for HMDs). To account for distortion, we need only note that both r_b and r_g can be expressed using Equation 5.7.9.2.1 from the previous section. In using that expression, however, we must use the paraxial value of m for the given wavelength according to the formula

$$m_\lambda = \frac{s_{i\lambda}}{s_o} = \frac{f_\lambda}{f_\lambda - s_o} \qquad (5.7.9.3.4)$$

We therefore have

$$r_\lambda = m_\lambda \cdot r_G + k(m_\lambda r_G)^3 \qquad (5.7.9.3.5)$$

The radial distance r_G is the same parameter I used in the previous section, and indicates the distance in screen space from the optical axis to the displayed point in *screen* space (including any screen-space errors); note that this should not be confused with r_g, which is the distance from the optical axis to the green image point in *image* space.

One additional complexity with Equation 5.7.9.3.5 is that the value of k is affected by the pupil location p.[25] Although there is an analytical expression relating the two parameters, it is quite complex and involves all of the other monochromatic aberrations in the system. Moreover, the effect turns out to be negligible for the UNC STHMD system, as we will see in Chapter 6. I will therefore treat k as a constant with the understanding that this assumption must be verified for each system.

If we express ∂r for red and blue light, we can now determine the amount of shift in the center of energy representing the imaged point. The general expression for the shift relative to the green image point comes from solving Equation 5.7.9.3.2 for ∂r_λ:

$$\partial r_\lambda = r_\lambda \cdot \left(\frac{s_i - p}{s_{i\lambda} - p}\right) - r_g = \mu_\lambda r_\lambda - r_g \qquad (5.7.9.3.6)$$

where I have defined

[25] Strictly speaking, k can also vary with wavelength, but this effect is usually small and will not be treated here.

$$\mu_\lambda = \frac{s_i - p}{s_{i\lambda} - p} \tag{5.7.9.3.7}$$

as the wavelength-dependent scaling term. Expanding the *r* terms we have

$$\partial r_\lambda = \mu_\lambda[m_\lambda r_G + k(m_\lambda r_G)^3] - [m_g r_G + k(m_g r_G)^3] = (\mu_\lambda m_\lambda - m_g)r_G + (\mu_\lambda m_\lambda{}^3 - m_g{}^3)kr_G{}^3 \tag{5.7.9.3.8}$$

The perceived radial distance should just be the average of the radii:

$$r_{avg} = \frac{r_r + r_g + r_b}{3} = \frac{(r_g + \partial r_r) + r_g + (r_g + \partial r_b)}{3} = r_g + \frac{\partial r_r + \partial r_b}{3} = r_g + \frac{(\mu_r m_r + \mu_b m_b - 2m_g)r_G + (\mu_r m_r{}^3 + \mu_b m_b{}^3 - 2m_g{}^3)kr_G{}^3}{3} \tag{5.7.9.3.9}$$

We can define two variables to shorten the expression:

$$\alpha = \mu_r m_r + \mu_b m_b - 2m_g ; \qquad ß = \mu_r m_r{}^3 + \mu_b m_b{}^3 - 2m_g{}^3 \tag{5.7.9.3.10}$$

which leads to

$$r_{avg} = r_g + \frac{\alpha r_G + ßkr_G{}^3}{3} \tag{5.7.9.3.11}$$

Note that both α and *ß* go to zero as μ approaches 1 and *m* approaches m_g. The net error due to lateral color alone is just

$$q_{LC} = r_{avg} - r_g = \frac{\partial r_r + \partial r_b}{3} = \frac{\alpha r_G + ßkr_G{}^3}{3} \tag{5.7.9.3.12}$$

To express the error in exact vector form as was done in the previous section, we note that the effect of lateral color and distortion is just scaling about the optical axis, which can be expressed as

$$\mathbf{v}_Q = a_{LC\&D} \cdot \mathbf{v}_G \tag{5.7.9.3.13}$$

where

$$a_{LC\&D} = \frac{r_{avg}}{r_G} = \frac{m_g r_G + k(m_g r_G)^3 + \frac{\alpha r_G + ßkr_G{}^3}{3}}{r_G} \tag{5.7.9.3.14}$$

is the radial scaling factor due to lateral color and distortion. That is, $a_{LC\&D}$ is the ratio of r_{avg}, the distance from the optical axis to the center of the blur spot in the virtual image, and r_G, the distance from the optical axis to the displayed point in object space. In other words, $a_{LC\&D}$ is the magnification for the centroid of the distorted red, green, and blue images representing the point.

Recall that the net within-image error is described by the vector $\mathbf{v}_{Q''\_Q}$ between the modeled displayed point Q" and the actual displayed point Q (see Figure 5.7.1.1). The equation for this vector which takes into account all of the display error sources is

$$\mathbf{v}_{Q''\_Q} = \mathbf{v}_Q - \mathbf{v}_{Q''} = a_{LC\&D}\mathbf{v}_G - m'\mathbf{v}_{G'} \tag{5.7.9.3.15}$$

which expands to

$$\mathbf{v}_{Q''\_Q} = \begin{bmatrix} (m-m')x_{G'} + m\,x_{G'\_G} + k\,m^3 r_G^2(x_{G'} + x_{G'\_G}) + \dfrac{\alpha + ßkr_G^2}{3}(x_{G'} + x_{G'\_G}) \\ (m-m')y_{G'} + m\,y_{G'\_G} + k\,m^3 r_G^2(y_{G'} + y_{G'\_G}) + \dfrac{\alpha + ßkr_G^2}{3}(y_{G'} + y_{G'\_G}) \end{bmatrix} \tag{5.7.9.3.16}$$

which is identical to the expression derived in the last section for distortion and screen-space errors except for the last term, which gives the non-linear scaling error due to not modeling lateral color. Recall that

- *m'* is the modeled magnification,
- *m* is the actual paraxial magnification,
- $x_{G'}$ and $y_{G'}$ are the coordinates of the point in screen space, and $r_{G'}$ is the radial distance from the optical axis to the screen point,
- $x_{G'\_G}$ and $y_{G'\_G}$ are the errors in screen space due to aliasing and device nonlinearity,
- *k* is the coefficient of optical distortion, and
- α and β are the linear and cubic lateral color terms.

This, then, is the net within-image display error due to the modeled error sources. It consists of scaling error in modeling the paraxial magnification, errors due to aliasing and display nonlinearity which are magnified by the optics, errors due to not modeling the optical distortion, and errors due to not modeling the effects of lateral color. The magnitude of the errors from these sources will be very system-dependent. Remember also from Section 5.7.7 that this term is plugged into the expression for the total display error in modeled image space:

$$q = \|\mathbf{v}_{Q'\_Q^\dagger}\| = \|\mathbf{v}_{I'\_Q^\dagger} - \mathbf{v}_{I'\_Q'}\| = \|T_{Proj} \cdot T_{I'\_I} \cdot (\mathbf{v}_{I\_Q''} + \mathbf{v}_{Q''\_Q}) - \mathbf{v}_{I'\_Q'}\| \tag{5.7.9.3.17}$$

The within-image-plane errors will add to any rigid-body transformation errors in the I CS, after which all of the errors will be projected back onto the modeled projection plane in I' to arrive at the final value for *q*. Assuming the pan/tilt rotations in $T_{I'\_I}$ are small, the projection operator will have little effect on the within-image errors, which means they will add directly to the net display error *q*.

A final note: this model assumes that the red, green, and blue pixels are coincident, which is true for frame-sequential CRTs, but not true for LCDs. The spatial separation of the RGB elements is a small factor in an already-small error source, so it has been left out for simplicity. A more complex version of the model would treat this separation by adding it to $\mathbf{v}_{G'\_G}$ and then analyzing both the radial and non-radial components to arrive at the net contribution.

6. Error Model Analysis

6.1. Chapter Overview

This chapter analyzes the implications of the error expressions derived in Chapter 5. Specifically, it assigns numerical values to the error bounds in those expressions and determines which of the error sources are most significant and under what circumstances each error source is at its worst. The organization of the first four sections follows that of Chapter 5, in that each of the main categories of registration error (acquisition/alignment, head-tracking, viewing, and display error) are analyzed in turn. The last section compares and ranks the various error sources and discusses some of the implications of this analysis.

In order to focus the analysis I will assume that the system used is that described in Chapter 3; that is, a cranio-facial surgery planning application using the UNC STHMD[26], a Fastrak magnetic tracker, and the Pixel-Planes 5 graphics system running with a Sun 4 host computer.

The general strategy for each major section will be

1. Look at the general equation(s) for the error source in the absence of other, non-interacting errors (error sources with interaction will be treated together),
2. Derive any necessary bounds for the inputs to the equations,
3. Using bounds and system parameters, determine a coarse estimate of the source's contribution to total error source under normal conditions,
4. Draw general conclusions about the error source.

[26] Because the current UNC STHMD has a fairly narrow field of view (about 27˚ vertically), I will examine errors for FOVs up to 60˚ in order to keep the analysis as general as possible. (60˚ was chosen since it easily covers the angle subtended by a typical head at 300-500 mm, and since it is at the upper end of the range of FOVs in currently available STHMDs, such as that made by CAE [Barrette 92]).

In all that follows I will use the exact error expressions for generating error bounds and graphs unless otherwise specified; the simplified expressions will be used to give insight into any graphs presented.

6.2. Acquisition/Alignment Errors

6.2.1. Overview

As discussed in Section 5.3, I treat all of the error sources associated with getting the head dataset into World space as one error category: acquisition/alignment error. This includes all errors in the CT scan, polygonalization of the resulting volume of numbers, and alignment of the polygonalized dataset in W. In Section 5, we saw that the general procedure for this part of the pipeline is as follows:

- The anatomy is imaged with a CT scanner, and the resulting volume is converted to a set of polygons.
- Landmarks are picked on the polygonal dataset and on the actual patient using a digitizer, a process which yields the *virtual* and *real landmark sets*.
- The real and virtual landmark sets are used as inputs to an alignment algorithm which computes the transformation for aligning the polygonal dataset with the real patient in World space.

As indicated in Section 5.3, an analytical model of these processes is beyond the scope of this dissertation. In order to estimate the registration error generated in this stage of the pipeline, I first performed a simulation in order to investigate the error propagation behavior of a particular alignment algorithm as a function of errors in its inputs, and then used results from the literature to estimate the errors in the medical imaging, polygonalization, and digitization processes. The simulation showed that as the number of landmarks increases (*i.e.,* if we digitize more points on the real patient and the virtual anatomy), the errors in picking the landmarks tend to cancel out and the net registration error is mostly due to errors in the virtual dataset itself (*i.e.,* errors in the medical imaging and polygonalization processes). The rest of this section describes the simulation and the results in more detail.

6.2.2. Description of the Simulation

In order to understand the simulation, let us first examine the whole alignment process in terms of a flowchart, shown below.

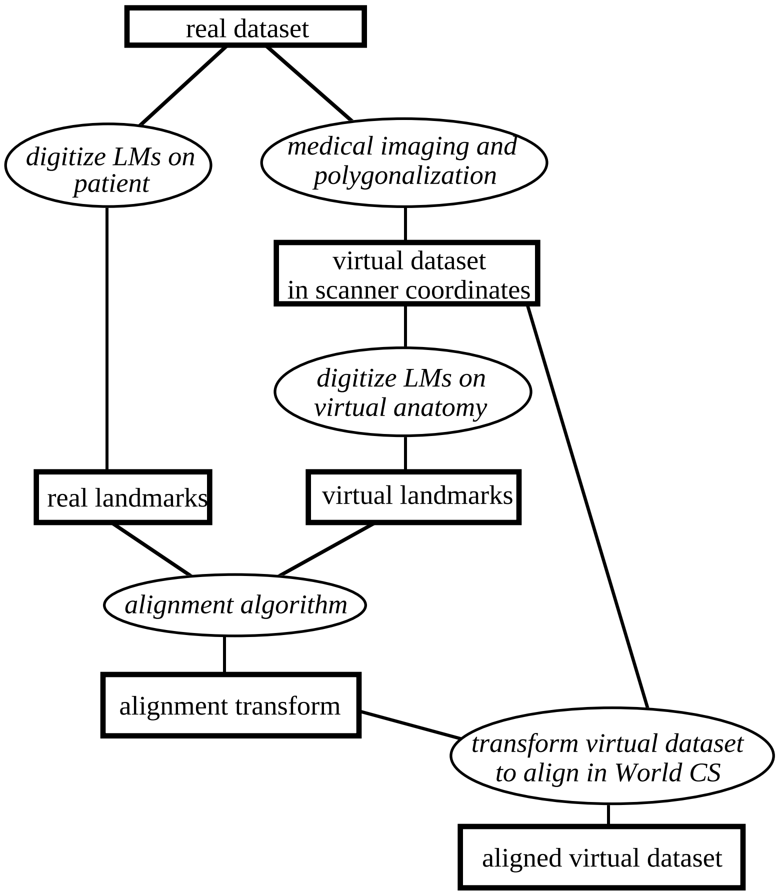

real dataset
digitize LMs on patient
medical imaging and polygonalization
virtual dataset in scanner coordinates
digitize LMs on virtual anatomy
real landmarks
virtual landmarks
alignment algorithm
alignment transform
transform virtual dataset to align in World CS
aligned virtual dataset

Figure 6.2.2.1. Flowchart of alignment procedure (rectangles indicate data and the ovals denote processes)

On the left side, the real landmarks are digitized on the real patient and are used as inputs to the alignment algorithm. On the right side, we see that the real data is imaged and polygonalized to yield the virtual dataset in CT scanner coordinates. This data is then used for picking the virtual landmarks (at locations corresponding to those for the real landmarks), which are also inputs to the alignment algorithm (which produces the alignment transformation). The alignment transformation is used to transform virtual dataset so that it is aligned with the real patient in World space (as indicated by the bottom right part of the chart). The net error for the whole procedure is determined by computing the distance between selected points (called *target points*) from the real dataset and the aligned virtual dataset. Note that the landmark sets are used only for computing the alignment transform, but not for evaluating the resulting alignment.

The simulation follows the chart, as described next.

1. The real dataset for the simulation is a human skull dataset (courtesy of Bill Lorensen of General Electric) consisting of 484 surface points[27]. These points are taken to represent the real anatomy exactly[28] and are therefore "ground truth" for the simulation.

2. To generate the real landmarks, a set of N_{LM} points are chosen from the real dataset (uniformly distributed through the volume). In the next step, noise vectors are added to these points to simulate measurement error in the digitization process. The noise vectors are generated by creating randomly oriented unit vectors and then scaling them by a zero-mean, normally distributed[29] random variable with standard deviation σ_{real}. The real landmarks are therefore copies of a subset of the points in the real dataset with noise added to them.

3. Medical imaging and polygonalization are treated as a single measurement process and the errors (hereafter referred to as MI/P errors) are modeled in the same way as just described for landmark digitization errors: zero mean noise vectors with normally distributed magnitudes described by $\sigma_{MI/P}$. To generate the virtual anatomy in scanner coordinates, a copy of the real dataset is translated and rotated[30], and then noise vectors are added to the points to simulate medical imaging and polygonalization artifacts. The virtual dataset is thus a translated and rotated version of the real dataset with noise added to each point.

4. The virtual landmarks are generated in the same general fashion as the real landmarks: points are picked from the virtual anatomy and more noise is added to them to simulate error in the digitization process. The standard deviation for the error-vector magnitudes is σ_{virt} (I do not assume that $\sigma_{real} = \sigma_{virt}$ since the digitization process for these two sets of landmarks will usually be different, as described in Chapter 5). The process of digitizing the virtual LMs is treated separately from that of medical imaging and polygonalization because the virtual

[27] The original dataset contained 19,340 points; a representative subset of the points was generated by taking one point in 40 from the file.

[28] For example, these could be interpreted as the polygon vertices of a polygonal head phantom whose geometry is known exactly by definition.

[29] I assume that the errors are zero-mean and normally distributed since they describe a measurement process and should therefore be independent of each other.

[30] For simplicity, the scale was left the same for both datasets.

LMs are only used for computing the alignment transform. Thus, error in this stage does not warp or distort the virtual dataset itself (as errors in imaging and polygonalization do): it only affects how accurately the alignment transform is computed.

5. The real and virtual landmarks are then used as inputs to the alignment algorithm, which gives the best transformation for aligning the virtual landmarks with the real landmarks (and, we hope, the real and virtual datasets). The alignment algorithm used is that described in [Horn 87] and used in [Turk & Levoy 94] because it is a closed-form solution that involves no approximation. The algorithm assumes that the landmark sets which are to be aligned are offset by only a rigid-body transformation and a scale factor. The best-fit translation between the two datasets is the vector between the centroid of the reference dataset (here the real anatomy) and the scaled and rotated centroid of the other dataset (the scanned anatomy). The scaling factor is just the ratio of the root-mean-square deviations of the coordinates in the two CSs from their respective centroids. The best-fit rotation is determined via a closed-form solution based on quaternions. For more details on the algorithm, see [Horn 87]. The transformation produced by the alignment algorithm is used to transform the virtual dataset so that it is aligned with the real dataset in World space.

6. In the final step, the resulting registration error for between the real and aligned-virtual datasets is computed. The error metric used is the distance between each of the target points in the real dataset and the aligned virtual dataset. For this simulation, I used all of the points in each dataset as target points in order to get the best estimate of the alignment error (although in a clinical setting one would most likely use only a subset of the points). The squared distances are summed and divided by the number of points, and the square root of the result is the root-mean-squared (RMS) error for that run. Because the noise varied from one run to the next, 20 runs were made for each set of parameter values and the RMS error values were averaged.

The data from the simulation follows.

6.2.3. Simulation Results

As indicated in the last section, the simulation divides the input errors into three categories: 1) real landmark digitization error, described by σ_{real}, 2) medical imaging and polygonalization error, described by $\sigma_{MI/P}$, and 3) virtual landmark digitization error, described by σ_{virt}. The other parameters to be varied are those specifying the translation and rotation between Scanner space and World space and the number of landmarks used (N_{LM}). I found that there was no significant variation of the results due to the rotation and translation parameters, so it was left at a single random orientation, and the translation was a random vector with magnitude roughly equal to half the model size. The parameters examined in this simulation are the three error sigmas and the number of landmarks.

Let us begin by examining the behavior of the alignment algorithm when there is no error in the virtual data, but there is digitization error in the real and virtual landmarks. The plot below shows the mean RMS error for aligning the real and virtual datasets for equal amounts of digitization error.

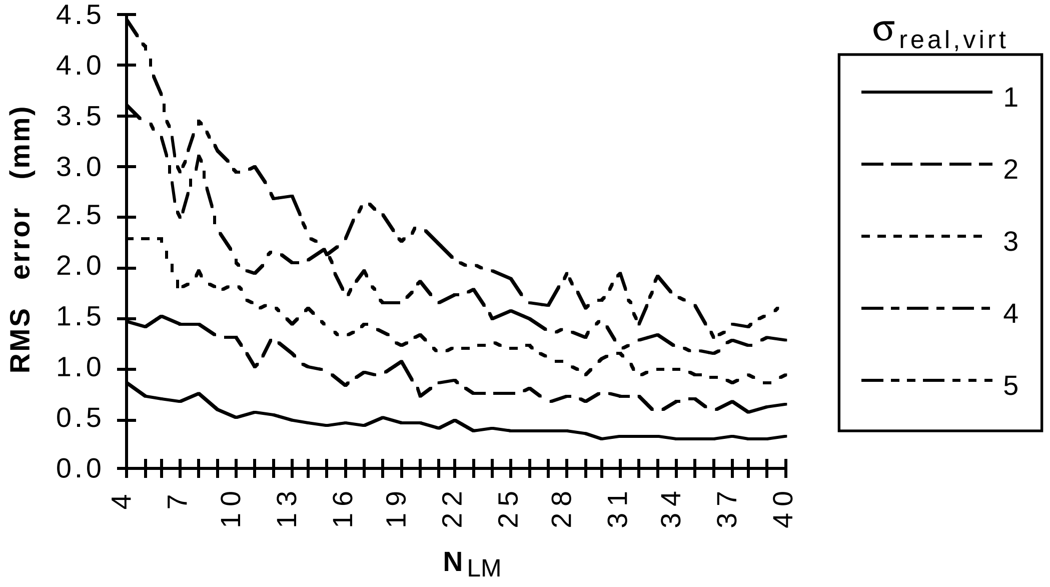


Figure 6.2.3.1. Mean RMS error as a function of LM digitization error only

For this figure $\sigma_{MI/P} = 0$ and $\sigma_{virt} = \sigma_{real}$ with values ranging from 1 to 5 mm. The number of landmarks (real and virtual) is plotted along the horizontal axis. The general behavior is that error in digitizing the landmarks tends to average out as more landmarks are used. For the case where $\sigma_{virt} = \sigma_{real} = 1$ mm, the error starts below 1 mm and quickly converges to about 0.5 mm. The other four cases show similar behavior, but since they start out with more error, it takes more landmarks to reduce the error.

In the next plot I increase the number of landmarks for the case where $\sigma_{virt} = \sigma_{real} = 5$ mm to verify that the error continues to decrease with the number of landmarks used.

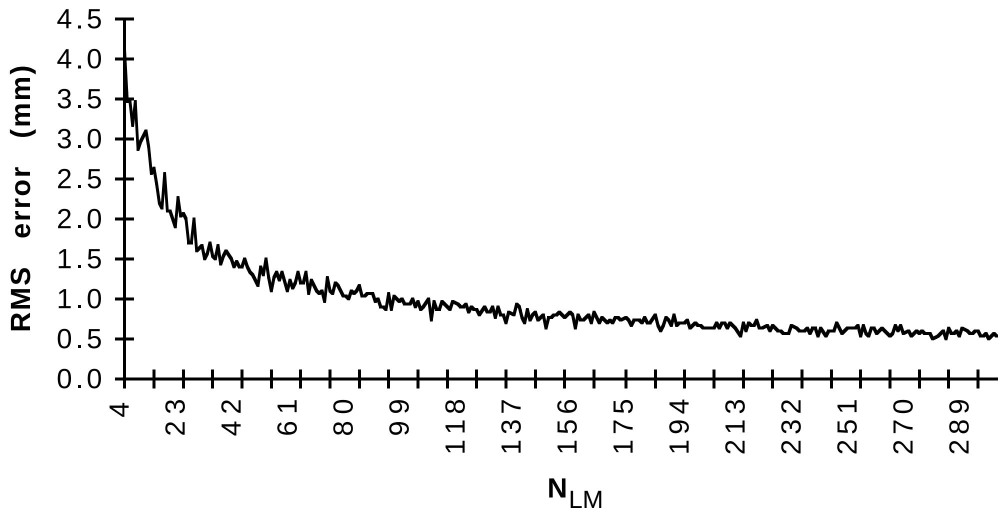


Figure 6.2.3.2. Mean RMS error for $\sigma_{virt} = \sigma_{real} = 5$ mm for up to 300 landmarks

This plot illustrates the general behavior of the error decreasing as more landmarks are used, even though picking 300 landmarks by hand may not be reasonable for this application.

We can now examine the case where the error is only in the medical imaging/ polygonalization process, as shown in the plot below.

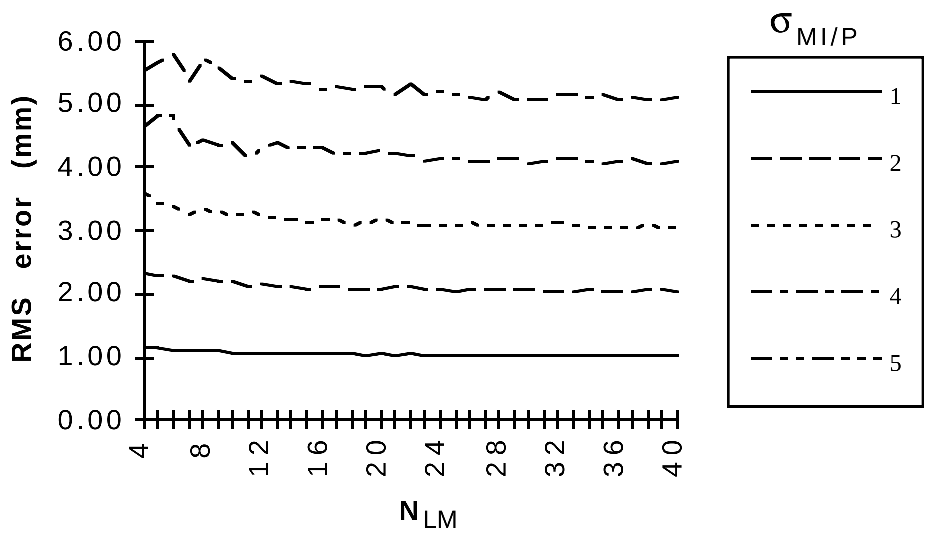


Figure 6.2.3.3. Mean RMS error for pure medical imaging/polygonalization error

The behavior here is different: the RMS error decreases with the number of landmarks used, up to $N_{LM} \approx 20$, where it plateaus and approaches $\sigma_{MI/P}$. That the RMS error should approach $\sigma_{MI/P}$ makes sense, since the error metric in this case is the same as the standard deviation of the error-vector magnitudes, which is just $\sigma_{MI/P}$.

Now that we have looked at the behavior of the errors in isolation, we can examine them in combination.

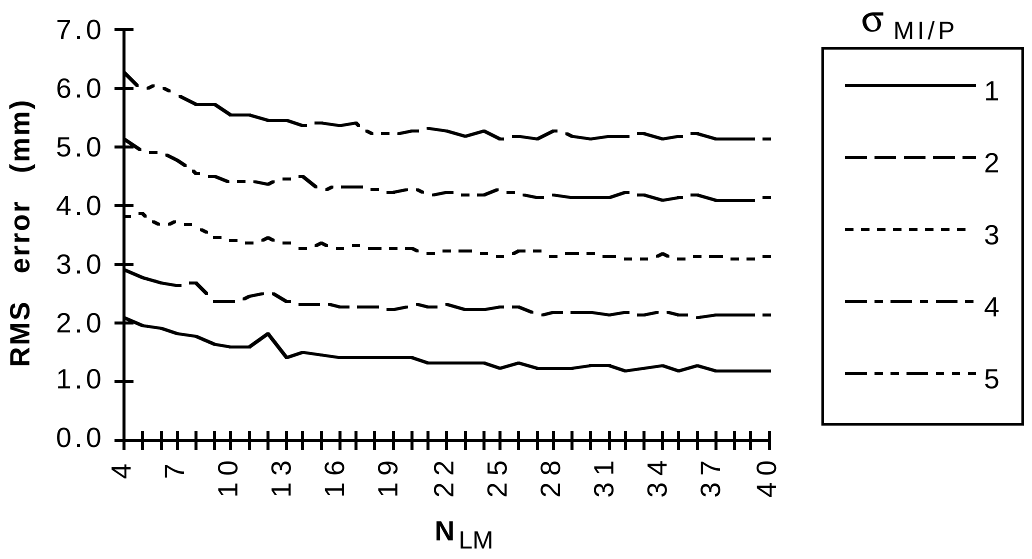


Figure 6.2.3.4. RMS error for both types of input error

For this chart $\sigma_{virt} = \sigma_{real} = 2$ and $\sigma_{MI/P}$ varies from 1 to 5 mm for various values of N_{LM}. The behavior for the comination of the two error types is just a combination of the behaviors already seen: the landmark digitization errors tend to average out for increasing values of N_{LM}, and the curves tend toward a terminal value of $\sigma_{MI/P}$ as N_{LM} increases. That is, the digitization errors add to the MI/P errors at first, but tend to average out as more landmarks are used.

Finally, we can examine whether the case where $\sigma_{virt} \neq \sigma_{real}$, as plotted below.

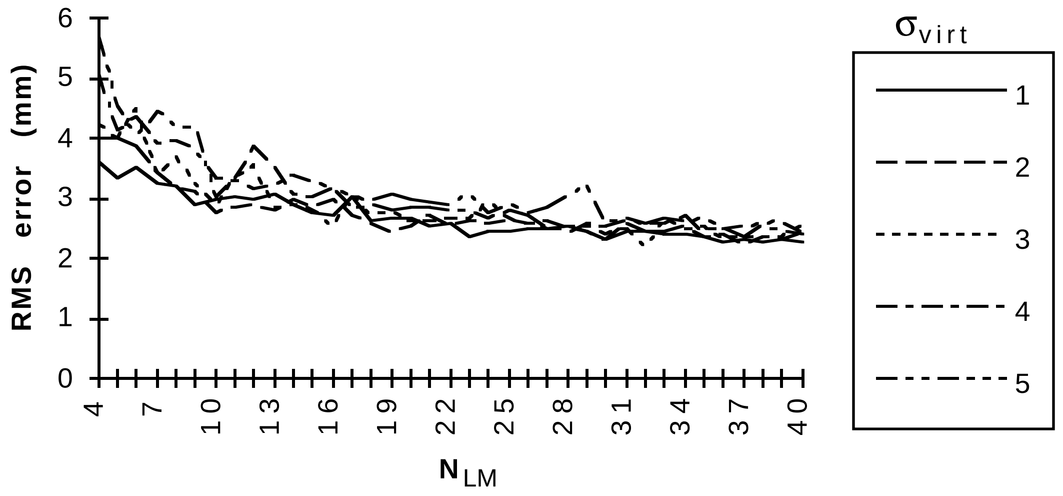


Figure 6.2.3.5. Error due to landmark error is dominated by the maximum of σ_{virt} and σ_{real}

In this plot $\sigma_{MI/P}$ = 2 mm, σ_{real} = 5 mm, σ_{virt} takes on integer values from 1 to 5 mm, and N_{LM} is varied as before. Note that the curves are bunched up together even though the value of σ_{virt} varies significantly. In general, the digitization error behavior is determined by the larger of σ_{virt} and σ_{real}, which means that any effort at reducing this error source should be targeted at the larger of the two.

In summary, the difference between MI/P error and digitization error is that digitization error only affects the alignment transform and tends to average out, whereas MI/P error adds error to the virtual dataset directly and therefore imposes a fundamental limit on how well it can be registered with the real dataset.

6.2.4. Estimated Error Bounds

Now that we have examined the general behavior of the alignment procedure, we can move on to the problem of estimating the amount of error in the landmark sets themselves.

For digitization errors we have seen that the sensitivity of the net registration error to digitization error is small if enough landmarks are used. Because of this, the error bounds for σ_{virt} and σ_{real} need not be estimated with great precision. I will use an estimate of 2 mm for σ_{virt} and σ_{real} for the following reasons:

- Accurate digitizers (such as the Faro arm [Faro 93]) can be used to achieve submillimeter accuracy for the real landmarks, so the digitizer need not be a limitation for this part of the process.
- Feedback mechanisms can be used to reduce errors in landmark positions. That is, if two corresponding landmarks are separated by a distance greater than some tolerance, they can be redigitized or removed. For more details on techniques, see [Toennies et al. 90].
- Soft-tissue error in the real landmarks can be reduced by estimating tissue depths from the CT data and by iterating the alignment procedure to detect failures in this estimation.
- As indicated in Section 5.3, the virtual landmarks can be chosen on a 2D monitor at any scale and can be constrained to lie on the polygonal surface, so the digitization error in them should be negligible.
- The human error in picking landmark locations on CT skull data is reported in [Herman 91]. He notes that although some methods for choosing landmarks cause significant variations in landmark locations (3-10 mm), reliable methods can be found which yield variations on the order of a pixel side, which in this case was less than a millimeter. Again, the feedback mechanisms described above can be used to ensure that human error is detected and corrected.

The error estimates for medical imaging and polygonalization are difficult to make since they are tied to so many parameters: the type of scanner used, the radiation beam intensity, the amount of patient movement, etc. I will use the one-half the voxel size as an estimate of σ_{virt} for the following reasons:

- The voxel size (and in particular, the slice thickness) is a key factor in the visual artifacts that are plainly visible in CT reconstructions. The size of many false structures such as “staircasing” around the eye sockets is determined by the voxel size. In addition, the size of the voxel is one of the key system parameters that limits the amount of detail in the reconstruction[31].
- Accuracy studies in the medical imaging literature are often given in terms of the pixel or voxel size (examples include variation in picking CT landmarks [Herman 91],

[31] Although [Lee & Rao 87] show that other components such as the reconstruction filter may limit the resolution of the device more than the pixel size does.

assessment of stereotactic frames as a function of CT slice thickness [Galloway et al. 91], and registration of CT data with the real anatomy using a mechanical digitizer [Adams et al. 90]).

- The polygonalization algorithm creates polygons defined in terms the voxel dimensions, which means that any artifacts due to the faceted nature of the resulting polygonal dataset will be on the order of the voxel size. In particular, since many landmarks are defined as curvature extrema and the curves will be defined in terms of the polygon vertices, one would expect the amount of error in such landmarks to be determined by the polygon size (and therefore the voxel size).

Since σ_{virt} is the standard deviation rather than an upper bound, this estimate allows for larger errors as well (more on this below).

Common slice thicknesses in cranio-facial surgery are 2-3 mm [Vannier et al. 91][Wojcik & Harris 91] with pixel sizes on the order of 1 mm by 1 mm. If we take one-half the diagonal of the voxel as our value for σ_{virt}, we get 1.2 mm for a 1x1x2 mm voxel and 1.7 mm for a 1x1x3 mm voxel. For a normal distribution, 95% of the values are within the range $\pm 2\sigma$, which gives 95% confidence bounds of 2.4 mm and 3.4 mm for these voxel sizes.

Under these assumptions, the net error should be in the range of 1-3 mm (assuming enough landmarks are used).

Summary/observations:

- Noise in the real and virtual landmarks tends to average out as the number of landmarks increases. The more noise there is, the more landmarks need to be picked before it averages out.
- The RMS error tends to converge to the standard deviation of the medical imaging/ polygonalization error. That is, the alignment process neither magnifies nor reduces the error in the scanned, polygonalized dataset.
- Digitization error is determined by the maximum of σ_{virt} and σ_{real}. This means that any effort at reducing it should be targeted only at the larger error source; there is probably no benefit to reducing the smaller of the two.
- Maximum error in this phase of the pipeline should on the order of 1-3 mm under the stated assumptions.

- Errors made in this stage will be stationary with respect to World space, in contrast to many of the errors made in the later stages of the pipeline.

6.3. Head-Tracking Error

6.3.1. World-Tracker Transformation

The error in this transformation is essentially a calibration error and is dependent on both the calibration technique used and the accuracy of the tracker itself. As reported in [Janin et al. 93], the origin and orientation of magnetic trackers is difficult to measure directly with any accuracy, since the origin is inside of a transmitter.

As stated in Section 5.4.1, if we have the option of defining the World CS to be coincident with the Tracker CS, we can omit this transformation, and therefore its error, entirely. The advantage to doing this should be evident from looking at the overall error expression derived in Section 5.4.3 for transforming a vector:

$$b_{head\_track} = \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{S'\_S}\| + 2\cdot\left|\sin\frac{Ø_{T'\_T}}{2}\right|\cdot\|\mathbf{v}_{T\_S'}\| + 2\cdot\left|\sin\frac{|Ø_{T'\_T}| + |Ø_{S'\_S}|}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \qquad (5.4.3.2)$$

Although the translational error just adds in without magnification, note that the angular term $Ø_{T'\_T}$ appears twice in the overall error expression and that the *sin* term is multiplied by vector magnitudes which are normally on the order of 500 mm. (Remember that the $2\cdot|\sin\frac{Ø}{2}|\cdot\|\mathbf{v}\|$ terms reduce to $\|\mathbf{v}\|\cdot Ø$ for small angles, which may give a more intuitive feel for the above expression.)

When the only error is in T_{W_T} (*i.e.*, $\|\mathbf{v}_{S'\_S}\| = 0$ and $Ø_{S'\_S} = 0$), this expression reduces to

$$b_{W\_T} = \|\mathbf{v}_{T'\_T}\| + 2\cdot\left|\sin\frac{Ø_{T'\_T}}{2}\right|\cdot\left(\|\mathbf{v}_{T\_S'}\| + \|\mathbf{v}_{S\_P''}\|\right) \qquad (6.3.1.1)$$

Let us look at the error for some plausible parameter values:

$Ø_{T''\_T} = 0.5^\circ$	(error in orienting the tracker in the World CS)
$\|\mathbf{v}_{T'\_T}\| = 1$ mm	(error in positioning tracker source in World CS)
$\|\mathbf{v}_{T\_S'}\| = 500$ mm	(head is about .5 m from source)
$\|\mathbf{v}_{S\_P''}\| = 500$ mm	(normal working distance has virtual image .5 m from head)

Now let all of the other errors be equal to zero so that we can examine the effect of a .5˚ error in orienting the tracker. We get an overall error of

$$b_{W\_T} \;=\; 1 + 2 \cdot \sin\frac{0.5^\circ}{2} \cdot (500 + 500) \;=\; 1 + 8.7 = 9.7 \text{ mm}$$

For magnetic trackers (such as the Fastrak), there is no reliable, external reference for orienting or positioning the tracker source in World space. Therefore, the most plausible approach involves taking tracker readings and deducing the T_{W_T} transform from them. In this case, one would expect the accuracy of the procedure to be bounded by the static accuracy of the tracker itself. The quoted specifications for the Polhemus Fastrak for distances up to 760mm [Polhemus 92] are 0.15˚ RMS for static angular accuracy and 1.32 mm static translational accuracy.[32] Even with the smaller angular term, we still get an error of 3.92 mm. The next section has more information on tracker accuracy in real laboratory environments.

For small angular errors, the behavior of this error source is roughly linear; if we use $\|\mathbf{v}_{T'\_T}\|$ = 1.32 mm and the values given above for the other parameters and vary $Ø_{T''\_T}$, we get the following plot for the registration error:

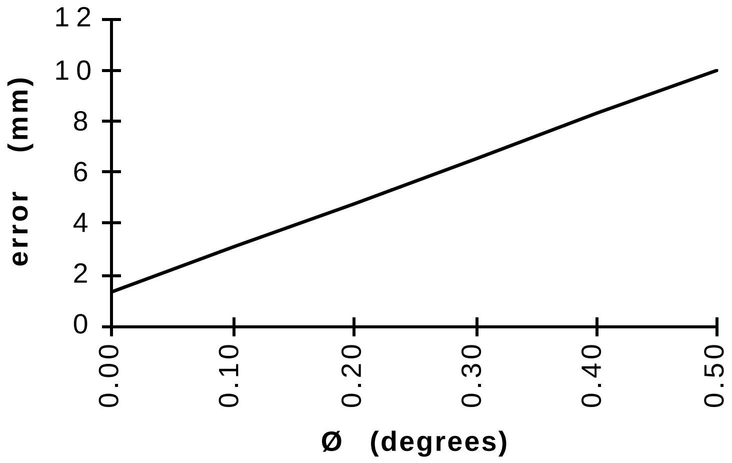


Figure 6.3.1.1. Registration error as a function of T_{W_T} orientation error

All of this is assuming rather reasonable values for the working distances and no other errors at all.

Observations:

[32] This figure calculated as the root mean square of the quoted X, Y, and Z RMS values.

- Translation error in T_{W_T} will be fixed in World space, which allows it to be distinguished from many of the other error sources.
- Linear alignment of the tracker source is not nearly as important as angular alignment, since the latter is scaled significantly by the sum of the magnitudes of the vectors between the source and the point to be displayed.
- It is the distance from the tracker origin to the point that matters for the angular error; any increase in the component vectors will, in the worst case, linearly increase the error.
- Moving the tracker source as close as possible the center of the working volume will help minimize $||\mathbf{v}_{T\_P'}||$ (this is discussed more in the next section).

Thus, while there may be a case to be made for maintaining the World CS as independent from the Tracker CS for some systems, it is clearly advantageous to coalesce the two CSs when possible. The heart of the problem is that angular errors in orienting the tracker precisely in W are magnified by the "moment arm" of the tracker-to-point distance, which can be quite large. If we can eliminate this transform from the system by measuring point locations relative to the tracker, the net error should go down. For systems that require a separate digitizer for aligning the virtual objects in the real environment (and therefore a World CS), it should be possible to use the scaling behavior of this error source in order to detect it and calibrate it out of the system; that is, by using large head-to-tracker and head-to-point distances, it should be possible to use this moment arm to advantage and reduce the errors in T_{W_T} to a negligible level.

6.3.2. Tracker-Sensor Transformation

The problem with quantifying tracker error is that it is very dependent on the tracker technology and the environment in which the tracker is used. For example, magnetic trackers are sensitive to metal and electromagnetic fields in their operating environment, yet most AR setups are in labs chock full of electronic equipment and have significant amounts of metal in walls, floors, etc. In such difficult environments, the error in the tracker measurements may exceed the manufacturer's specifications by an order of magnitude or more. If we use Equation 5.4.3.2 and assume no other errors are made, we have

$$b_{static} = ||\mathbf{v}_{S'\_S}|| + 2\cdot\left|\sin\frac{\text{Ø}_{S'\_S}}{2}\right|\cdot||\mathbf{v}_{S\_P''}|| \qquad (6.3.2.1)$$

which shows that translation error just adds to the registration error, but rotational error is magnified by the distance to the point. For $||\mathbf{v}_{S\_P}|| = 500$ mm, each degree of angular error yields about 9 mm of registration error. The next sections examine the various types of tracker error and generate approximate error bounds for each type.

6.3.2.1. Static Tracker Error

As discussed in Section 5.4.2, static tracker error is the error present due to noise in the tracker's outputs after static field distortions have been calibrated out.[33] This gives the amount of error one can expect when the user holds her head perfectly still.

The manufacturer's accuracy numbers give a good indication of the *best* we can hope for with a given tracker: for example, if we use the specified static accuracy for the Polhemus Fastrak we get

$$b_{static} = 1.32 + \left| 2 \cdot \sin\frac{0.15^\circ}{2} \right| \cdot 500 = 2.6 \text{ mm}$$

Since the manufacturer's specifications are rarely met in real laboratory environments, I next examine the literature to see what sort of accuracy others have achieved in their labs.

[Bryson 92] measured the position error vectors for a Polhemus Isotrak on a regular grid through the tracker's operating volume. They used a pegboard on a stand so that the sensor could be moved at regularly spaced intervals with a known position and orientation at each point. They report a systematic distortion of the measurement volume which is a function of distance from the tracker transmitter. The paper then describes a number of calibration methods used to compensate for the systematic errors. For distances up to about 40" (1,016 mm) from the transmitter, they were able to reduce the errors from roughly 5" (127 mm) to 1-2" (25-51 mm). They also made measurements of short-term and long-term jitter. The readings in a one-second interval had a standard deviation which varied from less than 0.1" (2.5 mm) for distances up to 30" (760 mm) to about 0.4" (10 mm) at 40". The readings for a fixed sensor from one day to the next also varied as a function of sensor-transmitter distance, with values under 1" (25 mm) for separations up to 20" (500 mm) and 3" (76 mm) at 40". Variations in orientation data were not measured, but it should be clear that registration errors for a system employing this tracker would be intolerably large.

---

[33] I assume that the tracker has been calibrated because uncalibrated trackers may have huge errors; for example, [Ghazisaedy et al. 95] report errors of up to three *feet* with a Flock of Birds magnetic tracker before calibration.

More recently, [Ghazisaedy et al. 95] report on work they have done towards calibrating an Ascension Flock of Birds tracker. Because their system is a CAVE environment, in which the virtual objects are drawn on rear-projection screens that surround the user in a 10'x10'x10' volume, they had to calibrate the tracker over a larger volume than one would need for a surgery planning application. However, because the data is displayed on the CAVE's walls rather than on an STHMD, the sensitivity to tracker error is lower and therefore need not be corrected to the same level of precision. They use a hand-held ultrasonic tracker as the reference tracker and compute the error vector between it and the Flock's sensor. The ultrasonic tracker has an accuracy of about one inch (25 mm), which clearly makes it unsuitable for millimeter-level calibration work. On the other hand, their errors for an uncorrected system were on the order of two feet (at the edge of the operating volume), which meant that the achieved level of calibration (about 1-3" (25 -76 mm)) was a substantial improvement.

Because both of these studies were geared toward correcting large errors over a large volume, the data does not give much of an idea of how well these trackers could be calibrated for close work. It seems entirely likely that the calibration methods were hampered by the very noisy data at the extreme end of the tracker's range.

Because it is tracker jitter that limits the success of a calibration method, I performed a simple study (detailed in Section 7.2.1) to examine the jitter in the Fastrak's readings in the lab at UNC. The following plot (reproduced from Chapter 7) shows the measured standard deviations in the Fastrak's position and orientation readings as a function of $||\mathbf{v}_{T\_S}||$, the transmitter-to-sensor distance.

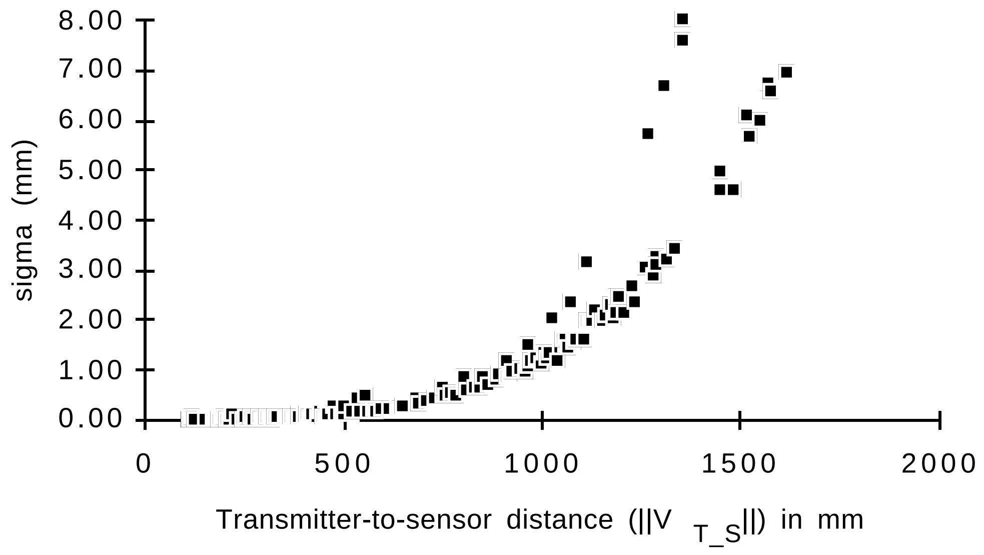


Figure 6.3.2.1.1. Translation sigma values for Fastrak

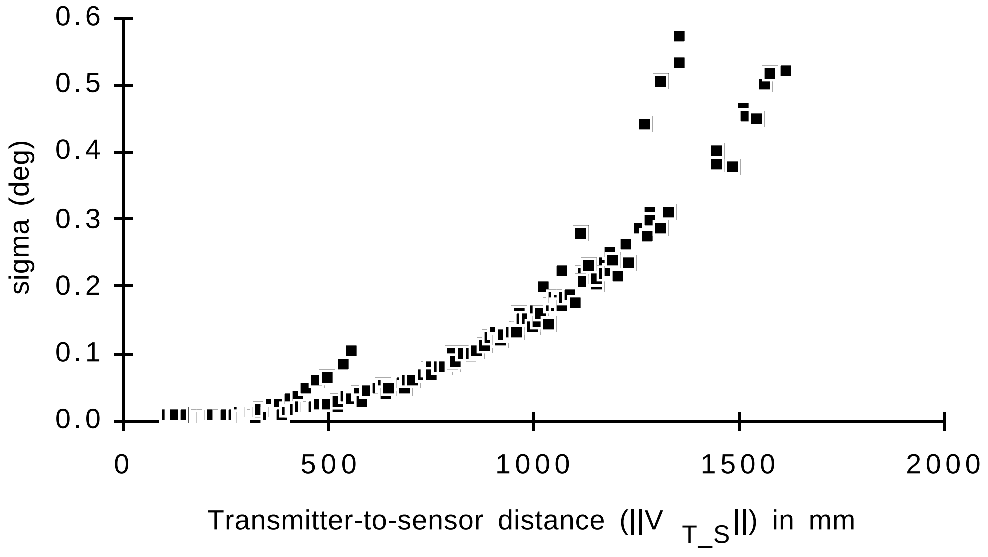


Figure 6.3.2.1.2. Orientation sigma values for Fastrak

The jitter (and thus the registration error) appears to be a function of the square of the source-sensor separation (because of the falloff of the magnetic field with distance) and becomes very large for separations beyond 1000 mm. Interestingly enough, the Fastrak is within its specifications for source-sensor separations of up to 762 mm and slightly beyond (about 1000 mm). The performance deteriorates rapidly from this point, however, with deviations approaching 7 mm and 0.5˚ at 1500 mm. The plot that follows is a magnified version of the data already presented and shows the behavior for small separations.

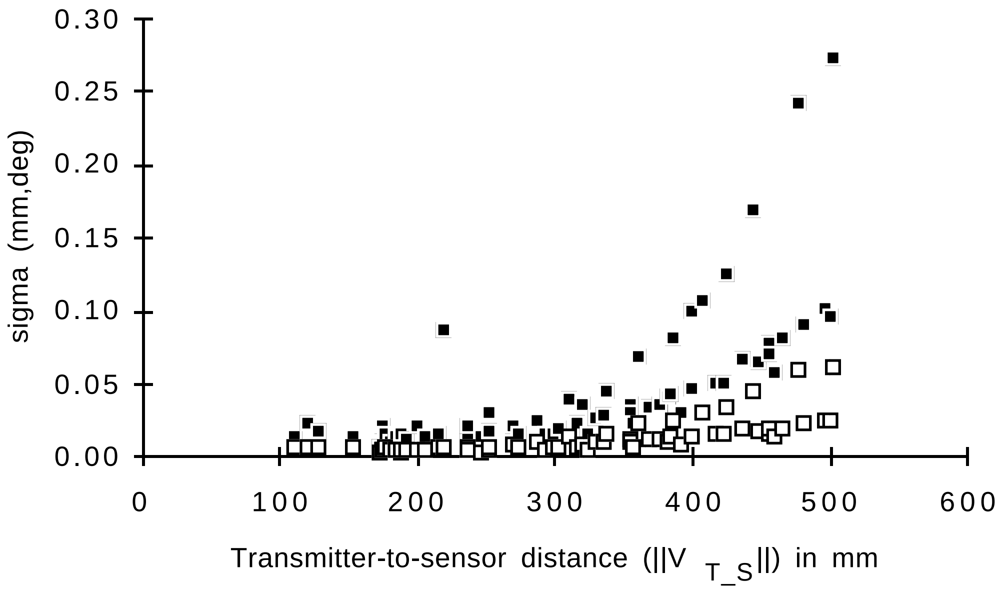


Figure 6.3.2.1.3. Translation (black squares) and orientation (white squares) sigma values for small (0-500 mm) separations

In this range, the jitter is quite small: the translation sigma values are 0.25 mm or lower, and the orientation sigmas are all below 0.05˚. Using these maxima gives 0.69 mm of registration error, which is quite reasonable. This gives hope that a carefully calibrated tracker used in a small working volume may be able to provide the required accuracy.

The registration errors corresponding to the entire range of plotted values are given in the table below.

$\|\mathbf{v}_{T\_S}\|$	**translation term**	**orientation term**	**total error**
200	0.02	0.00	0.06
400	0.07	0.02	0.21
600	0.21	0.04	0.57
800	0.61	0.08	1.35
1000	1.22	0.15	2.52
1200	2.35	0.24	4.42
1400	5.71	0.44	9.54

Table 6.3.2.1.1. Registration errors as a function of $\|\mathbf{v}_{T\_S}\|$

This table was calculated using Equation 6.3.2.1.1 and a local average (using 5 samples) for the values of $\|\mathbf{v}_{T\_S}\|$.

At this point, it is logical to examine what the range of transmitter-to-sensor distances is for the surgery planning application. As detailed in Section 7.2.2, I conducted a simple user study to gather data on typical head velocities and viewing distances for this task. The distribution of viewing distances is plotted below.

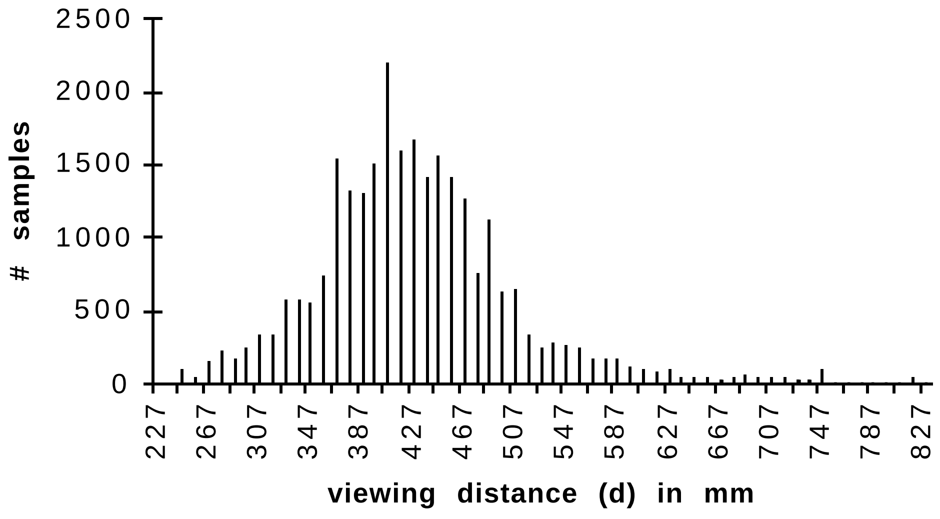


Figure 6.3.2.1.3. Histogram of viewing distances for surgery planning session

The chart clearly shows that most of the time, the surgeon's head is within 500 mm of the subject and almost never moves beyond 750 mm. From the table above, this range corresponds to registration errors (for a perfectly calibrated tracker) of less than a millimeter for most of the time.

The message of this section is therefore that the tracker must be calibrated and placed within the working volume to minimize the sensor-transmitter distance, a conclusion which should come as no surprise. If the calibration can be done so that it is limited only by the tracker jitter, it should be possible to achieve good static registration. In any case, Equation 6.3.2.1 can be used to determine the registration error as a function of the residual static error after calibration.

6.3.2.2. Dynamic Tracker Error

As discussed in Section 5.4.2, dynamic tracker error is any error in the tracker's measurements caused by the movement of the sensor while measurements are being made. Recent work by [Adelstein et al. 95] addresses this very issue. They used a swing-arm apparatus driven by a servo motor to move a sensor (for Polhemus Fastrak and Ascension Flock of Bird trackers) relative to the tracker source in a controlled manner. They then investigated the dynamic performance of the trackers as a function of oscillator frequency, which showed that there is no significant degradation of the measured values for frequencies below 4 Hz (although the readings do become noisier from 4-10 Hz). Since the range of volitional head motions is well below this frequency (as we shall see in the next section) and since non-volitional motions (such as tremor) have small amplitudes, this error source should not be a significant factor for these trackers. Of course, trackers with longer measurement periods will be more sensitive to this problem and therefore should be tested in a similar fashion.

6.3.2.3. Delay-Induced Error

The general expression for bounding the delay-induced error alone is

$$b_{delay} = \|\dot{\mathbf{v}}_{head}\|\Delta t + 2\cdot\left|\sin\frac{\dot{\O}_{head}\Delta t}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \qquad (6.3.2.3.1)$$

From Section 5.4.2, the delay Δt is given by

$$\Delta t = \Delta t_{tracker} + \Delta t_{im\text{-}gen} + \Delta t_{host} + \Delta t_{sync} + \Delta t_{frame} + \Delta t_{display} \qquad (5.4.2.3)$$

If we use the standard value (for this application) of 500 mm for $\|\mathbf{v}_{S\_P''}\|$, there are three remaining variables for evaluating the error expression: the amount of delay (Δt) and the linear and angular head velocities.

For the delay numbers, there is a rather complete study of the end-to-end delays in the UNC system which was performed by Mark Mine [Mine 93]. His values for the UNC system are as follows:

$\Delta t_{tracker}$: 11.0 ms (including transmission time to the host)[34]

$\Delta t_{im\text{-}gen} + \Delta t_{host}$: 54.4 ms. This is an average and was measured for a scene consisting of only one polygon. More complex scenes will obviously take more time; this value is used as a lower bound.

Δt_{sync}: The worst case here is to just miss the beginning of frame and to wait for an entire frame time before the new image is scanned out; for NTSC, this amounts to one field time or 16.7 ms. The best case is to have perfect synchronization, in which case there is no delay.

Δt_{frame}: The worst case is a small primitive at the lower right corner of the screen; it will be drawn at the end of the field; for NTSC, this is about 16.7 ms. The best case is a primitive at the upper left, for which the delay will be nearly zero.

$\Delta t_{display}$: For CRTs, there should be no additional delays within the display itself; for LCDs, however, Mine measured at least one field time (16.7 ms) of additional delay.[35]

Using these numbers, the worst-case delay is 115.4 ms and a best-case delay is somewhere around 65.4 ms.

More data on the image-generation delay is provided by [Cohen and Olano 94], who investigated the delay in Pixel-Planes 5 as part of a project to create a low-latency rendering system. They report that the amount of delay depends not only on the number of polygons in the scene, but also their arrangement and the path of the user's head. For a small

[34] This figure is for the Polhemus Fastrak; other trackers were measured in the same analysis and had higher latency.

[35] Mine and Bishop [Bishop 93] made timing measurements on an LCD in one of the UNC HMDs and found that the LCD takes 16 to 33 ms to respond to a change in pixel value. This may be due to the switching speed of the liquid crystals or due to internal processing by the display electronics.

hardware configuration (13 graphics processors, 5 renderers), they found the following average delays for three different datasets:

Model	**Primitives**	**Mean Latency**	**Std. Deviation**
Jet	417 triangles	76.8 ms	9.8
Molecules	2,620 triangles, 1,219 spheres	74.5 ms	8.4
Head	59,592 triangles	135.5 ms	41.3

Table 6.3.2.3.1. Average latencies for various datasets

They developed a low-latency rendering system for Pixel-Planes 5 that reduces $\Delta t_{\text{im-gen}}$ to one field time, or 16.7 ms. The current implementation, however, can only handle 100-200 triangle datasets if it is to guarantee this maximum latency. While future implementations may be able to handle larger datasets, it seems unlikely that this approach will be able to handle the large (10,000 to 100,000 polygons) anatomical datasets typical in medical applications any time soon.

Thus, depending on the size of the dataset, we may see image-generation delays anywhere from 17 ms to 135 ms and beyond. I will examine the effect of delay for a range of values in order to get a feeling for the impact of delay on registration error.

We can now examine typical values for $\dot{\text{Ø}}_{\text{head}}$ and $\dot{\mathbf{v}}_{\text{head}}$. These values are certainly application-dependent: one would naturally expect fighter pilots to have higher velocity values than surgeons, for example. For this application, I conducted a study of a simulated planning session (described in detail in Section 7.2.2) and measured the angular and linear velocities of the physician's head with the Fastrak magnetic tracker. For the head velocities in the following figures, the surgeon attempted to move his head as slowly as possible while still conducting a normal examination. The goal of this part of the study was to get conservative estimates of head velocities for this application (the other data was similar; see Section 7.2.2 for details).

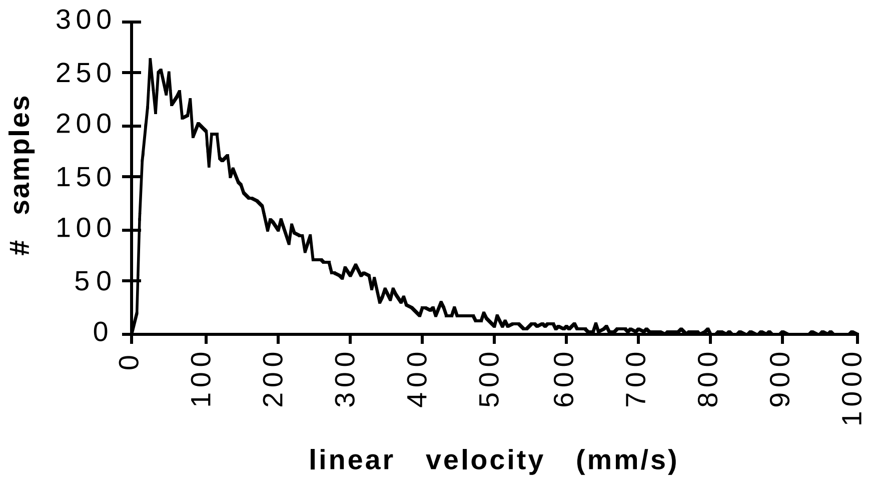


Figure 6.3.2.3.1. Histogram of linear velocity samples for head motion

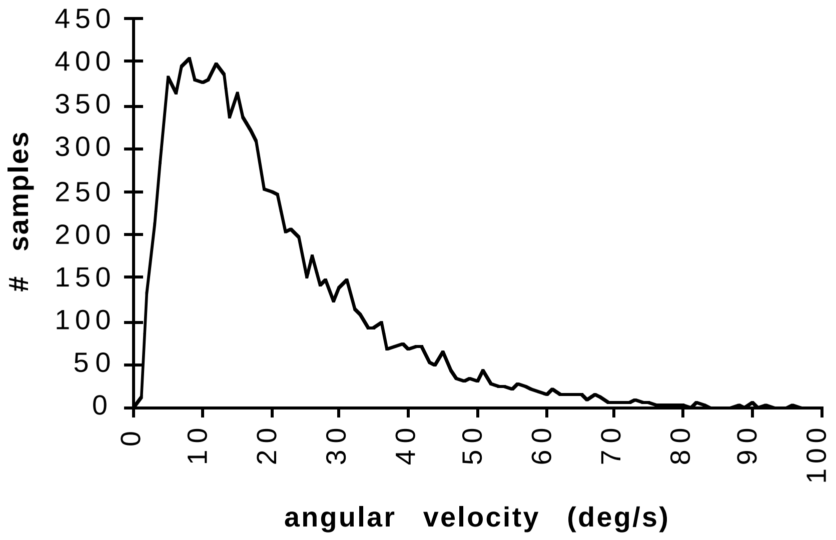


Figure 6.3.2.3.2. Histogram of angular velocity samples for head motion

The data in these charts represents about 10,000 samples taken in 3 sessions with different subjects. The mean linear velocity for these sessions was 164 mm/s and the mean angular velocity was 20 deg/s. The vertical bars in the histograms represent the number of samples in a range which was 5 mm/s for linear velocity and 1 deg/s for angular velocity.

From the data we can see that most of the head movements were slower than about 50 deg/s and 500 mm/s. This is consistent with data collected by Azuma [Azuma 95] for naive users in a demo application: the linear velocities peaked at around 500 mm/s, and most of the angular velocities were below 50 deg/s (although the peak velocities did get as high as 120 deg/s in some cases).

If we take 500 mm/s and 50 deg/s as fairly conservative upper bounds for head movement and plug them into the expression for b_{delay} for the minimum delay number for the normal Pixel-Planes rendering system (65 ms), we get

$$b_{T\_S} = 500 \text{ mm/s}\cdot.065\text{s} + 2\cdot\sin\frac{50 \text{ deg/s}\cdot.065\text{s}}{2}\cdot500 \text{ mm} = 28.4 + 32.5 = 60.9 \text{ mm}$$

which is clearly a very large error. If we use the mean velocities, we get an idea of the mean error:

$$b_{T\_S} = 164 \text{ mm/s}\cdot.065\text{s} + 2\cdot\sin\frac{20 \text{ deg/s}\cdot.065\text{s}}{2}\cdot500 \text{ mm} = 10.7 + 11.3 = 22 \text{ mm}$$

This is still quite a large error and gives an indication of just how serious a problem delay-induced registration error is. Note also that, at least for this application, the linear and angular terms contribute equally to the net registration error.

For small angles (*i.e.*, when $|\dot{\varnothing}_{head}\Delta t|$ less than about 30˚), the expression for angular delay error can be simplified to

$$b_{ang\_delay} \approx |\dot{\varnothing}_{head}\Delta t|\cdot\|\mathbf{v}_{S\_P''}\| \qquad (6.3.2.3.2)$$

which is clearly linear in all terms. For $\|\mathbf{v}_{S\_P''}\|$ = 500mm and $\dot{\varnothing}_{head}$ = 50 deg/s = 0.87 rad/s,

$$b_{ang\_delay} = 436\cdot\Delta t$$

which leads to 0.44·Δt for Δt in milliseconds, or about a half a millimeter per ms of delay. For translational delay-induced error with $\dot{\mathbf{v}}_{head}$ = 500 mm/s, we have .5·Δt for Δt in milliseconds, or again about a half a millimeter per ms of delay. Summing these gives the result that in the worst case, we get about 1 mm of registration error for every millisecond of delay.

Using the mean velocities, we get

$$b_{avg} = \left(20 \text{ deg/s} \cdot \frac{\pi}{180} \text{rad/deg}\cdot500\text{mm} + 164 \text{ mm/s}\right)\cdot\Delta t = (175 + 164)\cdot\Delta t \approx 33 \text{ mm/ms}$$

Thus, a simple rule of thumb for this application is that we can expect about 1 mm of registration error for every millisecond of delay in the worst case and $\frac{1}{3}$ mm/ms in the *average case.* Note the significance of this result: if our goal is registration to within 1 mm, unless we do predictive head tracking, the system will only have 1 millisecond to read the tracker, do its calculations, and update the displays! Even the most aggressive strategies for reducing system delay (discussed in Chapter 8) cannot hope to achieve this level of performance. *The only hope for good dynamic registration will be to use predictive head tracking.*

Now that we have examined the worst case and the average case, we can look at some more optimistic scenarios to see what the best case might be. The Polhemus Fastrak has the lowest latency of the magnetic trackers (for tracking one object) with a specified latency of 4 ms (Mine's number was somewhat higher but includes transmission time to the host). Even if there were no image-generation delay at all and we had a perfectly synchronized display system, we would still end up with 3.7 mm of registration error for the maximum head velocities and 1.4 mm for the mean velocities. If we add this delay to the 17 ms delay for the Pixel-Planes low-latency rendering system, we have $\Delta t \approx 21$ ms, which gives 20 mm for maximum velocities and 7 mm for the mean velocities.

The optimistic figures above do not take into account Δt_{frame}, the time to draw a full NTSC field. Unless this timing is compensated for (see Chapter 8), the last pixel drawn will be 17 ms later than this, which corresponds to 36 mm and 13 mm for the maximum and mean velocities. The upshot is that even in the best case (unless we attempt to use predictive methods), the registration error due to delay will usually swamp that due to other error sources.

The following figure varies both the delay and the head velocities to explore the resulting registration error.

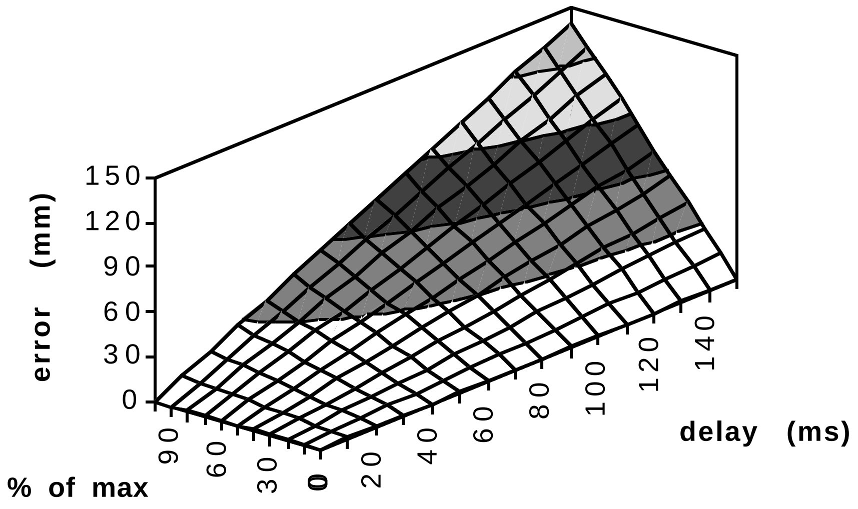


Figure 6.3.2.3. Registration error vs. delay and head velocities (maximum velocities = 50˚/s, 500 mm/s)

This chart plots b_{delay} vs. delay and percentage of maximum head velocities (linear and angular) with $||\mathbf{v}_{S\_P''}||$ = 500mm. The surface shading changes occur at 20 mm intervals. It should be clear from the chart that either high head velocity or high latency will induce significant registration error.

Observations on delay-induced registration error:

- Delay-induced error is the largest single error source. For the maximum head velocities and typical system delays, delay-induced registration error is greater than all other registration errors combined.
- Delay-induced registration error is significant even for moderate head velocities. Even small delays with small head movements can induce significant errors.
- Error depends on head velocity and amount of delay in system, implying that slow systems will require slow head movements in order to be usable.
- Linear and angular velocity terms contribute equally to the net registration error.
- Error due to angular term scales linearly with distance from head to point.

• Predictive head tracking and low-latency rendering will be essential for dynamic registration. (See [Azuma & Bishop 94] for a discussion of current work in this area.)

6.3.3. Head-Tracking Error Conclusions

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 uncalibrated trackers, and delay in propagating the information reported by the tracker through the system in a timely fashion. And, as we will see in the section on display error, tracker errors also complicate calibration procedures for locating the user's eyepoints and the virtual images of the screens. About the only good news about present-day trackers is that the dynamic error appears to be negligible.

Chapter 8 discusses some methods for combating the problems introduced by tracker error and especially delay.

6.4. Viewing Error

As we saw in Chapter 5, viewing error is the error in modeling the locations of the eyepoints which are used as the centers of projection for the left and right images. Section 5.6 gives a number of expressions for calculating this error for both monocular and binocular cases.

Recall from Section 5.6 that we can treat viewing error independently of display error but not vice versa. Therefore, this section examines viewing error alone, and the next section will treat display error in the presence of viewing error. I will begin with monocular viewing error (lateral and angular) in Sections 6.4.1 through 6.4.4. Section 6.4.5 will then consider the binocular case, which will include depth error. Finally, the last section will draw general conclusions about viewing error.

The monocular analysis is useful because it tells us how far "off" the virtual scene is from the real scene. The binocular analysis allows us to assign 3D locations to the displayed points and thereby investigate errors in depth. More importantly, it allows us to explore the warpings that the various error sources induce in the displayed objects. Of course, there is no guarantee that the human visual system behaves exactly according to the simple geometric model that I use, but it is a starting point for investigating the issue.

6.4.1. Monocular Viewing Error Overview

In Section 5.6, I derived a number of expressions for describing viewing error. The full expressions for the two monoscopic error metrics (angular and lateral error) are

$$\delta = \tan^{-1}\left(\left|\frac{z_e(x''-x_e) + (z''-z_e)\left(\frac{d'x}{d'-z''} - x_e\right)}{-z_e(z''-z_e) + (x''-x_e)\left(\frac{d'x}{d'-z''} - x_e\right)}\right|\right) \quad (5.6.2.8)$$

and

$$s_{view} = 2r\cdot\sin\frac{\delta}{2} \quad (5.1.3)$$

The approximations for the worst case for on-axis points are (respectively)

$$\delta \approx \tan^{-1}\left(\left|\frac{e\cdot z''}{e^2 + d'^2 - d'z''}\right|\right) \quad (5.6.2.9)$$

and

$$s_{view} \approx e\cdot\frac{|z''|}{d'} \quad (5.6.3.3)$$

The most generally useful of these is this last expression, although the angular error metric will prove useful as well. All of these expressions express the viewing error as a function of the magnitude and/or direction of the vector $\mathbf{v}_{E'\_E}$. The first step to interpreting them, then, is to examine the error sources that cause the actual eyepoint E to deviate from the modeled eyepoint E', which is done in the next two sections. Section 6.4.4 then derives overall monocular viewing error bounds in terms of these error sources.

6.4.2. Eyepoint Movement

As stated earlier, the point E that serves as the center of projection for the human eye is the center of the entrance pupil, which is the image of the pupil as seen from outside the eye, i.e., through the cornea.[36] This point is roughly 11 mm forward of the center of rotation (C) of the eye, as pictured below.[37]

---

[36] Note that since entrance pupil is defined by corneal optics rather than by the lens of the eye, this point should not move as accommodation changes.

[37] This value was calculated using data from [Longhurst 57] and [Boff & Lincoln 88]. The entrance pupil is about 0.5 mm forward of the pupil itself.

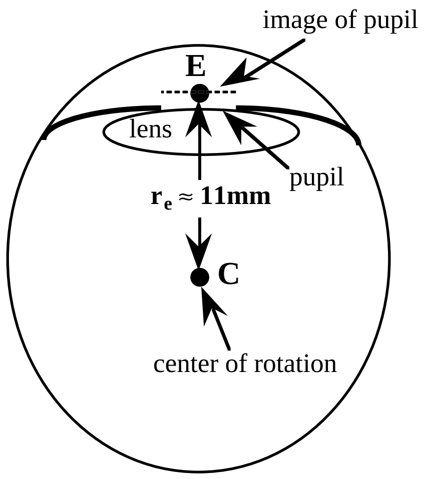


Figure 6.4.2.1. Simple schematic of eye showing center of entrance pupil, E

In the figure, I have defined r_e as a shorthand for $\|\mathbf{v}_{C\_E'}\|$. It should be clear that as the eye rotates, E moves along the surface of a sphere of radius r_e. If we start with the modeled eye position E' aligned with E looking straight ahead, we can examine the deviation of E from E' as the eye rotates. For an eye rotation by an angle ε (hereafter referred to as the *gaze angle*), the eyepoint E moves by a distance *e* from E', where

$$e = \|\mathbf{v}_{E'\_E}\| = \|\mathbf{v}_{C\_E} - \mathbf{v}_{C\_E'}\| = \|R \cdot \mathbf{v}_{C\_E'} - \mathbf{v}_{C\_E'}\| = \left|2 \cdot \sin\frac{\varepsilon}{2}\right| \cdot r_e \qquad (6.4.2.1.1)$$

This expression is just the one for the “difference vector” used in the rigid-body transformation derivations. For a 60˚ monocular field of view, the worst case would be for ε to range from -30˚ to +30˚, corresponding to an eyepoint movement of ±5.7 mm.[38]

Plugging this calculated value for *e* into the viewing error expressions already derived show that this amount of movement could induce lateral errors on the order of a few millimeters for points not in the projection plane. This error can be reduced in a number of ways: 1) an eye tracker can be used to determine gaze direction and to adjust E accordingly, 2) the gaze direction can be inferred from the image content in applications where there is an obvious, small region of interest (such as the tip of a virtual tool), or 3) an alternate center of projection can be used to reduce the error, as described next.

---

[38] Although [Kocian 88] indicates that after 20˚ of eye rotation, the user usually turns his head.

In applications for which the gaze direction cannot be predicted, there is reason to hope that eye tracking will not be necessary if the point C can be located with precision. That is, it turns out that using C as the modeled eyepoint may reduce the viewing error to a negligible level even without eye tracking.

The point C is always aligned with the true eyepoint for a point in the center of the eye's field of view (i.e., for a foveally fixated point), as shown in the figure below.

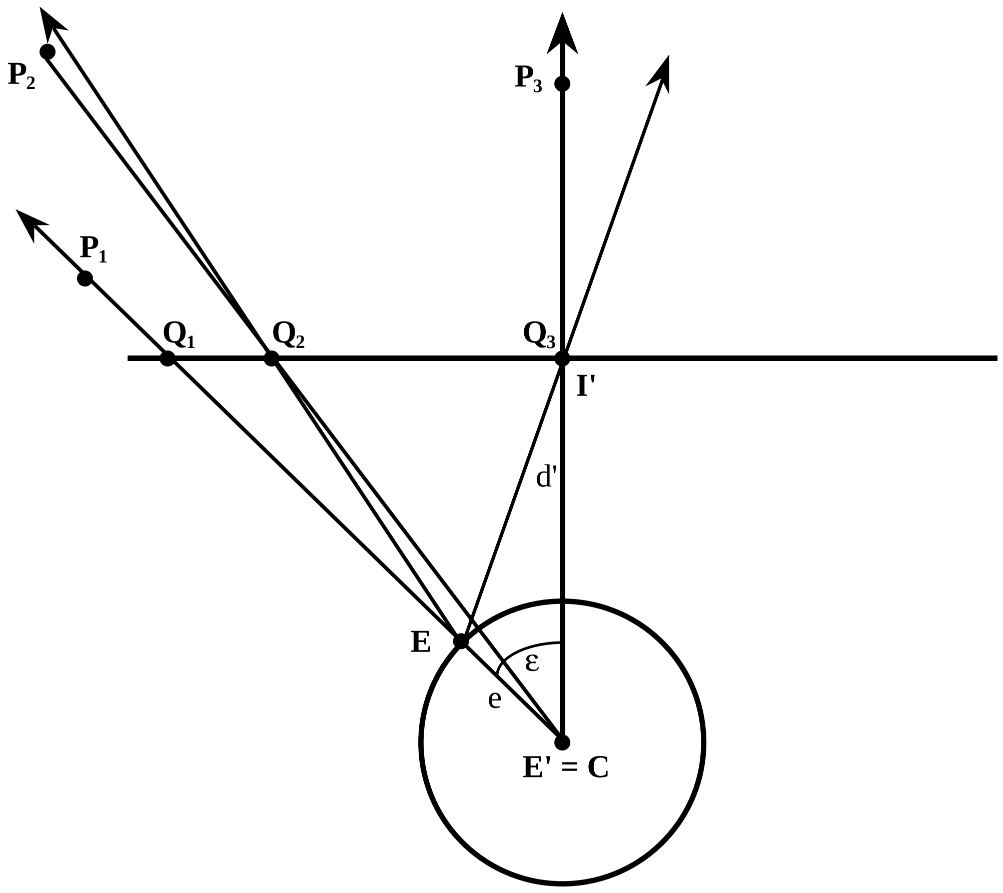


Figure 6.4.2.2. Viewing error for modeled eyepoint at center of rotation

The figure shows three points, P_1-P_3, projected using E' = C as the center of projection. As the eye rotates about C to fixate on P_1, E comes into alignment with E', Q_1, and P_1 and thus there is no viewing error for P_1. While the eye is fixated on P_1 (and Q_1), there is a slight viewing error for P_2 and a greater viewing error for P_3. Similarly, if the eye rotates to fixate on P_3 for example, E, E', Q_3 and P_3 will all fall on the same line and the viewing error for P_3 will then be zero. Thus, when the eye isn't looking at a point, it will have some amount of viewing error, but when it turns to look at the point, its viewing error goes to zero.

The next question is: How much viewing error is induced in the non-fixated points? The answer depends, of course, on the viewing parameters, but for an application such as ours, it turns out that the angular errors are fairly small. The following figure shows the angular error as a function of the direction to the displayed point (θ) and distance from the projection plane (z") for the group of fixated and non-fixated points described below.

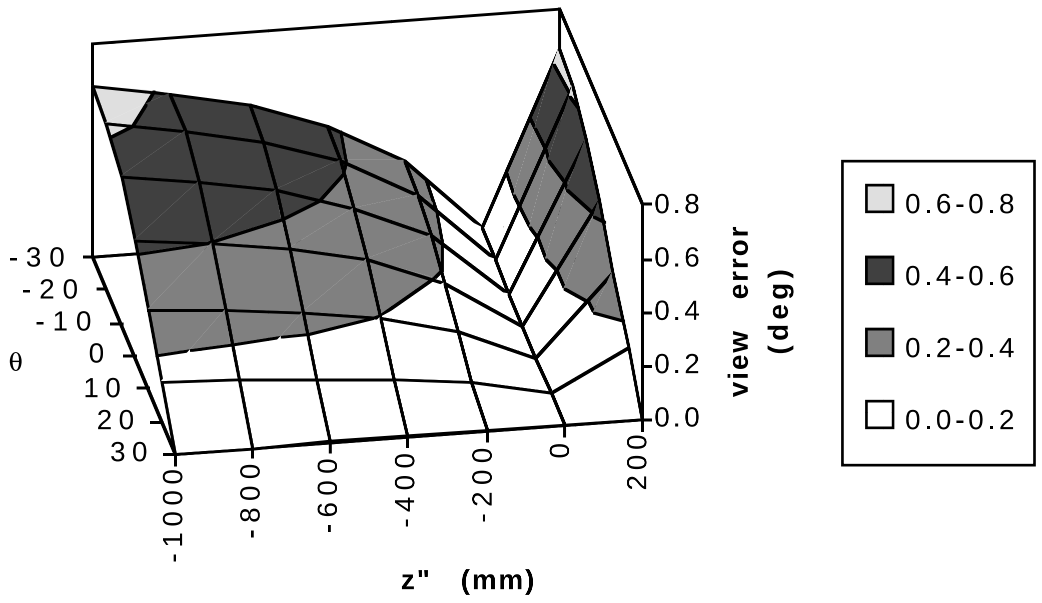


Figure 6.4.2.3. Angular viewing error due to eye movement with E' = C

This plot corresponds to the eye looking at a point at $\theta = 30^\circ$ (roughly equivalent to P_1 in the previous figure) and gives the angular viewing error for points in the range $-30 \le \theta \le +30^\circ$ and with z" coordinates in the range $+200 \le z'' \le -1000$, with $e = \|\mathbf{v}_{E'\_E}\| = 11$ mm and $d' = 500$ mm. This corresponds to the worst case for a 60˚ field of view STHMD in which the angle to the furthest point from the view direction is equal to the field of view of the system. In other words, the eye is turned to look at a point at one edge of the field of view ($\theta = \varepsilon = 30^\circ$) and the angle to the non-fixated points ($\varepsilon - \theta$) ranges from 0˚ to 60˚.

As predicted by the model, the viewing error is zero for all points within the projection plane (z" = 0) and for all points along the gaze direction ($\varepsilon = \theta = 30^\circ$). Clearly, the angular viewing error for a point is a function of the angle from the viewing direction to the point ($\varepsilon - \theta$) as well as the point's distance from the projection plane. The angular error rises comparatively quickly for closer points, but is still well under 1˚ for all points considered. For arbitrarily distant points, we can use Equation 5.6.2.10 to show that even for points infinitely distant, the angular error in this case is bounded by

$$\lim_{z'' \to -\infty} \delta = \sin^{-1}\left(\frac{e}{d'}\right) = \sin^{-1}\left(\frac{11}{500}\right) = 1.26^\circ$$

This bound applies to all viewing directions and eye directions for the given values of *e* and *d'*.

For very near points (i.e., closer than 300 mm to the eye), the angular error can approach 5˚, but this only happens in the range of eye-to-point distances that are inside of the range of comfortable focus[39] and were therefore not considered for the chart.

While the angular errors calculated here correspond to large lateral errors for distant points, because the angles corresponding to the errors are small, it is an open question whether the human eye could ever detect such errors, due to the falloff in acuity for non-foveal vision. Thus, although it remains to be confirmed by user studies, there is reason to hope that eye tracking will not be necessary for systems that can accurately locate the center of rotation of the user's eyes.[40] Another benefit to this result is that it is easier in some cases to find C than E, as we shall see in the calibration section.

6.4.3. Calibration Errors

Calibration error is akin to head-tracking error in that we are trying to determine the location of part of the user with respect to some coordinate system, a process which inevitably introduces error. Current eyepoint calibration methods attempt to locate the eyepoint in World space either by alignment with World-space reference objects with known geometry [Azuma & Bishop 94] or by camera metrology that images the eye and a reference object with known geometry [Janin et al. 93]. Given the results from the previous section, it should be clear that for systems without eye tracking, the calibration procedure should try

---

39 [Longhurst 57] cites the "least distance of distinct vision" as 250mm in front of the eye, which corresponds here to z" = 250 mm.

40 If the user is wearing eyeglasses, the point to use is the image of C as seen through the glasses; this point can be located with the use of the twin-vector eyepoint calibration method discussed in the next section.

to locate the center of rotation of the eye (C) rather than the eyepoint (E) itself. Note that this error term also includes any time-varying error sources, such as slippage of the STHMD due to head motion; such error sources will be treated simply as contributors to the overall error bound for determining C. This section begins with an analysis of the interaction between eyepoint movement and calibration error, then discusses current calibration techniques, and finishes with an analysis of viewing error given the error bounds and equations derived.

6.4.3.1. Calibration Error Analysis

Given that we are trying to locate C rather than E, we are interested in the effect of errors in locating C on the vector $\mathbf{v}_{E'\_E}$. The figure below depicts the situation.

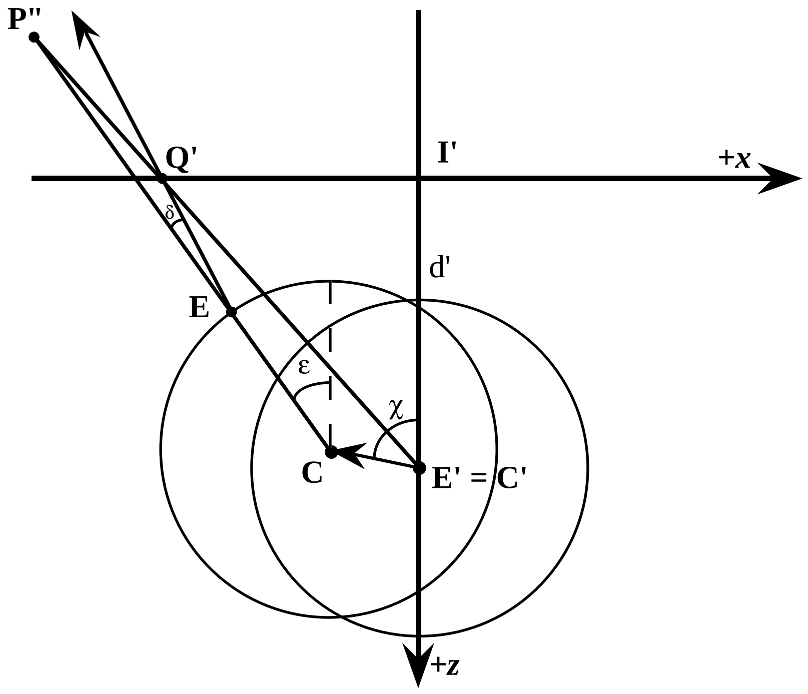


Figure 6.4.3.1.1. Effect of eyepoint calibration error on viewing error

Here, C is the real center of rotation of the eye, but the calibration procedure has incorrectly identified C' as the center. The orientation of the calibration error vector $\mathbf{v}_{C'\_C}$ is given by the angle χ, the gaze angle is ε, and the angular viewing error is given by δ. The net error vector for the eyepoint is

$$\mathbf{v}_{E'\_E} = \mathbf{v}_{C'\_C} + \mathbf{v}_{C\_E} \tag{6.4.3.1.1}$$

As in Section 5.6, we can use x_e and z_e to denote the components of $\mathbf{v}_{E'\_E}$:

$$x_e = -c \cdot \sin\chi - r_e \cdot \sin\varepsilon \qquad (6.4.3.1.2)$$

$$z_e = d' - c \cdot \cos\chi - r_e \cdot \cos\varepsilon \qquad (6.4.3.1.3)$$

where

$$c = \|\mathbf{v}_{C'\_C}\| \qquad (6.4.3.1.4)$$

$$r_e = \|\mathbf{v}_{C\_E}\| \qquad (6.4.3.1.5)$$

These expressions for the components of $\mathbf{v}_{E''\_E}$ are the same as those derived in Section 5.6.2 except for the added term due to calibration error. If we make no assumptions about the behavior of ε and χ, we can vary them from 0˚ to 360˚ and note the worst-case error. This turns out to be the same as the general case derived in Section 5.6.2 with $e = r_e + c$, which corresponds to the case when the vectors $\mathbf{v}_{C'\_C}$ and $\mathbf{v}_{C\_E}$ are aligned. As we found then, the worst case is for on-axis points when both vectors are at 90˚ or 270˚. Alternatively, we can limit the amount of eye rotation to be within the field of view of the system and calculate the error for this limited case. In either case, we need bounds on the magnitude of the calibration error, *c*, which is discussed in the next section.

6.4.3.2. Calibration Methods

Estimates for the error in locating C (or E) are difficult to derive because calibration procedures by necessity attempt to determine the values of multiple system parameters at the same time. In some cases, optimization procedures are used to determine the parameter values which minimize the net system error, further complicating the process of estimating the error bound for any single parameter. Finally, C (or E) is usually measured in World space but then expressed in Sensor space, which introduces tracker error into the process. Because of these complications, my approach in this section is to examine a number of calibration procedures and derive ballpark estimates of the error in locating C or E.

A very intuitive and simple method of locating the eyepoint in World space is to align two coplanar lines so that they intersect at the eye, as shown in the figure below.

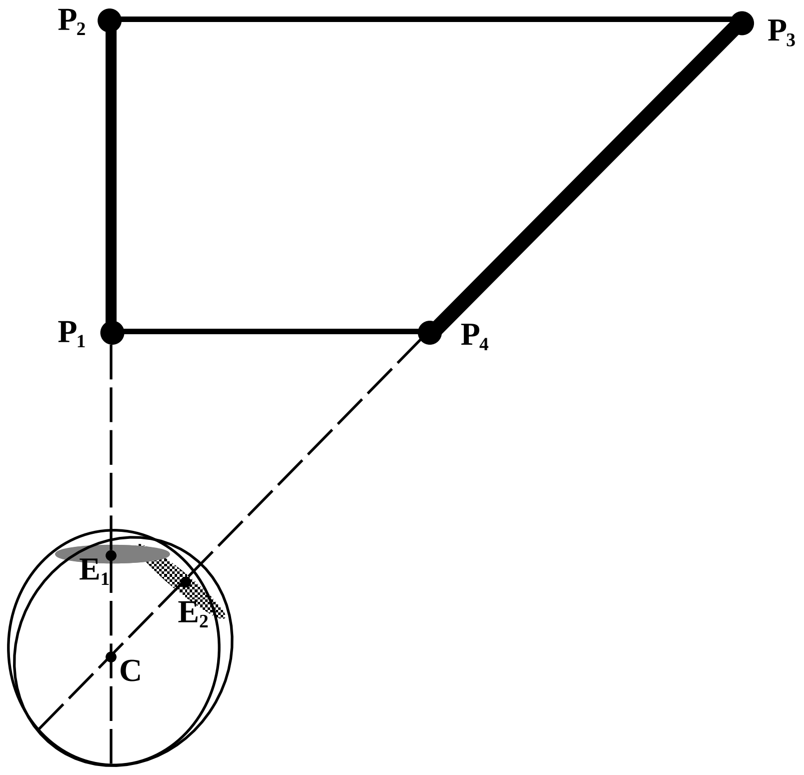


Figure 6.4.3.2.1. Twin-vector calibration method

As the figure shows, since the eye will normally swivel in order to line up with each vector separately, the point located in such a method is actually the center of rotation (C) rather than the eyepoint (E), which is what we actually want. It should also be clear that a small angle between the two vectors will make the procedure very sensitive to error in the inputs; an angle of 90˚ would be optimal since it minimizes this sensitivity.

This twin-vector method is used by [Azuma & Bishop 94], and a similar method has been used by [Caudell and Mizell 92]. With Azuma and Bishop's method, the user aligns his eyepoint with the edge of a box structure and then with two nails, at which point one or more tracker readings are taken. The calibration points have already been digitized by the tracker, so their intersection (and thereby the eyepoint) is known in Tracker space (to within the precision of the tracker). The error in this procedure is due to 1) any alignment error made by the user due to head tremor or whole-body movement and 2) tracker error in measuring the four calibration points and in locating the Sensor CS at the moment of alignment. Azuma and Bishop report an average standard deviation of about 5 mm for the location of their Eye CS, most of which was in the *z* coordinate. My experience with calibrating the prototype system (described in Chapter 7) was similar.

[Janin et al. 93] have adopted a method for eyepoint calibration that uses dual cameras to image the pupil and determine the eyepoint location using camera metrology. Note that since the pupil is seen through the cornea, the camera is imaging the entrance pupil and therefore the center of the imaged pupil corresponds to the true eyepoint E. A problem with this method is that it is only used as a startup calibration procedure (i.e., they do not use an eye tracker) and the calculated eyepoint is therefore incorrect as soon as the eye rotates. Since the eyepoint is calculated in World space and then expressed in Image space (which is defined relative to Sensor space), this method is also limited by the accuracy of the tracker (a Polhemus Isotrak). They report an accuracy of 0.05" (1.3 mm) for locating the eyepoint using the dual-camera method.

This data suggests that the center of rotation of the eye can be located precisely within World space, but that the tracker error and other uncertainties in the calibration process increase the uncertainty of the vector when expressed in Sensor or Image space. For the analysis that follows, I will use a value of 5 mm as a rough estimate for *c*.

6.4.4. Monoscopic Viewing Error Analysis

This section analyzes the error due to calibration error and eye rotation for both fixated and non-fixated points. For fixated points, the most useful error metric is probably the lateral error, since it gives a measure of how much misregistration the user is likely to perceive. For non-fixated points, the angular registration error is probably more useful since it gives a better idea of the visual angle subtended by the misregistered features and therefore a better idea of whether the misregistration will be perceptible for non-foveal vision. I will begin by analyzing the case of fixated points, since they are clearly of greatest interest.

To fully analyze viewing error induced by calibration error and eye rotation, one must consider a surprisingly large parameter space. That is, even when we fix the eye-screen distance *d'*, the eye radius r_e, and the field of view of the system, we still have a five-dimensional space defined by the following parameters:

θ:	view direction (related to x")
z'':	depth of point
χ:	calibration error angle
c:	calibration error magnitude
ε:	gaze angle

Since we are beginning with fixated points, let us assume that ε is roughly equal to θ (it would be exactly equal if E = E' = C'). We can also fix *c* at 5 mm and begin with the most common case of on-axis points, which gives a plot like the following for s_{view}:

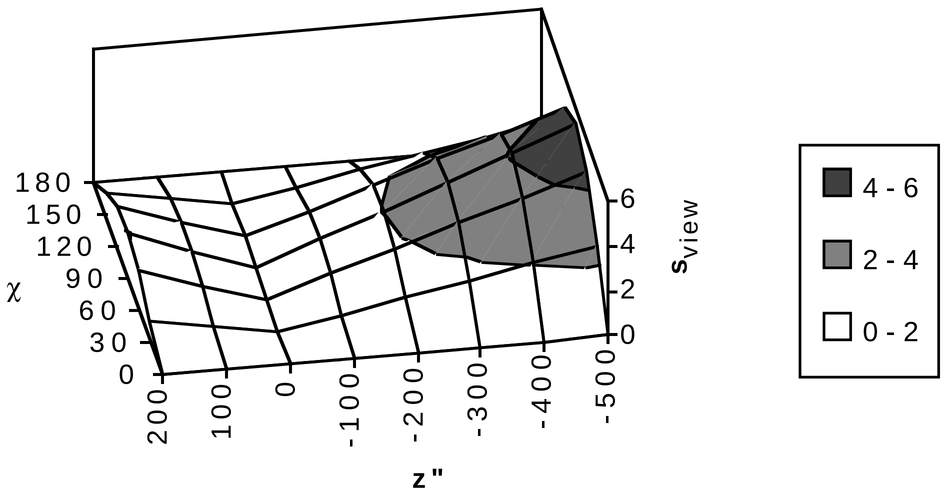


Figure 6.4.4.1. Lateral viewing error (in mm) for on-axis points vs. χ (degrees) and z" (mm)

Because of symmetry, we need only consider $0^\circ \le \chi \le 180^\circ$. We see here that the worst lateral error is for $\chi = 90^\circ$, which is consistent with the behavior seen for α in Section 5.6; that is, the worst viewing error is when the actual eyepoint is ±90˚ to the viewing direction. Along curves of constant χ the behavior is essentially linear in *z"*; in the case where $\chi = 90^\circ$, the lateral error is approximately described by

$$s_{view} \approx c \cdot \frac{|z''|}{d'} \tag{6.4.4.1}$$

which is the same as Equation 5.6.3.3 with *e = c*. This means that the eye rotation in this case does not substantially contribute to the viewing error since $\mathbf{v}_{C\_E}$ lies more or less along the projector. This is indicative of the general behavior in which eye rotation will lessen the error for fixated points and worsen the error for non-fixated points.

If we assume that our depth range is bounded by the human head depth (i.e., distance from nose to back of head), we can use the 95th percentile value of 208 mm from [Woodson & Conover 64] as a reference for how large the *z"* values are likely to get. When the projection plane is perfectly centered in the patient's head, the maximum *z"* value will be about 100 mm, corresponding to about 1 mm of lateral error for the given values of the other parameters. Doubling z'' to 200 mm also doubles the lateral error to 2 mm. Conversely, if we reduce c to 2.5 mm, we get half the lateral error in each case. Finally, note that calibration methods should, where possible, strive for accuracy in the lateral positioning of C rather than the precise depth placement, since the sensitivity of s_{view} to errors in the x and y directions is much greater than that for z.

The off-axis case for fixated points turns out to be much the same as for on-axis points. That is, if the eye rotates to look at off-axis points, the worst case is when $\theta - \chi = \pm 90^\circ$, and the lateral error is nearly identical (less than 1% difference) to the on-axis case. Furthermore, the lateral error is at its maximum for on-axis points when $\chi = 90^\circ$, just as we found in Section 5.6. Thus, if we assume the worst-case orientation for the calibration error, we can use Equation 6.4.4.1 to bound the viewing error for the off-axis case as well. Note also that if we had chosen the simpler approach of letting $e = r_e + c$ (as suggested in Section 6.4.3.1) we would have gotten an unnecessarily pessimistic error bound for the fixated case.

We can now examine the case of non-fixated points. The following figure shows the behavior of the angular viewing error (δ) as a function of viewing direction (θ) and gaze direction (ε).

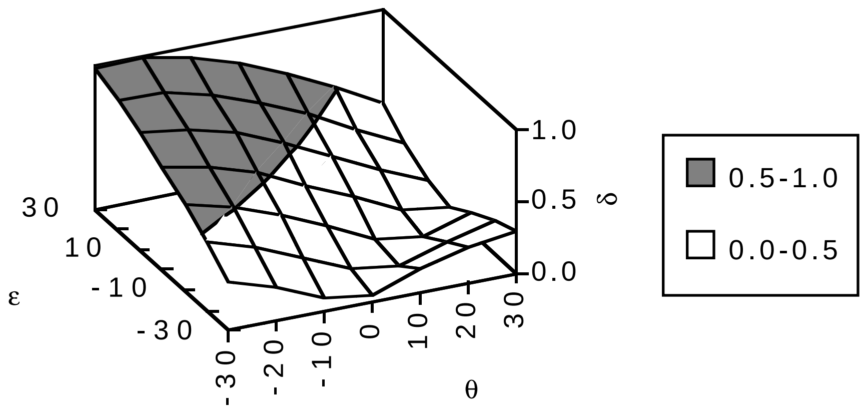


Figure 6.4.4.2. δ as a function of ε and θ for worst-case χ (all angles in degrees)

In this plot, $\chi = \theta + 90^\circ$ for all θ to give worst-case values for δ; $c = 5$ mm, d' = 500 mm as before. Note that the viewing error is less than 1˚ even in the worst case, which suggests that if the viewing error can be made acceptable for fixated points, the non-fixated points should not be noticeably misregistered (although this needs to be confirmed by user studies).

We can now move on to the binocular case in order to examine depth effects and object distortions induced by error in modeling the eyepoints.

6.4.5. Binocular Viewing Error

Now that we have treated the monocular case and have a good model for how the various error sources contribute to the lateral error, we can examine depth errors and object warpings. Because the equations for the binocular case are rather complex, I will only examine two special cases: pure rigid-pair translation and pure interpupillary distance (IPD) error. Other cases can, of course, be analyzed with the equations from Section 5.6.4.

The following equations from Section 5.6.4 will be useful for the next two sections:

$$x^* = \frac{x'' + \frac{z''}{d'}\frac{i'}{i}\left(\frac{x_{LE}+x_{RE}}{2}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.10}$$

$$y^* = \frac{y'' + \frac{z''}{d'}\frac{i'}{i}y_{LE}}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.11}$$

$$z^* = \frac{z''\left(\frac{d}{d'}\right)\left(\frac{i'}{i}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \tag{5.6.4.12}$$

$$b_{view} = \frac{\sqrt{\left(\frac{i'}{i}\right)^2(x_m{}^2+y_{LE}{}^2+(d-\frac{i'}{i}d')^2) + \left(\frac{i'}{i}-1\right)^2(x''^2+y''^2+z''^2) - 2\frac{i'}{i}\left(\frac{i'}{i}-1\right)(x_m x''+y_{LE}y''+z''(d-\frac{i'}{i}d'))}}{\frac{d'}{z''} + \left(\frac{i'}{i} - 1\right)} \tag{5.6.4.14}$$

where $x_m = \left(\frac{x_{LE}+x_{RE}}{2}\right)$ is the *x* coordinate of the midpoint of the line segment between LE and RE.

These equations describe the location of P* and the error bound for viewing error alone for translational error in the eyepoints as a pair and/or IPD error. The next two sections will examine the implications of these equations for two common types of error.

6.4.5.1. Rigid-Pair Eye Movement

Probably the simplest case of viewing error is case of constrained rigid-pair motion discussed in Section 5.6.4. In this case, the interpupillary line (the line joining the eyepoints) can translate in any direction so long as it does not rotate; i.e., the motion can be any combination of left/right, up/down, fore/aft. Anyone who has viewed a stereo image with polarized glasses has probably experienced this type of error: As you move your head away from the assumed viewpoint, the objects seem to turn to follow you, and they warp as a result. [Hodges & Davis 93] discuss this error briefly as the effect of tracker error for head-tracked stereo displays and point out that it causes shears, compressions, or elongations depending on the type of head movement. [Rolland et al. 95] derive expressions (which agree with those derived here) for describing the amount of error in the *x*, *y*, and *z* directions for this case and make similar qualitative observations based on these expressions.

We can analyze this in more detail using the model from Chapter 5. For this case, the equations for P* given above simplify to

$$x^* = x'' + \frac{z''}{d'} x_m \tag{6.4.5.1.1}$$

$$y^* = y'' + \frac{z''}{d'} y_{LE} \tag{6.4.5.1.2}$$

$$z^* = z''\left(\frac{d}{d'}\right) \tag{6.4.5.1.3}$$

where $x_m = \left(\frac{x_{LE}+x_{RE}}{2}\right)$. These equations make it clear that *x/y* rigid-pair motion induces a *z* shear in *x* and *y* and that fore-aft motion will induce a compression or elongation of objects along the *z* axis. Note that the error induced by the eye movement is along the same axis as the eye movement itself: That is, left/right eye movement induces right/left viewing error and similarly for up/down and fore/aft. Combinations of fore/aft and left/right/up/down movements result in combinations of the above warpings. In fact, circular clockwise motion of the eyepoints about their modeled locations induces counterclockwise registration error in the displayed points (and vice versa).

The 3D error bound derived in Section 5.6.4 is comparatively simple:

$$b_{view} = e \cdot \frac{|z''|}{d'} \tag{5.6.4.17}$$

where *e* in this case is just

$$e = \sqrt{x_m{}^2 + y_{LE}{}^2 + (d-d')^2} \tag{6.4.5.1.4}$$

This is the same as the bound that we found for the lateral error in the monoscopic case, which means that the overall error in 3D (including depth error) is no worse than the lateral error for this restricted case. Thus, in contrast to the other binocular error bounds, this one does not go to infinity for finite inputs, which means that this sort of viewpoint error cannot induce infinite depth error. It is also worth noting that in this case the expression above specifies exactly the amount of registration error caused by a rigid-pair movement of magnitude *e* rather than (as with the other expressions derived so far) just giving an upper bound on the registration error. That is, in this case, if the eyepoints move by a distance *e*, the registration error will be exactly $e \cdot (|z''|/d')$.

The figure below shows the case of left/right motion inducing a shear distortion.

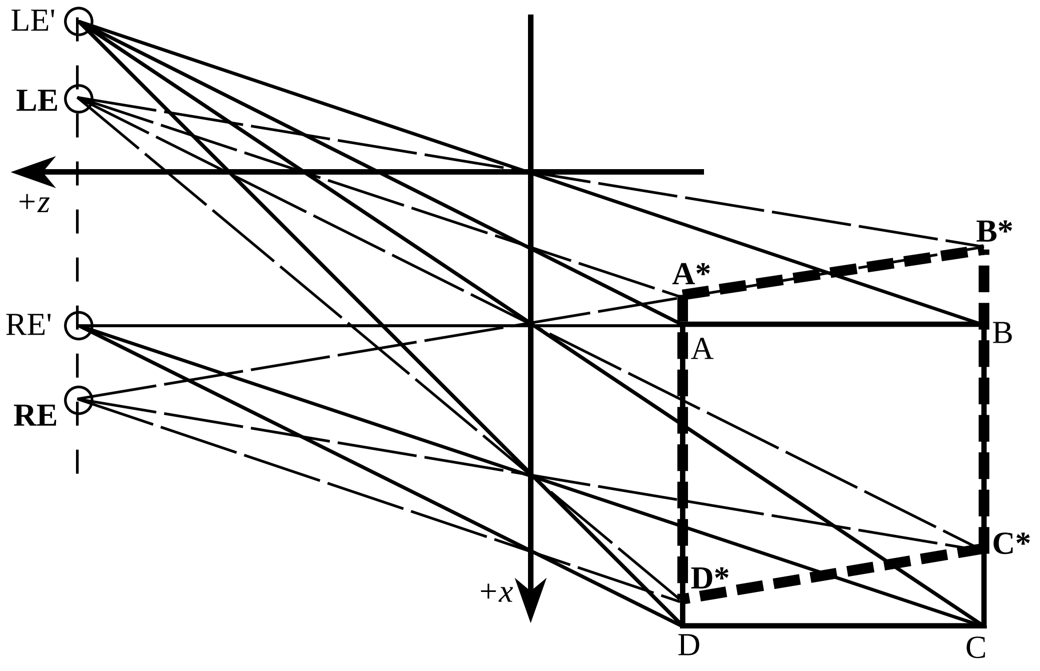


Figure 6.4.5.1.1. Shear induced by left/right rigid-pair motion

Here, the eyes have moved in the $+x$ direction, which causes the modeled square ABCD to warp into the quadrilateral $A^*B^*C^*D^*$. Note that $\|\mathbf{v}_{A*\_A}\| = \|\mathbf{v}_{D*\_D}\|$ and $\|\mathbf{v}_{B*\_B}\| = \|\mathbf{v}_{C*\_C}\|$, since these points have the same z'' coordinate and that the points further in z are displaced more, as predicted by the model. For fore/aft movements, the effect is a compression (or elongation) of objects, as pictured below.

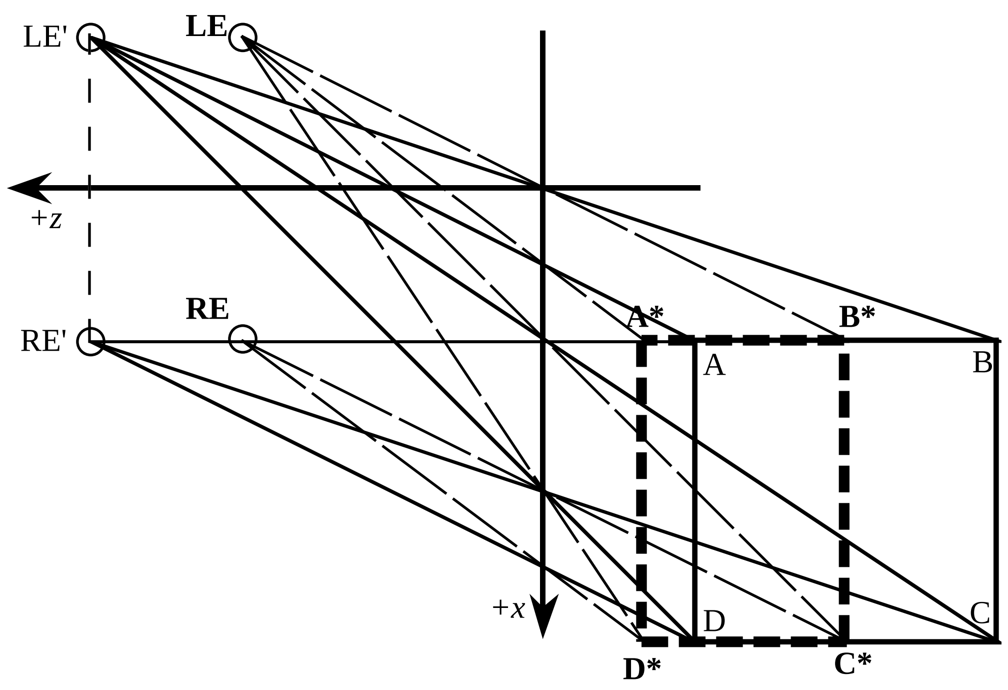


Figure 6.4.5.1.2. Compression of an object due to fore-aft motion

Again, the amount of misregistration for each point is exactly predicted by Equation 5.6.4.19. The effect for eye movement away from the screen image is an elongation in a similar fashion to the compression shown above.

Since the error expression for this case is the same as that for lateral error, the error bounds for depth error are the same as those for the linear error.

6.4.5.2. IPD Mismatch

Another common example of viewing error is when the user's IPD (interpupillary distance) is different from that used by the software (and/or hardware). For this section, I will use the term *IPD* somewhat loosely, since I will often use it to represent the separation of the centers of rotation of the eyes rather than that of the eyepoints themselves. In either case, the errors induced by mismodeling the separation are very similar and can be quite significant. Error in the modeled IPD is normally due to calibration error (or due to convergence for systems that do not use C for the center of projection[41]).

[41] [Hodges & Davis 93] has a chart of IPD change vs. convergence angle (although they use the first nodal point as the eyepoint). This is not a problem if C is used as the center of projection, however.

The equations for the coordinates in this case are

$$x^* = \frac{x''}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \qquad (6.4.5.2.1)$$

$$y^* = \frac{y''}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \qquad (6.4.5.2.2)$$

$$z^* = \frac{z''\left(\frac{i'}{i}\right)}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)} \qquad (6.4.5.2.3)$$

The error bound for this case is

$$b_{\text{view}} = \frac{\sqrt{x''^2 + y''^2 + (d' - z'')^2}}{1 + \frac{d'}{|z''|\left(\frac{i'}{i} - 1\right)}} \qquad (5.6.4.16b)$$

It turns out that the error vector $\mathbf{v}_{P''\_P*}$ for this case has a fairly simple form which lends some understanding to the nature of pure IPD error:

$$\mathbf{v}_{P''\_P*} = \kappa\cdot\mathbf{v}_{\mathbf{H\_P''}} = \left(\frac{1}{1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right)}\right)\cdot\begin{bmatrix} x'' \\ y'' \\ (z''-d') \end{bmatrix} \qquad (6.4.5.2.4)$$

where H is the midpoint of the interpupillary line along the +*z* axis. This means that in the presence of IPD error, the apparent point P* will appear along the line joining P'' and H, as shown in the figure below.

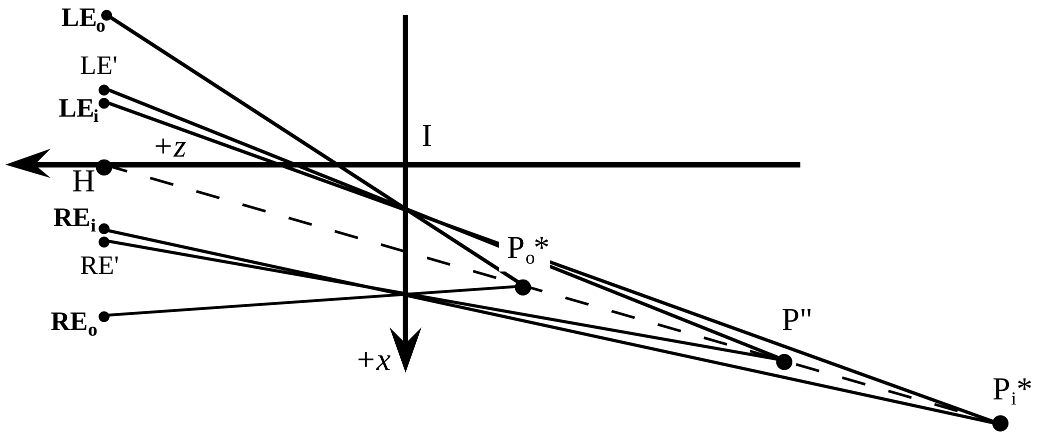


Figure 6.4.5.2.1. The effect of pure IPD error

Clearly, κ (and therefore the 3D error) can become infinite for certain points and certain IPDs. The actual IPD must be less than the modeled IPD in order to have infinite depth error[42], and the error tends to increase with z''. Note also that any one of the coordinates of P^* can become infinite, corresponding to projectors that become parallel and thus meet only at infinity. The depth error is therefore impossible to bound in general for this case, although we can bound it for certain limited cases.

If the real IPD is smaller than the modeled IPD, the objects will (theoretically at least) appear stretched in depth away from the projection plane. Points in front of the projection plane will be stretched away from it and toward the user; points beyond the projection plane will be stretched away from both, as shown in the following figure. If the IPD is greater than the modeled IPD, objects will appear compressed toward the projection plane.

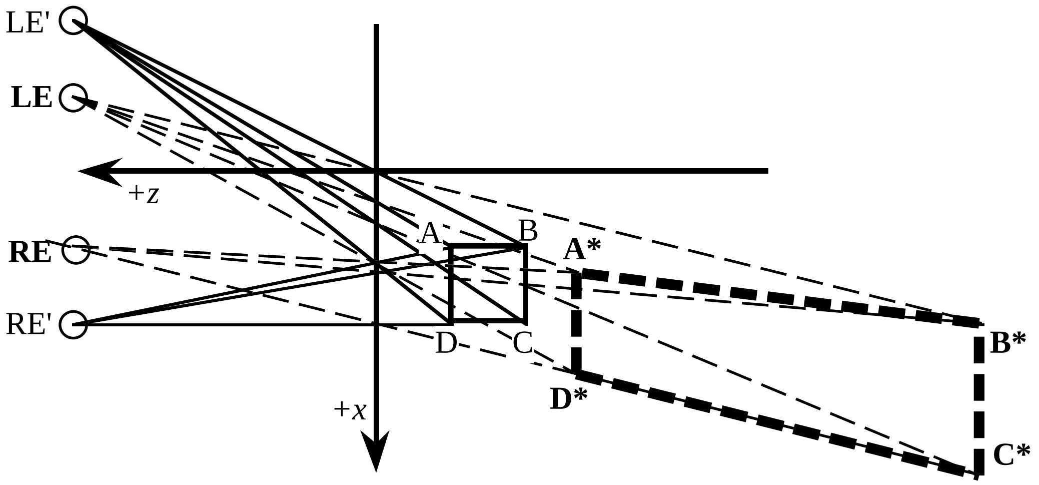


Figure 6.4.5.2.2. Real IPD smaller than modeled IPD

Note that the lateral error for each point is generally much smaller than the depth error; this is generally the case for IPD error.

I will begin the analysis by plotting the depth error for a range of IPDs from 5th to 95th percentiles [Woodson & Conover 64], in the chart below.

42 This is because the denominator can only go to zero for i' > i, since for non-clipped points, $d' \geq z'' \geq -\infty$, which means that $1 + \frac{z''}{d'}\left(\frac{i'}{i} - 1\right) = 0$ only when $i' > i$ and $z'' < 0$.

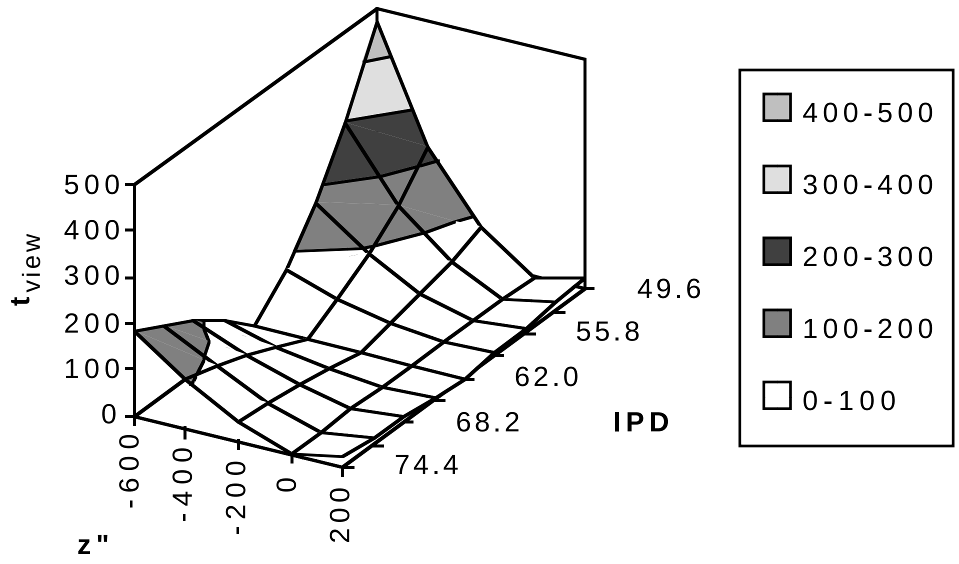


Figure 6.4.5.2.3. Depth error for a range of IPDs (all distances in mm)

For this chart, d' = 500 mm and θ = 0˚; the modeled IPD = 62 mm and the actual IPD varies ±20% in increments of 5%. Note how rapidly the depth error grows as *z"* increases, reaching nearly 500 mm for the smallest IPD at only 600 mm beyond the image plane. Even for more modest errors, the errors are quite large: for a 5% error in IPD (IPD = 58.9 mm), a point only 200 mm beyond the projection plane has a 15 mm depth error. In contrast, the lateral error for the same set of IPDs is small by comparison:

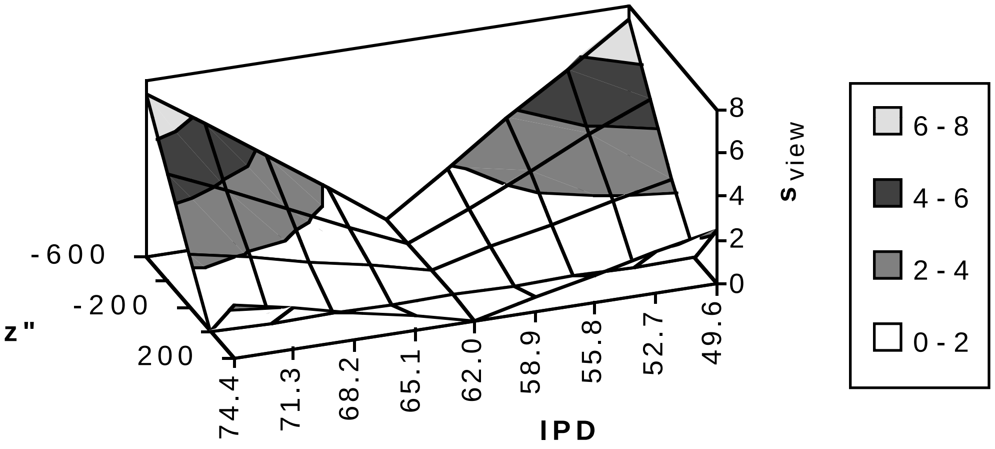


Figure 6.4.5.2.4. Lateral error vs. IPD error (all distances in mm)

Note that over the same range as that used above, the maximum lateral error is less than 8 mm. For the 5% error cited above, the corresponding lateral error is only 0.6 mm.

Given the sensitivity of the depth error to errors in modeling the IPD, we have a strong incentive to measure the IPD as accurately as possible. While most current calibration procedures derive the IPD as a byproduct of locating the eyepoints, the sensitivity shown here argues for a further calibration step in which the IPD is measured directly with a pupilometer (a device used by opticians for precisely measuring the IPD). This measurement is purely anatomical and therefore is unaffected by tracker error or calibration error in the rest of the system; it should therefore be possible to make the measurement with considerable precision (within 0.5 mm or better). Then, even if we have error in where the eyepoints are with respect to the rest of the system, we at least have an accurate value for the IPD.

Any calibration errors in this case will affect only the modeled location of the eyes as a rigid pair, inducing the sorts of error described in the previous section.

If we examine the depth error for small IPD errors and for points close to the projection plane, we get the following chart:

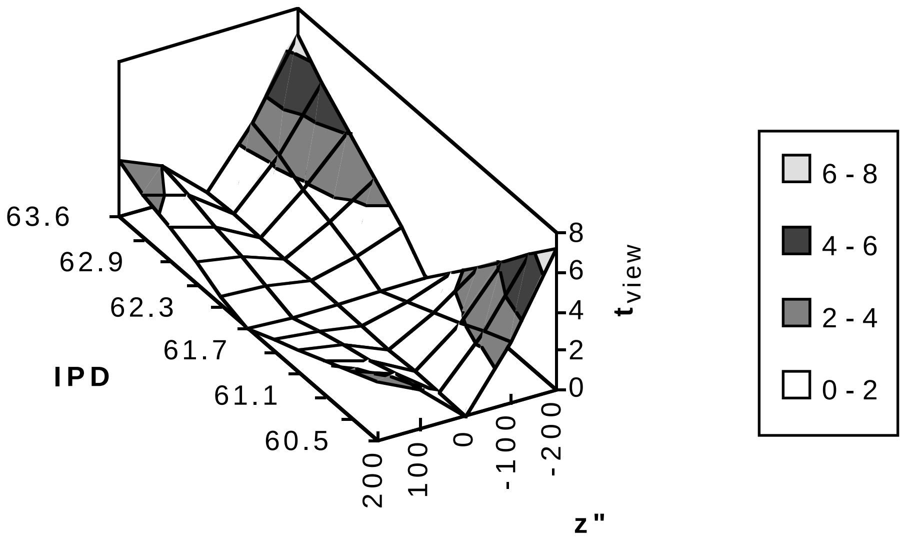


Figure 6.4.5.2.5. Depth error for small IPD error and small z"
(all distances in mm)

Here the IPD error is plotted in 0.5% increments for on-axis points within 200 mm of the projection plane. Even in this small range, the depth error can reach 6 mm for errors of 1.5 mm and points only 200 mm beyond the projection plane. As I noted earlier, it may well be the case that these theoretical errors will not be a problem in the presence of other, correctly rendered depth cues, but it is a matter worthy of further study.

6.4.6. Viewing Error Conclusions

In general, viewing error is not likely to be a major error source for carefully calibrated systems, especially if the eyes' centers of rotations are used as the viewpoints (since this should essentially eliminate eye rotation as an error source). The lateral errors induced by calibration error should be on the order of 1-2 mm, and if an independent IPD measurement is incorporated, the calibration error should manifest itself largely as rigid-pair viewing error. The depth error sensitivity is very large, but careful calibration and the presence of other depth cues may serve to mitigate its presence.

Summary and observations:

- Lateral error increases linearly and symmetrically with screen-image-to-point distance in both directions.
- For points in projection plane (z" = 0), viewing error has no effect.

- Viewing error is worst for on-axis points with $\mathbf{v}_{E'\_E}$ perpendicular to viewing direction.
- Using the eye's center of rotation as the eyepoint can eliminate the viewing error due to eye rotation (for fixated points).
- IPD error can induce very large depth errors, although they may not be perceived as such.
- Separate calibration of IPD should significantly reduce depth error.
- Rigid-pair motion of the eyepoints relative to their modeled locations induces shears and/or compression/elongations of displayed objects. The amount of error is exactly related to the amount of input error, and the bounds for lateral and depth error are the same for this case.
- Tracker error is a significant error source in eyepoint calibration procedures and limits the accuracy of many current calibration methods.
- For arm's-length applications, the screen image will need to be at the approximate depth of the virtual objects so that the user can focus on both the real and virtual objects. In this case, z" will tend to be small for points in the region of interest.
- Systems displaying objects at different depths may need to have an adjustment for screen-image distance (preferably one that adjusts automatically depending on the user's gaze direction). This would have the double benefit of minimizing viewing error and making the accommodation distance for the virtual and real objects the same.
- While it might seem that letting $d \rightarrow \infty$ (i.e., using collimated images) would reduce viewing error to zero, the situation is somewhat more complex. *z* is measured relative to the virtual image (not the eye), so when *d* becomes large, *z* also becomes large (for relatively close points). Thus, moving the projection plane to infinity will make $z/d \rightarrow 1$ for near points, inducing lateral errors approaching *e*. For close work with shallow depths of field, this is probably not desirable. However, for applications requiring a large range of depths, putting the virtual image at infinity has the advantage of capping the lateral error at a value equal to *e*, while a smaller value for *d* can induce large lateral errors for $z/d >> 1$.
- The fact that the registration error due to viewing error is zero in the projection plane can be used for calibration procedures. That is, one can verify alignment of a real object with the virtual-image plane by moving the viewpoint during the alignment

procedure; when the registration error goes to zero (or some minimum), the real and virtual planes are aligned. I used this technique for calibrating the UNC system; further details are in Section 7.3.

6.5. Display Error

As was done for viewing error, this section starts with an overview, then moves on to bounds on the display error sources, then to the monocular and binocular analyses, and finishes with a section on conclusions/implications.

6.5.1. Overview

As explained in Section 5.7, there are two main components to the display error model: 1) For a given amount of error q in the image plane, how much registration error is induced? and 2) What are the contributions to q?

The exact expressions for display error in terms of q in the presence of viewing error are

$$\gamma = \tan^{-1}\left(\left|\frac{q\, z_e}{z_e^2 + (x_{Q'} - x_e)^2 \pm q(x_{Q'} - x_e)}\right|\right) \qquad (5.7.2.9)$$

and

$$s_{v\&d} = 2r \sin\left(\frac{\delta+\gamma}{2}\right) \qquad (5.7.3.1)$$

where δ is the angular viewing error and x_e and z_e incorporate any viewing error. The approximate expression for lateral display error alone is

$$s_{display} \approx q\cdot\left(1 - \frac{z''}{d'}\right) \qquad (6.5.1.1)$$

The contributions to q are given by

$$q = \|\mathbf{v}_{Q'\_Q^{\dagger}}\| = \|\mathbf{v}_{I'\_Q^{\dagger}} - \mathbf{v}_{I'\_Q'}\| = \|T_{Proj} \cdot T_{I'\_I} \cdot (\mathbf{v}_{I\_Q''} + \mathbf{v}_{Q''\_Q}) - \mathbf{v}_{I'\_Q'}\| \qquad (5.7.7.1)$$

and

$$\mathbf{v}_{Q''\_Q} = \begin{bmatrix} (m-m')x_{G'} + m\, x_{G'\_G} + k\, m^3 r_G^2 (x_{G'} + x_{G'\_G}) + \frac{\alpha r_G + ßkr_G^3}{3}(x_{G'} + x_{G'\_G}) \\ (m-m')y_{G'} + m\, y_{G'\_G} + k\, m^3 r_G^2 (y_{G'} + y_{G'\_G}) + \frac{\alpha r_G + ßkr_G^3}{3}(y_{G'} + y_{G'\_G}) \end{bmatrix} \qquad (5.7.9.3.16)$$

These two equations contain terms corresponding to the following display error sources: image misalignment, aliasing, display-device nonlinearity, distortion, and lateral color. The effect of the optics is included in the second equation.

At this point, I will make a few general observations about the behavior of display error, after which I will treat each of the display error sources in turn and derive approximate bounds for the contributions each makes to *q*.

General observations on display error:

- Unlike viewing error, which is zero at the projection plane, display error is zero at the eyepoint and increases with distance from the eye.
- Lateral display error is roughly linear in *q* and eye-to-point distance.
- Because *q* is the within-image plane error magnitude, it is by itself a good indicator of registration error, since many points will be in or near the projection plane (this is in contrast to the viewing error *e*, whose effect on the registration error is less direct).
- Most of the display error sources have the potential to cause a difference in the projected *y* values for a point, which is known as *dipvergence*. Even small amounts of dipvergence make stereo fusion difficult or impossible. In a similar vein, errors that cause the images to be of different sizes or shapes can induce disturbing visual effects.

A plot of the lateral error (using the exact formula) as a function of *q* will give us a reference for interpreting the numerical bounds on *q* derived in the following sections:

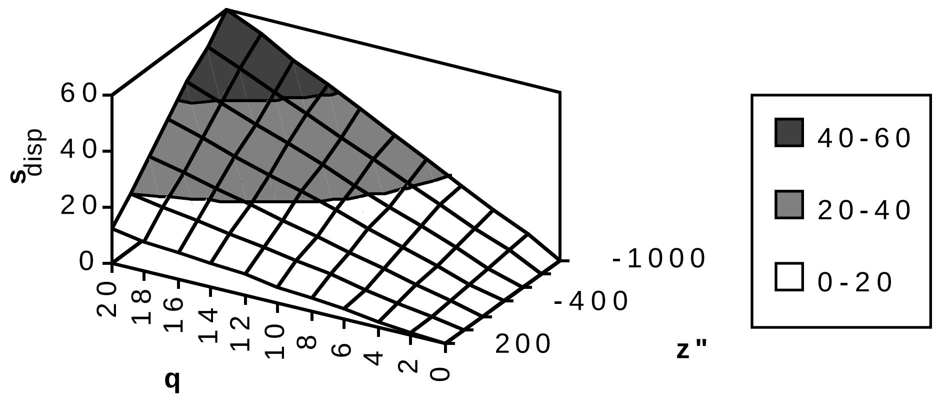


Figure 6.5.1.1. Lateral display error as a function of *q*

The next section is a discussion of the display error sources and their effect on the total registration error. In the sections that follow, I will use a subscripted version of the variable q as a shorthand for $\|\mathbf{v}_{Q''\_Q}\|$ in order to make the notation simpler (e.g., q_{dist} is the bound on $\|\mathbf{v}_{Q''\_Q}\|$ due to optical distortion error). Note that the subscripted qs are measured in actual image space, rather than in the modeled image plane as unsubscripted q is (this distinction is not important when the image misalignment is small).

6.5.2. Display Error Sources and Their Effects

6.5.2.1. Virtual-Image Misalignment

In Section 5.7.7 we saw that rigid-body transformations of the screen images relative to their modeled locations cause display error and derived the following exact and approximate bounds for total display error:

$$q = \|\mathbf{v}_{Q'\_Q^\dagger}\| = \|\mathbf{v}_{I'\_Q^\dagger} - \mathbf{v}_{I'\_Q'}\| = \|T_{Proj} \cdot (T_{I'\_I} \cdot \mathbf{v}_{I\_Q}) - \mathbf{v}_{I'\_Q'}\| = \\ \|T_{Proj} \cdot T_{I'\_I} \cdot (\mathbf{v}_{I\_Q''} + \mathbf{v}_{Q''\_Q}) - \mathbf{v}_{I'\_Q'}\| \qquad (5.7.7.1)$$

and

$$q \approx \|\mathbf{v}_{Q''\_Q}\| + \|\mathbf{v}_{I'\_I}\| + 2 \cdot \left| \sin \frac{\text{Ø}_{I'\_I}}{2} \right| \cdot \|\mathbf{v}_{I'\_Q'}\| \qquad (5.7.7.4)$$

The bound for $\|\mathbf{v}_{Q''\_Q}\|$ is treated in the subsections that follow; here we seek to bound the angular and translational error of the screen images relative to their modeled locations (i.e., $T_{I'\_I}$, which is the error in $T_{S\_I'}$). We therefore have

$$q_{im\_align} \approx \|\mathbf{v}_{I'\_I}\| + 2 \cdot \left| \sin \frac{\text{Ø}_{I'\_I}}{2} \right| \cdot \|\mathbf{v}_{I'\_Q'}\| \qquad (6.5.2.1.1)$$

for small misalignments of the images.

The critical observation regarding virtual-image misalignment is that the resulting error is fixed in image space and is therefore independent of viewing direction, etc. Moreover, certain misalignments are readily detectable (and therefore correctable). For example, if the screen image is shifted in x or y relative to its modeled location by 2 mm, *all* of the points in the scene will be shifted by that amount. Rotation about the z axis will displace points as a function of their distance from the rotation axis and can easily be detected with the use of a crosshair or grid. Errors in the remaining three degrees of freedom (z translation, x and y axis rotation) are less easily detected, *but that is precisely because they do not induce much registration error unless the error is severe.* That is, these errors for the most part move points along their projectors, which, because of the T_{proj} operator in Equation 5.7.7.1, has very little effect on *q*. If the errors are severe, they will be systematic and can therefore be distinguished from other error sources and corrected. In summary, errors in this transformation which induce noticeable registration error can be corrected, and those which do not induce noticeable registration error can be ignored. This assertion is supported by the calibration work described in Section 7.3.

Based on these observations and the experience with the prototype system, I estimate that this error can be reduced to 1 mm or less. That is,

$$q_{im\_align} \approx \|\mathbf{v}_{I'\_I}\| + \left| 2\sin\frac{\text{Ø}_{I'\_I}}{2} \right| \|\mathbf{v}_{I'\_Q'}\| \leq 1 \text{ mm}$$

which corresponds to lateral errors between 0.6 mm and 1.4 mm for points within 200 mm of the image plane.

Finally, it is worth noting that because this calibration procedure is user-independent, it can be done once with great care in order to achieve high accuracy.

6.5.2.2. Aliasing

The bound for aliasing in the common case of square pixels is

$$\|\mathbf{v}_{G'\_G}\| \leq \frac{\sqrt{2}}{2} h = \frac{\sqrt{2}}{2}\frac{g_x}{p_x} \qquad (5.7.8.1.3)$$

where g_x and g_y are the size of display screen in millimeters, and p_x and p_y give the resolution of the frame buffer (or the display screen in the case where the reconstruction process contains further sampling). Recall that the image-space error is just this term scaled by the linear magnification, *m:*

$$q_{alias} = m\cdot\|\mathbf{v}_{G'\_G}\| = \frac{\sqrt{2}}{2}\cdot m\cdot\frac{g_x}{p_x} \qquad (5.7.8.1.4)$$

For LCD screens in UNC HMD, if we assume resampling in the LCD as the worst case, and a value of 6.0 for *m*, we get

$$q_{alias} = \frac{\sqrt{2}}{2}\cdot m\cdot\frac{g_x}{p_x} = \frac{\sqrt{2}}{2}\cdot 6\cdot\frac{54.8 \text{ mm}}{120 \text{ pxls}} = 1.8 \text{ mm}$$

If we do not assume resampling, we have

$$q_{alias} = \frac{\sqrt{2}}{2}\cdot m\cdot\frac{g_x}{p_x} = \frac{\sqrt{2}}{2}\cdot(6.0)\cdot\frac{54.8 \text{ mm}}{640 \text{ pxls}} = 0.36 \text{ mm}$$

for NTSC resolution (640x480)[43]. Note that this answer applies to any display device which does not perform further sampling, such as CRTs.

While it is possible that the LCDs do introduce enough error to be concerned about, it also seems unlikely that any serious system would use such low-resolution devices, especially given that at least some of them seem to introduce further sampling errors and additional delays. It is therefore more useful in the author’s opinion to examine the second case above, in which we see that the net image-space error is less than a half a millimeter. This image-space error induces a lateral error of 0.2 mm at z" = 200 and 0.5 mm at z" = -200.

For the binocular case, [Hodges & Davis 93] point out that rounding to the nearest pixel also introduces discretization of depth levels, as shown in the figure below.

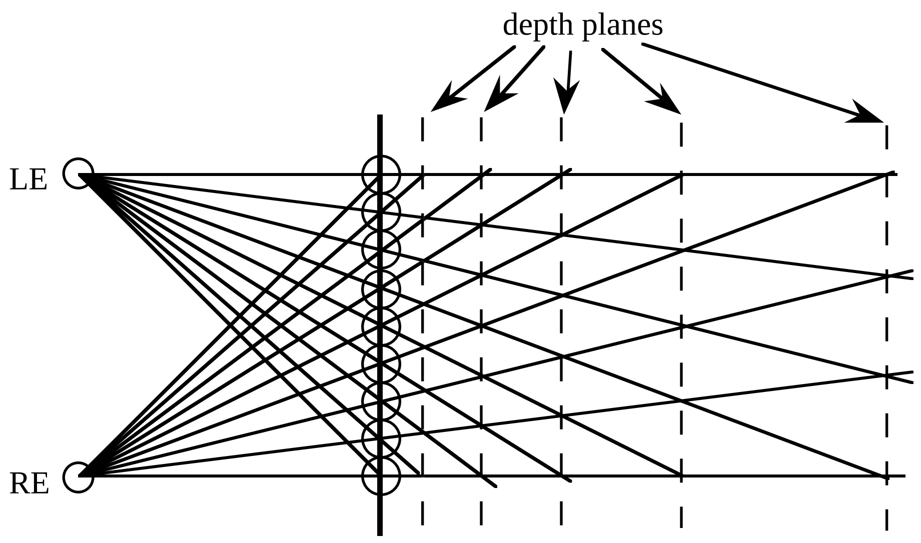


Figure 6.5.2.2.1. Depth discretization due to aliasing

From the figure, we can see how the discrete *x* locations for pixels leads to discrete depth planes, between which no (non-antialiased) point can be represented. The worst-case depth error for a point is about one half of the distance between the nearest two depth planes. Hodges and Davis give an expression relating the pixel size, IPD, and screen image distance to the location of the *z* planes shown in the figure. In terms of the notation used in this dissertation, the expression is

43 Note that because of the aspect ratio, using $\frac{g_y}{p_y}$ yields the same answer.

$$z_n = \frac{d}{\frac{2i \cdot p_x}{n \cdot w_x} - 1} \qquad (6.5.2.2.1)$$

where *n* is the number of the depth plane, p_x is the horizontal resolution, w_x is the width of the image plane, *i* is one-half the IPD, and *d* is the screen-image-to-eye distance. In the 500 mm beyond the projection plane, there are 11 depth planes for $p_x = 120$; for the 640 case, there are 60. Since the 640 case is more relevant, we can examine the depth-plane spacing in more detail. The following graph shows the depth-plane spacing for $p_x = 640$, $w_x = 328.6$, $d' = 500$ mm, IPD = 62 mm:

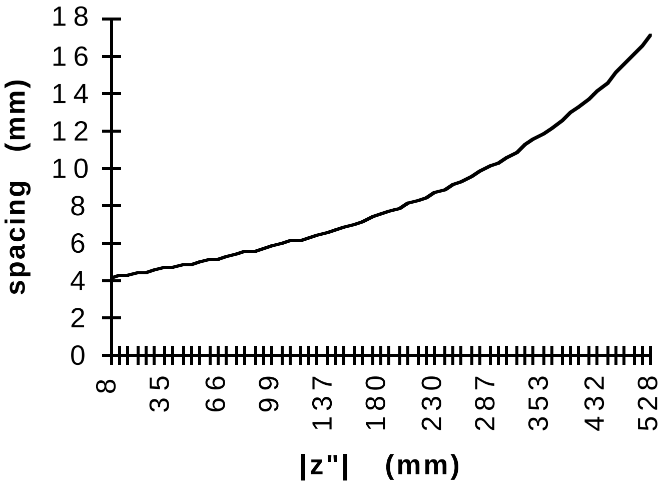


Figure 6.5.2.2.2. Depth plane spacing for $p_x = 640$

In the chart, the depth planes near the projection plane (z" = 0) are spaced at intervals of about 4 mm, but by z" = -500 mm, the planes are 16-17 mm apart. This means that the depth error varies from about 2 mm to 8 mm over this range.

However, because the display is head-mounted and the viewpoint is constantly changing, the image of any particular point will be represented by a number of pixels in any given sequence, which should tend to average out the errors. It seems unlikely that aliasing will be cause for concern (with respect to registration error) for head-mounted displays for the foreseeable future.

6.5.2.3. Display-Technology-Specific Errors

Recall that for display nonlinearity, we have

$$\|\mathbf{v}_{G'\_G}\| \leq \sqrt{\eta\, g_x^2 + \eta\, g_y^2} = \eta \sqrt{g_x^2 + g_y^2} \qquad (5.7.8.2.1)$$

where we use η to represent the absolute nonlinearity of the CRT or other device. The image space error q in the absence of other errors is

$$q_{nonlin} = m \cdot \|\mathbf{v}_{G'\_G}\| = m \cdot \eta \sqrt{g_x^2 + g_y^2} \qquad (5.7.8.2.2)$$

For a 1" CRT such as is used in the Tektronix NuColor displays, the dimensions are 0.768"x0.6" (19.5mm x 15.2 mm) [Chi 93] and the nonlinearity may be as high as 3% [Allen 94], which gives

$$q_{nonlin} = 6.0 \cdot 0.03\sqrt{19.5^2 + 15.2^2} = 6.0 \cdot 0.03 \cdot 24.75 \text{ mm} = 4.5 \text{ mm}$$

(assuming a linear magnification of 6.0 as before). Nonlinearities of 0.1%, 1% and 2% yield 0.15 mm, 1.49 mm, and 3.0 mm of image space error, respectively.

The 3% distortion figure applies to the first generation of Tektronix displays and probably overestimates the error that we should expect to see in these displays over the next few years. Improved drive electronics will likely move the nonlinearity into the 1% range, which corresponds to the 1.49 mm image-space error quoted above. Moreover, if the nonlinearity is constant in time, it should be possible to use an image prewarp similar to that done for optical distortion in order to compensate for the remaining nonlinearity.[44] The tradeoffs in such an approach are discussed more in the next section.

6.5.2.4. Optical Distortion

Before beginning the analysis of optical distortion error, we should first consider the argument that says that this error source does not matter because it is correctable. To my knowledge, there are three ways to correct (or avoid) optical distortion:

1. Design the optical system in the STHMD so that the distortion is negligible.
2. Prewarp the image electronically in the display device by introducing the opposite distortion in the display hardware (this is possible with CRTs).
3. Prewarp the image in software in the rendering process.

The first two options have the advantage that they do not add delay to the system and are therefore usually preferable to the third option. The problems with these two approaches is 1) they must be implemented in the HMD hardware design, which is not an option for

[44] It is worth noting, however, that the nonlinearity will itself be distorted by the optics, so any prewarp method will need to take this into account.

users who buy off-the-shelf STHMDs, and 2) designing a system with no distortion may be infeasible due to other constraints, such as cost, weight, and minimization of other optical aberrations[45] [Kocian 88]. In addition, electronic correction is not possible for display devices (such as LCDs) that do not use a deflected beam to draw the picture.

For systems without optical or electronic distortion correction, the distortion can be corrected by prewarping the images in the rendering process. The problem with this approach is that the computation delay for warping the images may induce more rigid-body transformation error than the distortion error it corrects. Moreover, the display-induced error misregisters the entire image, not just the periphery, which means that the prewarping could easily make the registration worse in the center of the image. Thus, whether a given application should prewarp in the rendering process depends on *a)* how severe the distortion is, *b)* how fast the predistortion can be done on that system, and *c)* whether the more accurate image is worth the possibly slower head speeds imposed by the prewarping. For Pixel-Planes 5, [Lastra 94] reports that he was able to achieve 20 frames per second with predistortion, but only by adding it as a stage in the rendering pipeline. This added a frame of delay, or about 50 ms, which corresponds to 50 mm of error for the maximum head velocities observed in this application. This is clearly a greater registration error than that introduced by the distortion itself and leads to the conclusion that predistortion in the rendering process is not a quick and simple fix for this problem, at least for some systems. That said, we can begin the analysis of optical distortion in order to see how much error it causes if left uncorrected.

For distortion in the presence of screen-space errors, we have the following expressions from Section 5.7.9.2:

$$\mathbf{v}_{Q''\_Q} = \mathbf{v}_{Q} - \mathbf{v}_{Q''} = \begin{bmatrix} (m-m')x_{G'} + m\,x_{G'\_G} + k\,m^3 r_G^2 (x_{G'} + x_{G'\_G}) \\ (m-m')y_{G'} + m\,y_{G'\_G} + k\,m^3 r_G^2 (y_{G'} + y_{G'\_G}) \end{bmatrix} \qquad (5.7.9.2.9)$$

and

$$q_{dist} = k\,m^3\,r_G^3 \qquad (5.7.9.2.10)$$

[45] Even the CAE FOHMD [CAE 1986], which is part of a million-dollar flight simulation system, has some distortion (~1.5% according to the sales literature).

The first expression gives the error due to mismodeling the linear magnification (m' ≠ m), linear magnification of screen-space errors (as we have seen in the previous sections where we magnified the errors by *m*), and distortion of both the screen-space errors and the modeled points. The second expression makes it clear that the values for *m* and *k* (the coefficient of optical distortion in mm^{-2}) will greatly affect the within-image error. I will begin by examining the case of the UNC STHMD and then try to generalize to other systems.

The calculated value for *m* for the UNC STHMD is 6.0; the value for *k* requires some explanation. The coefficient of optical distortion is itself a function of the *eye relief*, which is defined as the distance from the last optical element to the eye. Because the expression for the variation of *k* with eye relief is quite complex, I will give a plot of the variation of the distortion for the UNC STHMD as one example, with the understanding that systems with significantly different optical properties may exhibit different behavior.

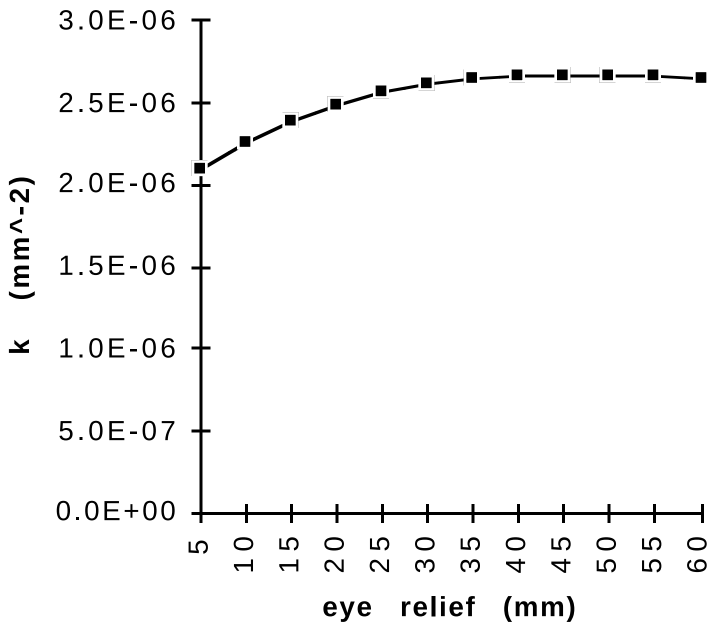


Figure 6.6.5.1. Coefficient of distortion vs. eye relief for the UNC 30˚ STHMD

This plot gives *k* as a function of the eye’s distance from the beam splitter. The modeled eye relief for the system is 40 mm; the graph shows that *k* only varies about 6% within 20 mm of the modeled eye relief. Thus, in this case, it is reasonable to treat *k* as constant if we assume that the actual eye relief will be within 20 mm of the modeled eye relief (and perhaps beyond). I will therefore use a worst-case value of *k* of 2.66 x 10^{-6} mm^{-2}. With this value and the value for *m*, we can move on to an analysis of distortion-induced registration error for the UNC STHMD.

From Equation 5.7.9.2.10 above, we can see that distortion error is at its maximum when r_G is at its maximum, which is at the corners of the screen.[46] The dimensions of the LCDs used in the UNC STHMD are 54.7x41 mm, so the maximum radius is the distance from the center of the LCD to one of the corners, i.e.,

$$r_{G\_max} = \frac{\sqrt{54.7^2 + 41^2}}{2} = 34.2 \text{ mm}$$

If we normalize the radial distance in terms of this maximum and plot the pure distortion error for, say, the upper right quadrant of the screen image, we get the following plot:

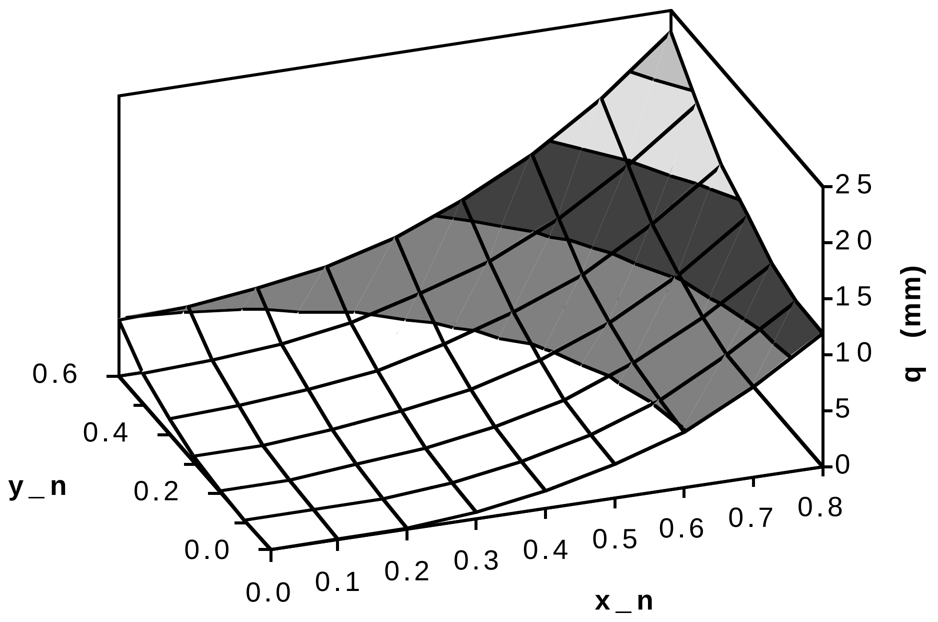


Figure 6.6.5.2. Pure distortion error (in image space) vs. normalized screen-space coordinates. Color changes occur at 5 mm intervals.

46 Assuming the screen is centered relative to the optics (which is the case with this system), the corners will all be equidistant from the optical center; for decentered systems, the furthest corner in screen space will have the greatest radius and therefore the greatest distortion.

Here, *x_n* is the normalized *x* coordinate in screen space (similarly for *y_n*) and I have used the previously calculated values for *k* and *m*. Because of the 4:3 screen aspect ratio, the normalized screen-space coordinates have maxima in *x* and *y* at 0.8 and 0.6, respectively, and the corner is at unit screen-space radius. It is clear from the plot that the error in image space becomes quite significant in the corner(s) of the image, where the distortion error is 23 mm (corresponding to 11.2% distortion). At the center of right edge, the distortion error gets up to 11.8 mm (7.2%), and at the center of the top edge it reaches 5 mm (4%). For points 200 mm beyond the screen image, the lateral error is 32 mm at the corner, 17 mm at the left/right edge, and 7 mm at the top/bottom.

In general, the distortion error scales linearly with *k* and as the cube of $m \cdot r_G$. Thus, for a given value of *k*, using a larger display device to increase the system field of view will also increase the distortion error significantly at the edges of the larger virtual image. For example, doubling the screen size will increase the error by a factor of eight. In a similar fashion, increasing *m* (to obtain a wider field of view, for example) will also generally increase the distortion due to the $(m \cdot r)^3$ term.

If we have the distortion percentage at a given location in the image (say, the right edge), we can translate this directly into image-space error via

$$q_{dist} = \rho \cdot \|\mathbf{v}_{I\_Q''}\| = \rho \cdot \frac{w_x}{2} \qquad (6.5.2.4.1)$$

The following graph shows the maximum distortion error at the right edge for a system with a 60˚ field of view (corresponding to an image plane 577 mm wide at a distance of 500 mm):

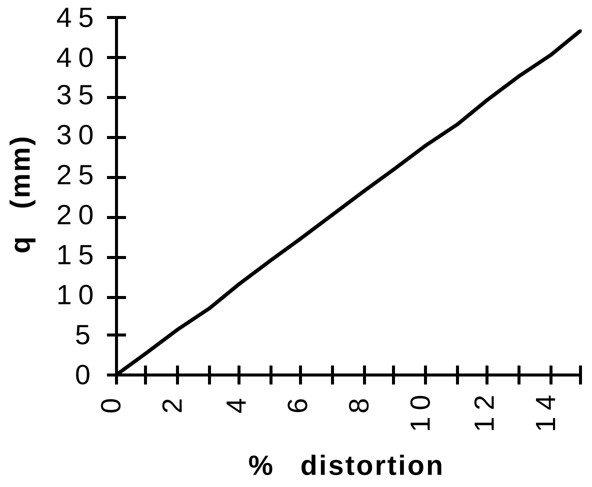


Figure 6.6.5.3. Error at right edge of 60˚ field of view vs. percent distortion

As shown in the plot, when the distortion is expressed as a percentage, the worst-case image-space error just varies linearly with it. The plot shows that for this field of view, each percentage point of distortion adds about 2.9 mm of error at the right edge (for the UNC STHMD, each point adds 1.6 mm).

Finally, because distortion is systematic, we can look at the binocular case to see what sort of warpings in depth it is likely to cause. The following figure is a top view that shows the warping of a rectangle caused by distortion.

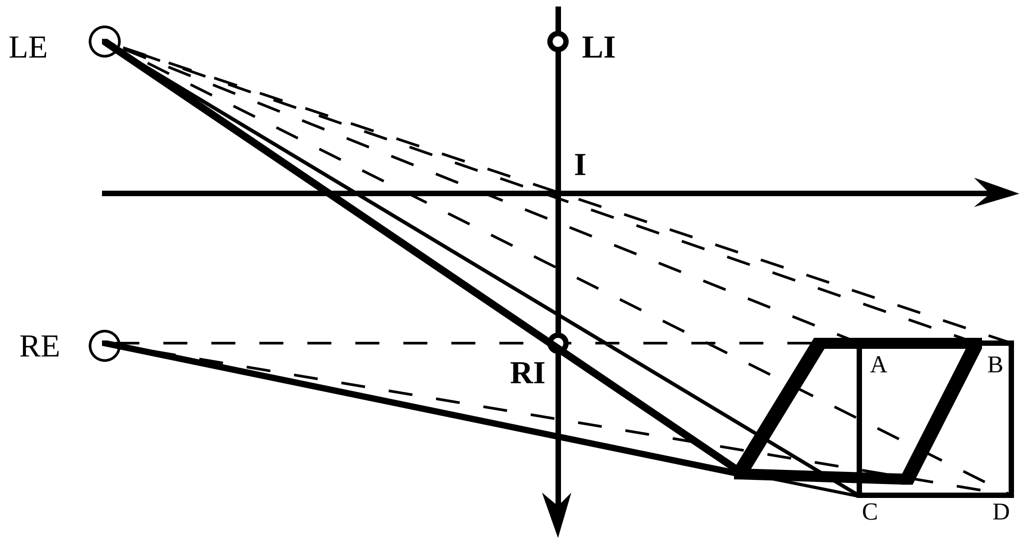


Figure 6.6.5.4. Warping in depth due to optical distortion

The points LI and RI are the centers of the two Image coordinate systems (CSs) and therefore the centers for distortion in each image. Thus, the projectors from the points A and B that pass through RI are not affected by the distortion, whereas all the other displayed points are moved by distortion in both images. The distorted projectors are shown only for the point C; note how much more the projector from LE is warped than that from RE, since the image-plane point for the left eye is much further from its center. In general, distortion tends to cause peripheral objects to “wrap around” the user; that is, it tends to move points further into the periphery but shifted inward toward the user.

Summary/Observations:

- Distortion is a small error source in the center of the images but increases rapidly in the periphery and becomes quite large (for the UNC system it is on the order of 20 mm). For objects with finite extent, misregistration near the edges may be large.
- Because distortion is an image-space error, the amount of warping will be a function of where the object is drawn within the field of view. Thus, as the user’s head moves, the object will seem to warp dynamically, which may be distracting.
- Distortion must be accounted for in calibration procedures in order to minimize the artifacts from it. Paraxial or near-paraxial values should be used for all parameters in order to minimize the error near the centers of the images.
- Because the eyes converge to look at an object, one or both eyes will typically be looking at image points that are not at the exact center of the image, which means that some amount of distortion error will be present even in the best of cases.
- While there is interaction between distortion and the screen-space errors already discussed, for systems with 10-20% distortion, the additional error will be a relatively small effect and need not be analyzed in detail here. (High-precision calibration procedures should take this effect into account, however.)

6.5.2.5. Lateral Color

Lateral color, like distortion, is greatly dependent on the optical design used for the STHMD. As such, it is hard to draw specific conclusions about it without using a specific optical design. I will therefore begin with the general expressions for lateral color error, plug in values from the UNC STHMD to get error bounds for this system, and then discuss the implications.

From Section 5.7.9.3, the general expression for display error in the presence of lateral color error is

$$\mathbf{v}_{Q''\_Q} = \begin{bmatrix} (m-m')x_{G'} + m\,x_{G'\_G} + k\,m^3 r_G{}^2(x_{G'} + x_{G'\_G}) + \dfrac{\alpha + ßkr_G{}^2}{3}(x_{G'} + x_{G'\_G}) \\ (m-m')y_{G'} + m\,y_{G'\_G} + k\,m^3 r_G{}^2(y_{G'} + y_{G'\_G}) + \dfrac{\alpha + ßkr_G{}^2}{3}(y_{G'} + y_{G'\_G}) \end{bmatrix} \qquad (5.7.9.3.16)$$

In the case of pure lateral color error, the first three columns go to zero and k = 0 eliminates the term involving β, so we have

$$\mathbf{v}_{Q''\_Q} = \left(\frac{\alpha}{3}\right)\cdot\mathbf{v}_{G'} \qquad (6.5.2.5.1)$$

But this term is linear, so we can correct for it by adjusting the modeled image scale and reduce the error to zero[47].

In the case when the distortion is not zero, we can examine the simple case in which there are no errors in *m* (i.e., m' = m) and no display-device errors (i.e., G' = G and $\mathbf{v}_{G'\_G} = \mathbf{0}$), so that

$$\mathbf{v}_{Q''\_Q} = \begin{bmatrix} k\,m^3 r_G{}^2 x_G + \dfrac{\alpha + ßkr_G{}^2}{3}x_G \\ k\,m^3 r_G{}^2 y_G + \dfrac{\alpha + ßkr_G{}^2}{3}y_G \end{bmatrix} \qquad (6.5.2.5.2)$$

and for which we have already defined (in Section 5.7.9.3)

$$q_{LC} = r_{avg} - r_g = \frac{\partial r_r + \partial r_r}{3} = \frac{\alpha r_G + ßkr_G{}^3}{3} \qquad (5.7.9.3.12)$$

Note that r_g in this case is the radius of a point in the *distorted* green image, so we are only examining the additional error caused by first- and third-order lateral color.

We can now plug in values for the UNC STHMD to get an idea of the magnitude of this error. For this system, α = 0.0067, β = 5.83, and using the maximum value of r_G (34.2mm), we get

$$q_{LC} = 0.28 \text{ mm}$$

Thus, the net error due to lateral color for the current UNC STHMD is comparatively small and can probably be ignored. It is worth noting, however, that the individual variations in

47 Recall the assumption from the previous chapter that we are concerned only with the center of energy for the imaged point: the red, green, and blue images of each point could still deviate from the modeled location, but the centroid would be correct.

the red and blue images are *not* small; the variations at the image corners are given below for the linear (paraxial) case and the cubic case (in which distortion is also present).

Quantity	**Linear**	**Linear + Cubic**
∂r_r	1.20 mm	3.46 mm
∂r_b	-0.98 mm	-2.61 mm
$\frac{\partial r_r + \partial r_b}{3}$	0.08 mm	0.28 mm

Table 6.5.2.5.1. Deviations due to lateral color for red and blue images

Thus, the variations of the red and blue images from the green (reference) image are significant in the extreme periphery of the image, but average effect is quite small.

If separation of the three images is a problem and cannot be corrected optically, there are computational means for correcting it, but, as with distortion, it may introduce more delay error than the error it corrects. A straightforward (but potentially expensive) solution would be to use a predistortion algorithm that corrected separately for the linear and cubic magnifications of the red, green, and blue images.

Because lateral color is simply a combination of linear and cubic error terms that have been discussed already, there is no need to analyze it further.

6.5.3. Display Error Conclusions

We have now examined the behavior of the net registration error due to display error and can draw the following conclusions:

- The most troublesome display error source is optical distortion, which can easily cause image-space errors on the order of 20 mm. Correcting distortion is possible, but current methods are so slow that the resulting delay-induced error is worse than the distortion-induced error it corrects.
- The binocular effect of distortion is that objects tend to “wrap around” the user’s field of view, with the more distant parts of the object appearing closer than they actually are.

- Display errors can be distinguished from other errors since they are fixed in image space. The lateral error induced by display error increases linearly with eye-to-point distance, which means that calibrating using distant points allows for fine corrections to be made. Also, display lateral error behaves differently from viewing lateral error, which can also be used in calibration procedures.
- Screen-space errors must be evaluated with the system's magnification in mind, since the optics will exaggerate even tiny errors in screen space.
- Aliasing error is a small error source for a given static viewpoint, and over time its effects should average out and therefore be negligible.
- Image alignment, aliasing, display-device nonlinearity, and lateral color should not be significant error sources for carefully calibrated systems.

6.6. Analysis Summary

The upshot of the analysis is that the main culprits in the problem of registration error are the "usual suspects": tracker delay, tracker error, and distortion. To a lesser extent, eyepoint movement and acquisition/alignment errors play a part in misregistration of real and virtual objects. Finally, the even-smaller error sources, namely, virtual-image alignment, lateral color, display-device nonlinearity, and aliasing, contribute to make sub-millimeter registration a distant goal.

An approximate ranking (in order of severity) of error sources for this application follows.

1. System delay error: 20-60+ mm
2. Distortion: 0-20 mm, depending on object location in image space
3. World-Tracker error: 4-10 mm (although this transform may be omitted)
4. Tracker Error (static, dynamic, jitter): 1-7+ mm
5. Scanning/alignment errors: 1-3 mm
6. Viewing error: 0-2+ mm depending on object depth relative to image plane
7. Display nonlinearity: 1-2 mm, depending on magnification of optics and display device itself
8. Others (< 1mm): image misalignment, lateral color, aliasing.

This ranking is somewhat arbitrary, since the actual registration error depends on many factors, including

- System hardware: tracker, optical design and display device for STHMD, medical imaging hardware
- System calibration
- System parameters (working distance, virtual image distance, medical imaging parameters, etc.)
- Viewing parameters at any instant (object position within image plane, head-tracker separation, eye-to-point distance)

but gives some idea of which error sources are the most troublesome.

One of the most useful features of the error model is that it gives the person calibrating system information on how to distinguish errors of each type, since, as [Deering 92] notes, "the visual effect of many of the errors is frustratingly similar." Here, then, is a summary of the distinguishing characteristics of the error sources.

Distinguishing Characteristics

1. Acquisition/Alignment Error

1.1. Acquisition errors: Warpings of the object itself. Unlikely to be confused with other error sources since they tend to be fixed in World space (W) and localized.

1.2. Alignment errors: Errors in aligning the virtual object in W. Error is fixed in W, but translation error may be confused with translation error in T_{W_T}, which is also fixed in W. These two can be distinguished by adding a reference object in W; if it is also misaligned, then the error is in T_{W_T}.

2. Head-Tracking Error: Error in locating the head tracker in the World CS and error in locating the tracker sensor with respect to the Tracker CS. Induces rigid-body transformation errors in the virtual objects, but no warping.

2.1. World-Tracker error: Error in position and orientation of tracker in W. Error is fixed in W; orientation error can be identified by its scaling behavior, given in Equation 5.4.3.2.

2.2. Static Tracker-Sensor error: Error in immobile head's position and orientation as reported by the tracker. Translation error is not usually fixed in W and is therefore distinguishable from World-Tracker translation error. Angular error has different scaling property from that of T_{W_T} (Equation 5.4.3.2 again).

2.3. Delay-induced registration error: Error for head in motion. For a system without eye tracking or other dynamic measurements, this is the only error that varies with head motion (unless the STHMD slips on the user's head during motion).

3. Viewing error: Error in the modeled eyepoint locations. Can cause warpings (*e.g.,* shears, compressions, and elongations) as described in Section 6.4. Error goes to zero for points in image plane, and lateral error increases linearly in both directions as point moves away from image plane. This distinguishes it from display lateral error, which increases linearly from the eyepoint out. Can be detected by noting that as a point is moved through the image plane, its error decreases to zero and then increases again in the opposite direction (see Section 7.3.4 for details).

4. Display error: Error in creating the image of the virtual scene. May cause warpings of virtual objects. In general, display lateral error for a point increases linear from eyepoint, which means that very distant calibration points should be used to detect and correct errors of this type.

4.1. Image misalignment: Error in modeling position and orientation of virtual images. *x* and *y* errors in T_{S_I} move all points in image uniformly, and *z* axis rotations roll the image. *z* translations may appear as uniform scaling error. Small *x* axis rotations cause scaling errors in *y* direction and vice versa.

4.2. Aliasing: Pixel-rounding error. Causes points to snap to the nearest pixel and may cause discrete rather than continuous movements of scene as head moves. Error is generally localized and does note affect entire objects uniformly.

4.3. Display-device nonlinearity: Warping of image displayed on the screen itself. Nature of error depends on device used; for CRTs, there may be warpings which resemble optical distortion. The nature of these distortions can be determined by making measurements of device with optics removed.

4.4. Optical distortion: Warping of virtual image due to optical aberration. Causes objects to "wrap around" in the periphery. Easy to distinguish from simple scaling error because of its nonlinear, 3rd-order behavior.

4.5. Lateral color: Separation of red, green, and blue images due to variation of magnification with wavelength. Has linear and cubic terms which just add to linear and cubic scaling errors. Distinguishable because of color separation.

The next chapter details the experiments for generating input error bounds and for verifying the error model.

7. Experiments

7.1. Overview

This chapter presents the experiments run both to find first-order values for unknown system parameters and experiments run to verify the error model. These experiments were run to get ballpark values for input errors and to serve as a sanity check for the error model and therefore were designed to be simple rather than precise.

7.2. Input-error experiments

7.2.1. Tracker Jitter

As discussed in Section 5.4, tracker jitter is noise in the tracker's outputs and limits the degree to which the tracker can be calibrated. Even though I use a Faro arm for tracking in Section 7.3, it is not particularly well-suited to head tracking for real applications; I therefore present data for the Polhemus Fastrak here since it is more likely to be used as the head tracker in a real system. (The Faro's jitter numbers were rather consistent; translation sigma values were less than 0.05 mm and orientation sigmas were around 0.005˚.)

The jitter in the Fastrak was measured by taking a number of readings for an immobile sensor and computing the standard deviation in position and orientation. During these measurements, a single sensor was attached to a Virtual Research Flight Helmet on a Plexiglas mount that raised it about 80 mm from the top of the HMD. The HMD was laid on a non-metallic surface at various positions and orientations throughout the working volume and 100 readings were taken at each location. All measurements were taken in our lab under normal operating conditions, which means that there were both metallic objects and CRTs nearby.

For the position measurements, the standard deviation of the *x*, *y*, and *z* values was calculated and then combined into a single root-mean-squared (RMS) value for each position. This gives an estimate of the RMS standard deviation for position. For orientation, the measured quaternions were averaged (by the computing the mean rotation-axis vector and the mean rotation angle), and then the angle between this mean orientation and the sample orientations was computed for each reading. These deviation angles were then squared, summed, and divided by the number of readings to give an angular variance term. The square root of the angular variance is the angular standard deviation used in the plots that follow.

The Fastrak has user-changeable filter settings, which means that one can reduce the jitter at the expense of increased system latency[48]. The filter parameters used for this study[49] are the same as those used by Mark Mine [Mine 95] in the lag study in which he measured the Fastrak latency to be about 11 ms, which is relatively small (1/5 of the current host/image generation delay).

The next two figures show the standard deviation plots for translation and orientation for the Polhemus Fastrak in the lab at UNC.

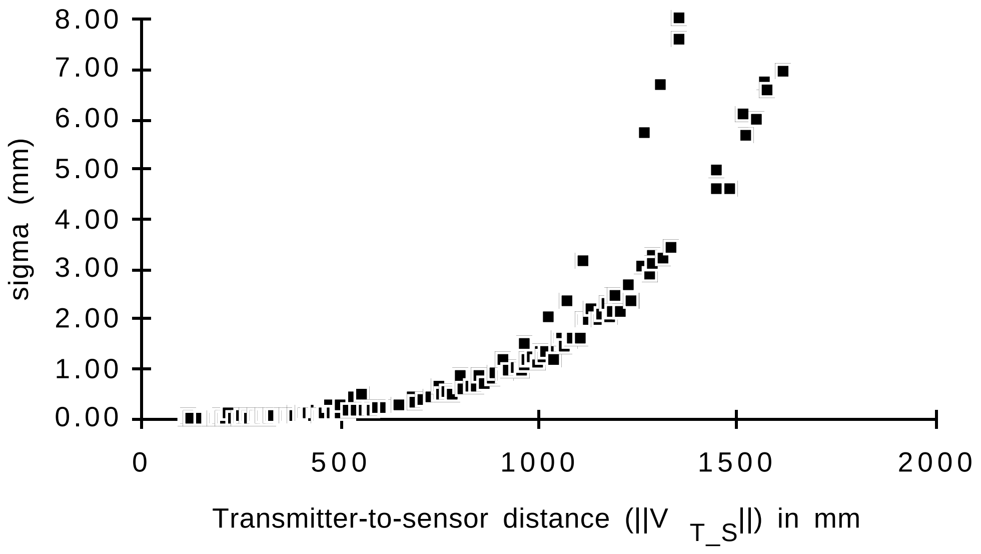


Figure 7.2.1.1. Translation sigma values for Fastrak

48 For example, Polhemus quotes the latency for its Isotrak II as 23 ms with filtering off, and 40 ms with filtering on.

49 Specifically, F = 0.2, F_{low} = 0.2, F_{high} = 0.8, and Factor = 0.8 for both position and orientation. The explanations for these values are in [Polhemus 92].

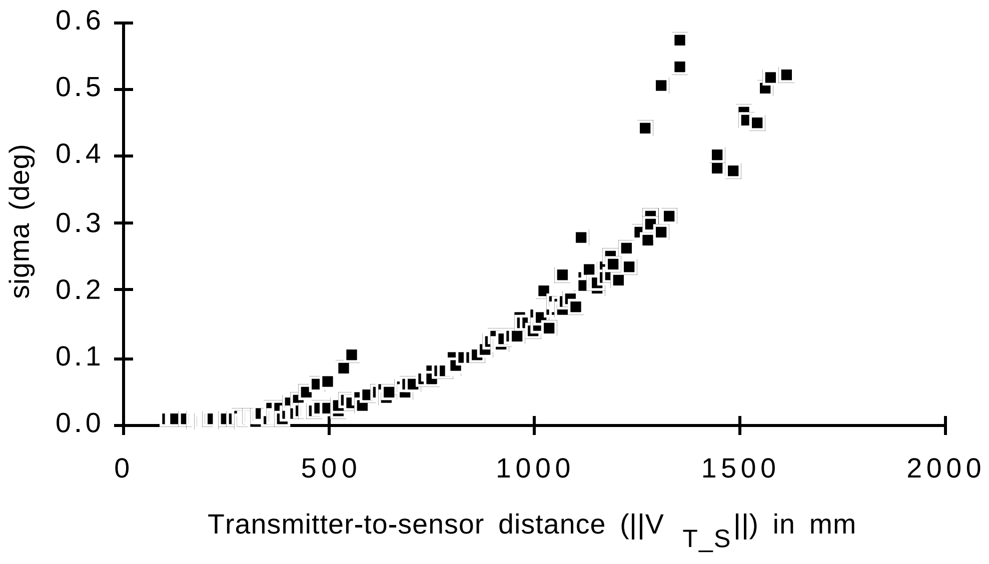


Figure 7.2.1.2. Orientation sigma values for Fastrak

Recall that $||\mathbf{v}_{T\_S}||$ is the distance from the tracker's transmitter to the sensor. The implications of these measurements are discussed in Section 6.3.

7.2.2. Head Motion Measurement

I conducted a simple study in order to get an idea of typical head velocities and viewing distances for a surgery planning application. For this study, a surgeon (Dr. Vincent Carrasco from the Department of Surgery, Division of Otolaryngology of UNC Hospitals) conducted a simulated planning session[50] on three subjects[51] while his head's position and orientation was tracked. A Polhemus Fastrak sensor was mounted to a headband that Dr. Carrasco wore; the position and orientation of this sensor was measured approximately 60 times per second. The sessions lasted about 50 seconds on average, which corresponds to about 3000 measurements.

In a normal planning session, the surgeon sits on a wheeled stool which allows him to roll easily to different viewpoints in order to evaluate the patient. In addition to moving himself

---

[50] The session was conducted in our graphics laboratory because of the special equipment required for head-tracking; a more representative sample of head motion data could be collected by conducting the study in a clinical setting.

[51] The subjects (two male, one female) were volunteers from the Computer Science department.

about the patient, the surgeon may move the patient's head in order to more easily see inside the nose, ears, or mouth.

The study of each patient was divided into three sessions. The goal of the first session was to simulate a normal planning session as described above. For the second session, the patient's head was kept fixed, since it will probably be necessary to immobilize the patient's head in order to register the virtual anatomy with it. In the third session, the head was also fixed and Dr. Carrasco was asked to move as slowly as he could while still conducting a normal examination. The objective was to see whether there were high-velocity movements that might be reduced voluntarily (without impeding the examination). If so, the velocities in these runs might be taken as a lower bound on what one could expect for this application.

I had initially set up the system so that the data could be tagged during periods of careful scrutiny in order to distinguish these samples from those collected during viewpoint-changing movements. However, it became clear during the first practice runs that Dr. Carrasco was studying the patient even as he moved. It appears that the motion-based cues (such as kinetic depth effect and head-motion parallax) are an integral part of the exam, and therefore all data was treated equally.

Finally, the data was also used to compute a range of E-P (eye-to-real/virtual-point) distances to derive the range of expected values for the parameter *d*. The distance used for this was from the bridge of the patient's nose to the midpoint between the surgeon's eyes (these two points were digitized before the sessions started). The range of distances used in all sessions is plotted in Figure 7.2.2.7.

The data did not vary significantly between patients and only slightly between sessions for a given patient. Figures 7.2.2.1-6 show the linear and angular velocities for the three types of session with all three patients given in each chart. For the linear and angular velocities, the histograms were generated by grouping the samples into intervals spaced 5 mm/s apart for linear and 1 deg/s for angular and counting the number of samples in the interval. In all of the figures, the solid line is for the Subject 1, the medium-dashed line is for Subject 2, and the finely dashed line is for Subject 3. Each session with each subject consisted of about 3,000 samples.

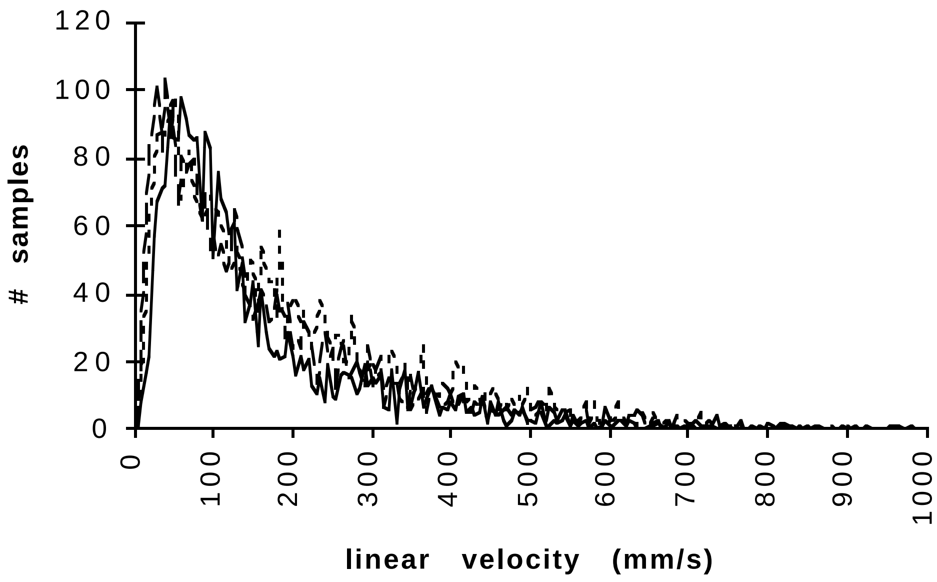


Figure 7.2.2.1. Linear head velocity histogram for Case 1: Normal exam

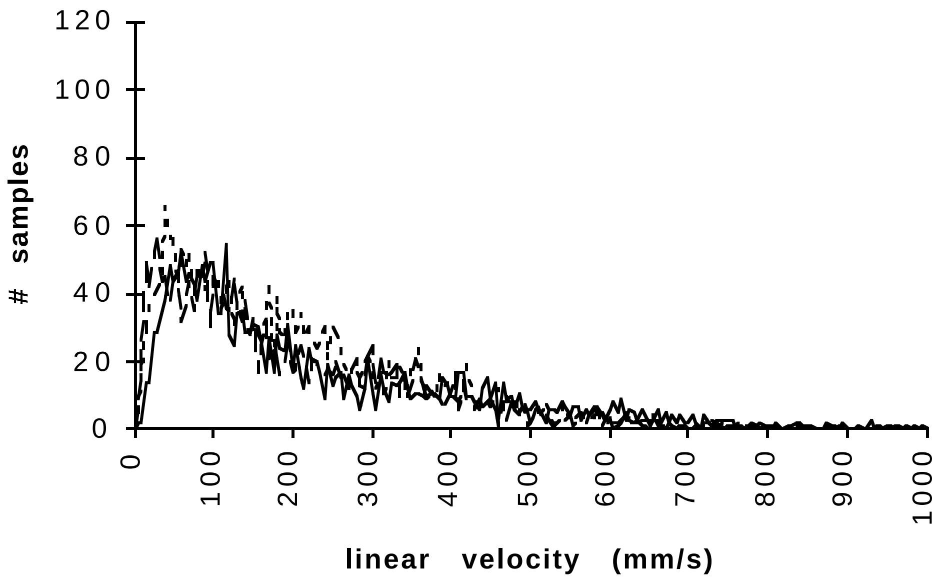


Figure 7.2.2.2. Linear head velocity histogram for Case 2: Fixed head

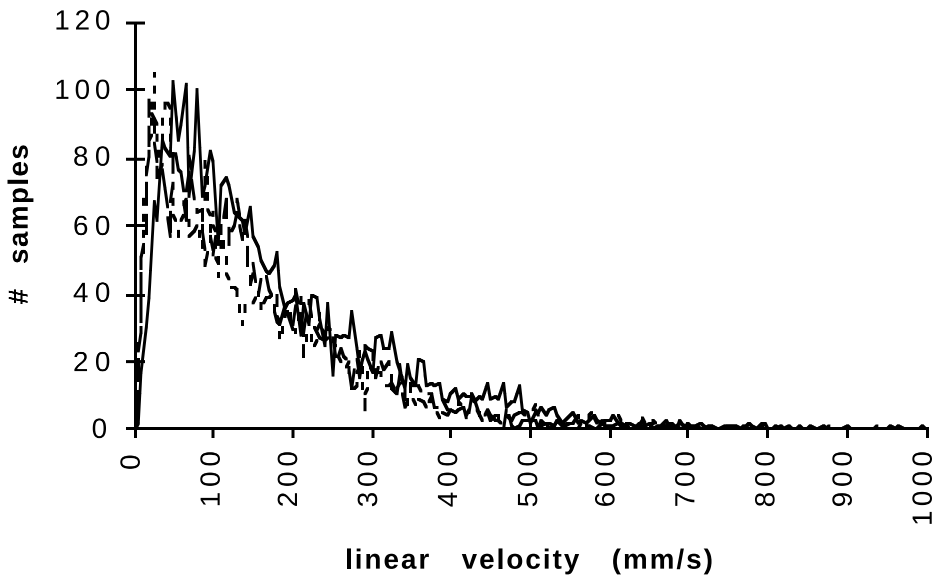


Figure 7.2.2.3. Linear head velocity histogram for Case 3: Slow movement

The next three figures show the angular velocities for the three cases.

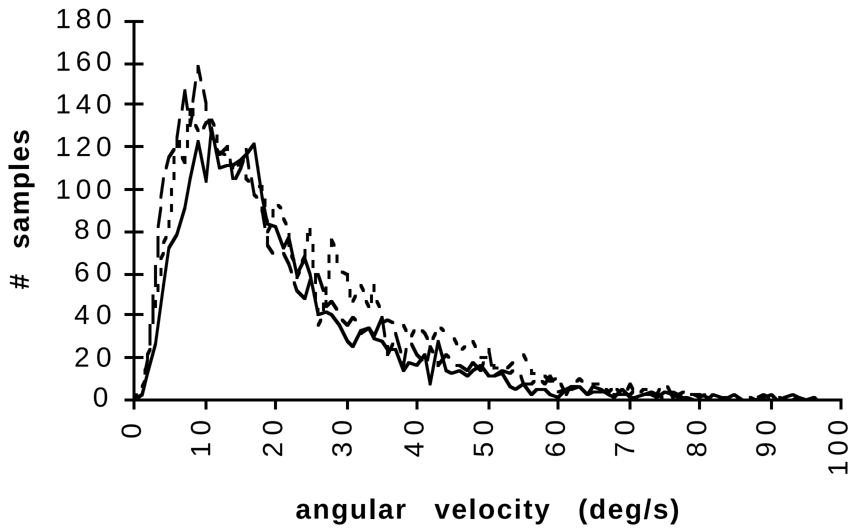


Figure 7.2.2.4. Angular head velocity histogram for Case 1: Normal exam

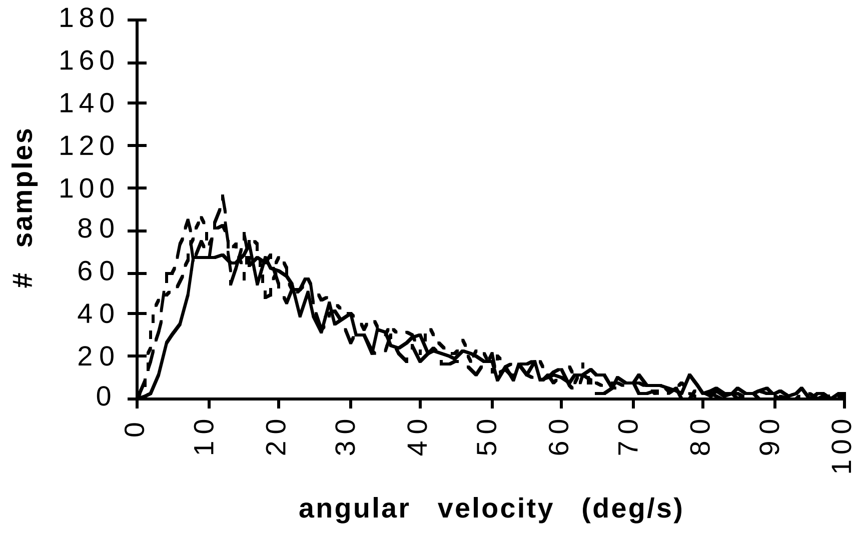


Figure 7.2.2.5. Angular head velocity histogram for Case 2: Fixed head

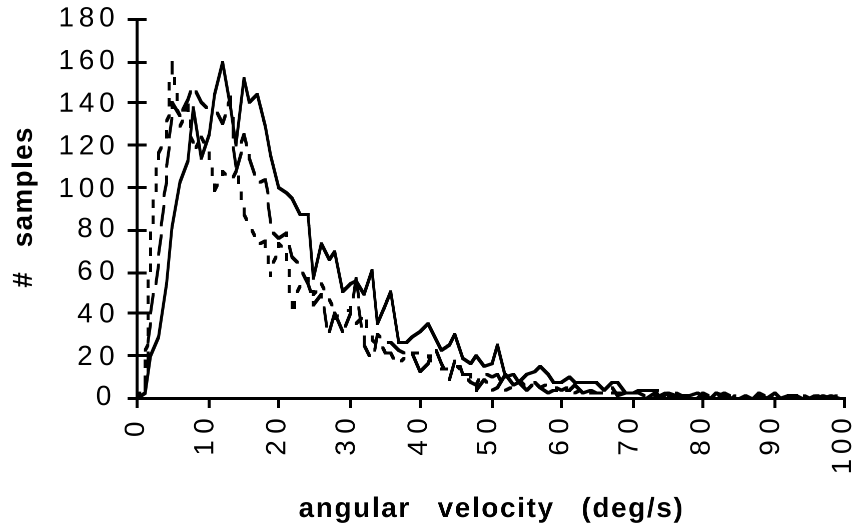


Figure 7.2.2.6. Angular head velocity histogram for Case 3: Slow movement

The average linear velocities for the three cases were 170 mm/s, 211 mm/s, and 164 mm/s. The average angular velocities for the three cases were 21 deg/s, 26 deg/s, and 20 deg/s. The higher velocities for the second case can be explained at least in part by the additional head movement required because the patient's head was fixed. Note that in the third case, voluntarily slowing the head motion only reduced the velocities slightly as compared to the normal case, but did result in about a 20% reduction vs. the second case.

The viewing distance data is plotted in Figure 7.2.2.7. Again, the distance measured is that from the bridge of the surgeon's nose to the bridge of the subject's nose. The histogram below represents approximately 27,000 samples.

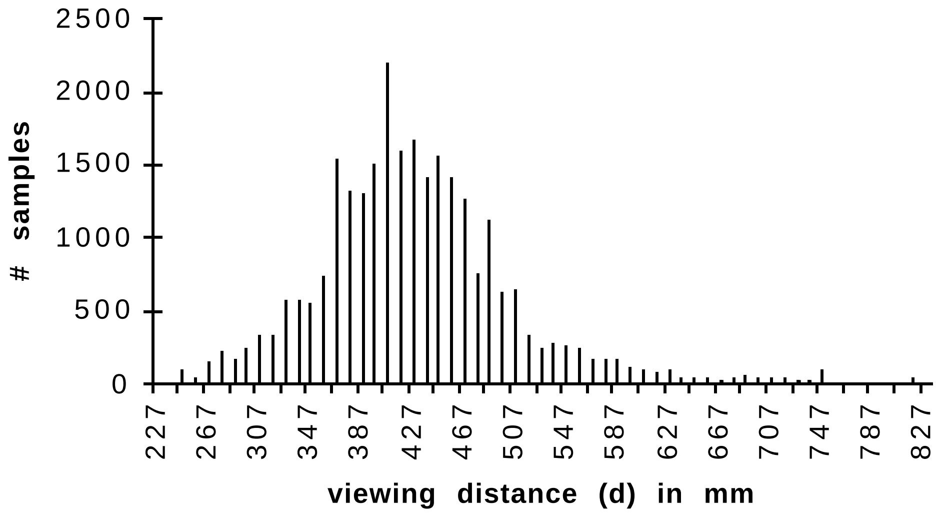


Figure 7.2.2.7. Distribution of viewing distances (d) in mm

The implications of this data are discussed in Chapter 6.

7.3. System Calibration

In order to perform the sanity-check experiments with a real system, I first had to calibrate the system to reduce the errors as much as possible. The system used for these experiments was the UNC 30˚ STHMD connected to Pixel-Planes 5 and a Faro Industrial Metrecom mechanical tracker. The Faro arm is a very accurate tracker/digitizer[52] and was used so that the system could be calibrated accurately; however, due to its limited range and unwieldiness, it is not an ideal solution for a real surgical planning system. The figure below shows the experimental setup.

52 According to [Faro 93], the 2σ value for single-point repeatability is .3 mm, and the 2σ value for linear displacement is 0.5 mm. My experience is that it meets these specifications in real use.

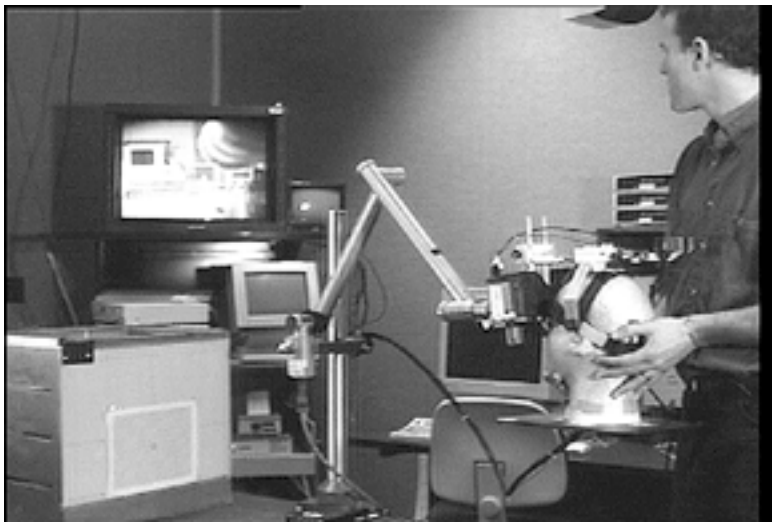


Figure 7.3.1. System overview

A first-order calibration of a STHMD requires that the following parameters be determined:

1. The location of the virtual objects in World space (if W is not used, then in Tracker space).
2. The World-Tracker transform T_{W_T} (if used)
3. The Sensor-Image transform T_{S_I}
4. The Image-Eye transform T_{I_E}
5. The image size: w_x, w_y (or the linear magnification m and screen dimensions g_x and g_y) and the coefficient of optical distortion, k.

This assumes that the tracker has been calibrated (if necessary) for repeatable, systematic errors and that problems with display nonlinearity, lateral color, and aliasing are either negligible or have been compensated for. I now briefly describe the steps taken in this system for determining these parameters.

7.3.1. Location of Virtual Objects

The location of the virtual objects was done by digitizing key landmark points with the stylus tip of the Faro arm. The objects were defined in terms of these points and thus the alignment phase was rather straightforward. The error in each point's location is bounded by the positional accuracy of the Faro, or about 0.5 mm.

7.3.2. World-Tracker Transform

The World CS was defined in terms of a wooden crate that is used in subsequent calibration procedures (described below). I defined the CS using the following procedure:

- Digitize C, R, and A, where C is the corner of the crate and the origin of the World CS (see Figure 7.3.2.1 below), R is a point along the *x* axis, and A is a point along the *y* axis.
- These three points define the *x*-*y* plane, and we can determine the normal to the plane via

$$\mathbf{n} = \mathbf{v}_{C\_R} \times \mathbf{v}_{C\_A}$$

- The *y* axis is given by $\mathbf{y} = \frac{\mathbf{v}_{C\_A}}{\|\mathbf{v}_{C\_A}\|}$, the *z* axis by $\mathbf{z} = \frac{\mathbf{n}}{\|\mathbf{n}\|}$, and the *x* axis by $\mathbf{x} = \mathbf{y} \times \mathbf{z}$.

Note that the last operation assures that all axes are orthogonal and may move the *x* axis slightly from its measured value. I defined the *x* axis in terms of the *y* axis because the *y* axis is used in the boresight procedure (described below).

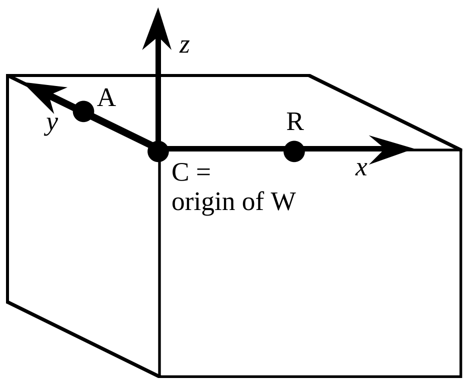


Figure 7.3.2.1. World CS and calibration frame

These vectors are all initially expressed in the Tracker CS; the rotation matrix relating the two CSs is just

$$R_{T\_W} = [\,\mathbf{x} \mid \mathbf{y} \mid \mathbf{z}\,]$$

Since $\mathbf{v}_{T\_W} = \mathbf{v}_{T\_C}$, the transformation is fully specified; we simply invert T_{T_W} to get T_{W_T}.

It is worth noting that this procedure is much more accurate with the Faro arm than it would be for a magnetic tracker, since the Faro comes calibrated for digitizing and since it is so accurate.

7.3.3. Sensor-Image Transform

The determination of T_{S_I} required several steps, which are detailed below.

Pixel cropping: I defined the origin of the Image CS to be the virtual image of the center pixel of the visible portion of the frame buffer. Because not all of the pixels in the frame buffer are visible on the display screen[53] (this is explained in [Rolland et al. 95]), I moved a cursor in screen space and viewed it with a microscope to determine which pixels in the frame buffer are cropped by the display device. In the case of the UNC system, the cropping is significant: The frame buffer is 640x512, yet the first visible pixel on the left LCD is (x = 19, y = 8) (where *y* in this case is measured from the top down) and the bottom right pixel on the screen is (606, 483). This means that the centermost pixel is at (312, 245) ($\pm\frac{1}{2}$ pixel). The cropping was compensated for by setting the viewport for rendering to match the visible pixels on the display.

Determination of image distance: To calculate the image's location in depth, I used the observation from Chapter 5 that the viewing error goes to zero for points in the projection plane. That is, if we draw a cursor at a point and visually align a real marker object with the cursor, the two objects are at the same depth when moving the eyepoint laterally causes no relative motion of the real and virtual points. Once the two points are aligned laterally and in depth, the real point's location can be digitized.

We can use the model to tell us how well such a procedure should work. We can set the angular viewing error to be equal to the limit of normal visual acuity, or about 1 minute of arc (= 1/60˚) and then find what amount of error this corresponds to along each axis. For example, for a point in the center of the image (x = y = 0), d' = 500 mm, and moving the eyepoint off-center by 10 mm, we find that the minimum amount of z error that can be detected under these assumptions is about ± 7 mm. That is, for points within 7 mm of the image plane, the viewing error will probably be too small to detect.

We can improve the precision of the procedure by positioning the real point until it is just noticeably too distant and then until it is just noticeably too close and use the average of these two distances as a good approximation. Furthermore, increasing the amount of eye motion will increase the accuracy of this procedure as well; unfortunately, one can only move so far before the point of interest is no longer visible and, in addition, we get more optical aberrations when we stray too far from the optical axis.

[53] The displays for this STHMD are Sony LCD Color Monitors, Model FDL-330S.

For this study, I found that by using the pixel edges rather than the blurry spot representing the cursor, I could limit the range where the relative motion appeared to be zero to about 20 mm. I then chose the midpoint of this range as the depth of the point. This procedure is rather tedious and error-prone, since distortion at the edges of the field of view can induce false relative motion and because the movements to be detected are so small. Fortunately, the error model (Section 5.6) shows that errors in depth have very little effect on the registration error, since the error lies almost completely along the projector. Thus, the procedure is good enough to determine the depth of a point to within 1-2 centimeters, and any error that is detectable is by definition correctable (since it is the depth differences that we cannot detect that induce the error in the first place).

Defining $\mathbf{T_{S\_I}}$: Given the procedure for locating points within the image plane, I defined T_{S_I} using a similar procedure to that used for defining T_{W_T}:

- Digitize C, R, and A, where C is the image of the center pixel, R is a point along the *x* axis, and A is a point along the *y* axis.
- These three points define the image plane, and we can determine the normal to the plane via: $\mathbf{n} = \mathbf{v}_{C\_R} \times \mathbf{v}_{C\_A}$
- The *x* axis is given by $\mathbf{x} = \frac{\mathbf{v}_{C\_R}}{\|\mathbf{v}_{C\_R}\|}$, the *z* axis by $\mathbf{z} = \frac{\mathbf{n}}{\|\mathbf{n}\|}$, and the *y* axis by $\mathbf{y} = \mathbf{z} \times \mathbf{x}$.
- These vectors are all initially expressed in the Tracker CS; I then attached the Sensor to the STHMD (without moving it) and got the current value for T_{T_S}. I then transformed the vectors into the Sensor CS and created the rotation matrix

$$R_{S\_I} = [\,\mathbf{x} \mid \mathbf{y} \mid \mathbf{z}\,]$$

Since $\mathbf{v}_{S\_I} = T_{S\_T} \cdot \mathbf{v}_{T\_C}$, the transformation is fully specified.

Because the eyepoint is expected to vary from session to session, I initially approximated $\mathbf{v}_{I\_E}$ to be (0, 0, 500), and then corrected it with the boresight procedure, described next.

7.3.4. The Boresight Procedure: Fine-tuning $\mathbf{T_{S\_I}}$ and $\mathbf{v_{I\_E}}$

The STHMD in this system is not particularly rigid because it has moving parts which allow adjustment of the IPD and up/down, in/out placement of the optics relative to the user's eyes. Because the STHMD is connected to a mechanical tracker (the Faro arm), there is an appreciable amount of torque exerted on the STHMD, which causes the true T_{S_I} to deviate from its measured value. As we saw in Section 6.5.2.1, errors in the *x* and *y* positioning of the Image CS relative to the sensor are particularly important since they induce registration error directly and are largely unaffected by the projection operator T_{proj}.

In addition, the eyepoint (as specified by $\mathbf{v}_{I\_E}$) must be measured with the user in the system, since the value of $\mathbf{v}_{I\_E}$ will usually be different each time the user dons the STHMD. For these reasons, a final calibration step was added. This *boresight procedure* is almost identical to that used by [Azuma 95]. That is, the edge of a crate was used as a sighting device to fine-tune the *x* and *y* coordinates of $\mathbf{v}_{S\_I}$ and $\mathbf{v}_{I\_E}$. The *z* coordinate of $\mathbf{v}_{I\_E}$ was determined by placing the eye at the intersection of two vectors in Tracker space; the *z* coordinate of $\mathbf{v}_{S\_I}$ was deemed accurate enough since its effect on the registration error is generally small.

The figure below shows the situation.

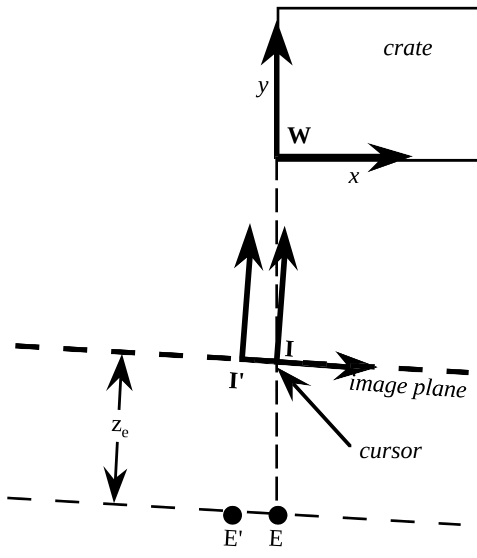


Figure 7.3.4.1. Fine-tuning the *x* and *y* coordinates for E and I

Before the boresight, we have the measured value of $T_{S\_I'}$ (described in the previous section) and the estimated eyepoint specified by $\mathbf{v}_{I\_E'}$. These quantities yield *I'* and *E'* shown in the figure above. Because of error in the measured Sensor-Image transform $T_{S\_I'}$ and in the estimate of $\mathbf{v}_{I\_E'}$, the actual Image CS *I* and eyepoint *E* deviate from *I'* and *E'*.

In the first part of procedure, a cursor is drawn at the center of the screen, which should then appear at the center of the actual Image CS, *I*. The user then moves his head so that the cursor is visually aligned with the edge of the crate (the *y* axis of World space, as defined in Section 7.3.2). At the instant that the cursor is aligned with the crate edge, we have the situation depicted in the figure above: the -*y* axis of World space passes through the origin of *I* (where the cursor is) and through the actual eyepoint *E*. The user presses a button when these three are aligned and a tracker reading ($T_{T\_S'}$) is taken. Using this information, the intersection of the -*y* axis with the *xy* plane of *I'* is calculated and the new *I'* is defined to be at this point. Similarly, the intersection of the -*y* axis of W with the $z = z_e$ plane of I' is calculated and E' is moved to this point. Thus, only the *x* and *y* values of I' and E' are affected (but these are the most important).

It is also important to note that this procedure could also be used to adjust for roll (or *z*-axis rotation) of the Image CS relative to its modeled/measured location. That is, rather than just aligning the center of the crosshair with the *y* axis, one could also (at the same time) align a horizontal line in Image space with the *x* axis of the crate to correct for any *z*-axis rotation of *I'* vs. *I*. Because there was no obvious angular misalignment, I did not implement this adjustment in the test system.

In the second part of the boresight procedure, the *z* value of $\mathbf{v}_{I\_E}$ is calculated. Here, the user aligns his eye with the *y* axis as before, but also aligns it with two nails on top of the crate (see figure below). The nails' positions have been digitized in advance with the tracker and are therefore known in World space. At the moment of alignment, the center of rotation of the user's eye is at the intersection of the two lines. At that instant, the user presses a button, a tracker reading ($T_{T\_S'}$) is taken, and a new vector $\mathbf{v}_{I\_E}$ is calculated. Because it is difficult to hold one's head still while doing all of this, I use only the *z* coordinate of the new $\mathbf{v}_{I\_E}$ and use the more stable values for *x* and *y* calculated in the previous step.

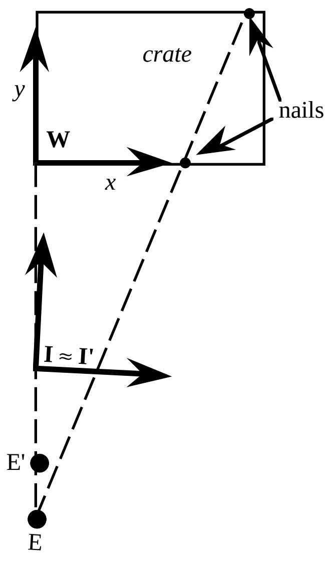


Figure 7.3.4.2. *z* value for eye in Image space ($\mathbf{v}_{I\_E'}$)

Finally, it is worth mentioning one additional technique used to differentiate viewing error from other error source (especially display error). Consider the situation depicted in the figures below.

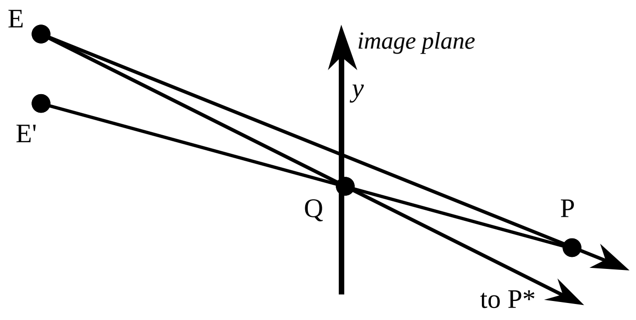
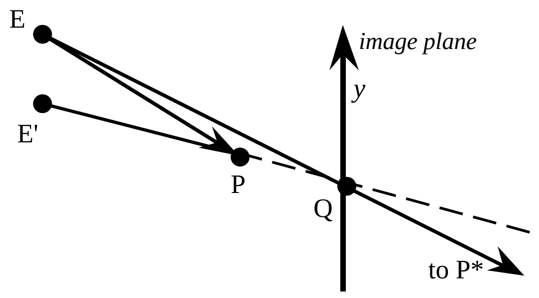


Figure 7.3.4.3.a



Figures 7.3.4.3.b. (Side view) Registration error crosses over as P passes through image plane

Here, the point P is a fixed calibration point in the environment which the system has projected onto the image plane at Q using E' as the eyepoint. The deviation of E from E' induces the registration error shown in the figure; that is, P* appears to be below P. But note what happens when we move the STHMD (*i.e.*, the eyepoint and image plane together) closer to the point so that P is on the other side of the image plane, as depicted in the second figure above. Here, there is still registration error due to viewing error, but now P* appears to be *above* P. The reason is that not only does the registration error go to zero within the image plane, but it changes direction as P passes through the image plane. This gives a way to distinguish viewing error from other types of error (in particular, from display error). In cases when there was still registration error and when I found that the error changed direction when the image plane was moved through the point, I re-ran the calibration procedure.

7.3.5. Determination of Virtual Image Size

The first measurement of the scale of the image was determined by moving the cursor at 20-pixel intervals along the X axis and digitizing the corresponding locations in the real environment. A grid was placed in the real environment in the calculated image plane in order to facilitate the digitization. These X values were then plotted with a paraxial and third-order distortion model as shown in the figure below.

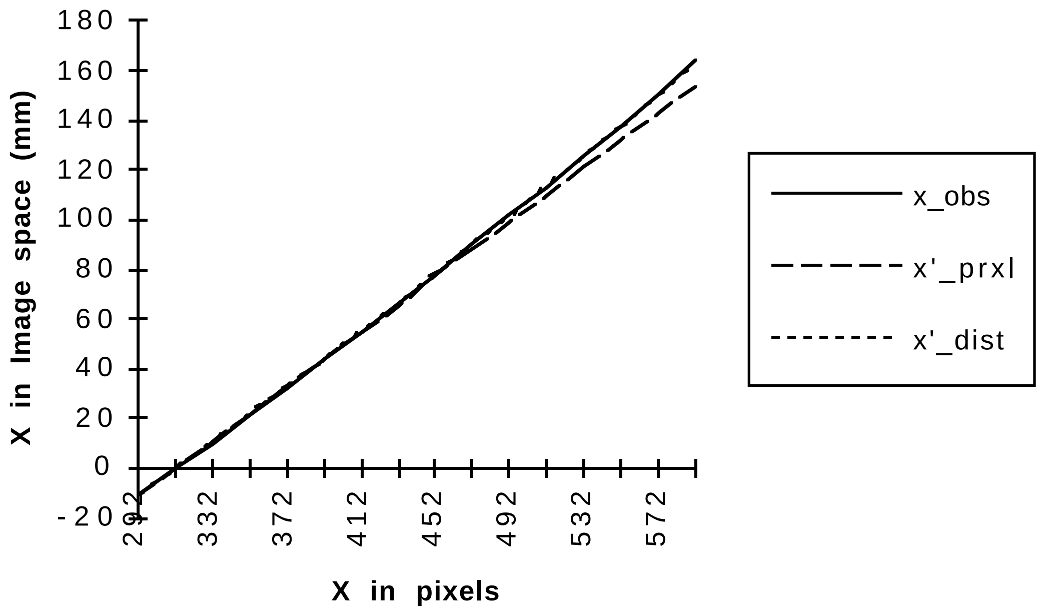


Figure 7.3.5.1. Observed scale for see-through HMD

In the graph, *x_obs* indicates the observed *x* values, *x'_prxl* denotes the paraxial (or linear magnification) case, and *x'_dist* is the values with a third-order distortion model applied. The best-fit value of *m* was 5.9, which is within 2% of the calculated theoretical value of 6.01. The value for *k* was 2.6 x 10^{-6} mm^{-2}, which was also within 2% of the theoretical value. The paraxial image extents were therefore calculated to be

$$w_x = m\, g_x = 5.9\ (54.8\ \mathrm{mm}) = 323.3\ \mathrm{mm}$$

$$w_y = m\, g_y = 5.9\ (41.3\ \mathrm{mm}) = 243.7\ \mathrm{mm}$$

The vertical field of view for the display software is

$$\mathrm{FOV_v} = 2\tan^{-1}\left(\frac{w_y}{2\cdot z_{I\_E}}\right) = 2\tan^{-1}\left(\frac{243.7}{2\cdot 532.9}\right) = 25.8^{\circ}$$

where z_{I_E} is the eye-to-image distance measured in the previous section.

The next adjustment was to compute the pixel aspect ratio. The aspect ratio in screen space was just the extent of the pixels in the *x* direction divided by that for the *y* direction. For *x*, there were 588 pixels across a screen that was 54.8 mm wide, leading to a pixel size of about 0.932 mm/pixel. For *y*, it was 41.3/476 = 0.087. The pixel aspect ratio was therefore *x/y* = 1.07.

Finally, after all the direct measures were done, I fine-tuned the system by adjusting some of the parameter values until the registration error was at a minimum. Specifically, I adjusted w_x, w_y, and k interactively by scaling them up or down until the registration error was at a minimum. I estimate the net *static* registration error for this system to be 1-2 mm. Most of the parameters' final values were within a few percent of their measured values. Having good initial measurements for the parameters was important, and the knowledge of the errors' behaviors provided by the error model made this process straightforward. Having to "tweak" the parameter values is clearly not an ideal solution, but I believe that some sort of user-in-the-loop final adjustment may be an unavoidable step in system calibration.

The level of static registration (1-2 mm) achieved by this system is the best of which I am aware: [Azuma 95] reports static registration errors of ±5 mm,[54] and [Janin et al. 93] report errors of ±13 mm. Although the improved registration in this system is probably due to the accuracy of the tracker used, some of the improvement can be attributed to the improved calibration techniques which are based on insights from the error model.

7.4. Error-Model Verification

The final set of experiments was a run-through of the system to verify that the model was both complete and accurate. That is, I wanted to verify that each error source contributed to the net registration error in the expected fashion *and* that I had not left out any significant error sources. While the derivations of each part of the error model should be complete enough for verifying the model's mathematical correctness, it seemed appropriate to run through a simple set of tests to double-check the model with a real system to make sure that it was both correct and complete. I only checked the major sources of error as listed at the end of Chapter 6; that is,

- Delay-induced error
- Static head-tracking error: Static error in World_Tracker and Tracker_Sensor transforms
- Distortion

[54] However, the dynamic performance of Azuma's system was strikingly better than this system's performance due to the lower-latency tracking and the prediction used (with the aid of accelerometers and rate gyros).

- Viewing error[55]

The smaller error sources (image alignment, aliasing, display nonlinearity, lateral color, and acquisition errors) were not verified except to note the absence of any major effect due to them. Object alignment errors were treated in Section 6.2.

The general approach was to calibrate the system as well as possible and then deliberately introduce errors of each type and record their effect on the overall registration error. The setup used for most of these experiments was as follows: A small video camera (a Panasonic model GP-KS102 color CCD camera) was inserted into a Styrofoam head with the entrance-pupil center positioned roughly at the head's eyepoint (see picture below).

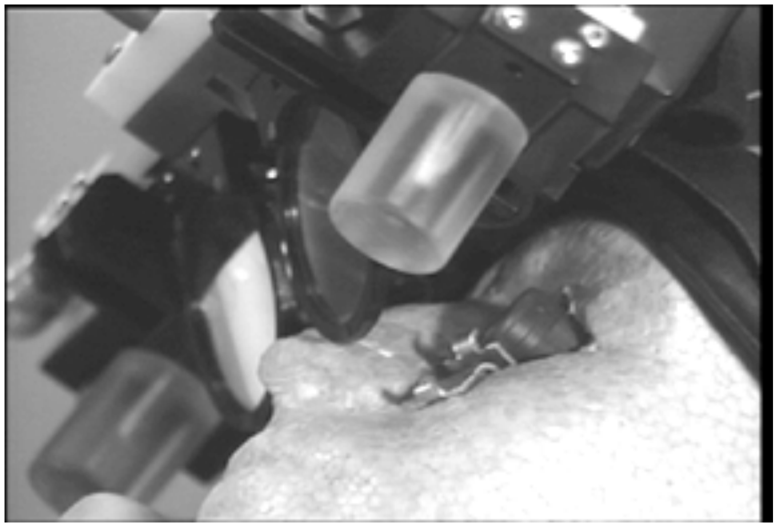


Figure 7.4.1. Camera setup used for verifying model

This setup was calibrated and then errors from the list above were introduced and their corresponding registration errors measured. The test point in World space was a point on a sheet of graph paper surrounded by a ruled grid, which was used to measure the size of the World-space lateral error. For image-space errors, I used a predistorted grid displayed in image space. Because the system is calibrated and the errors are artificially introduced, the system has full knowledge of the errors in the transformations and can calculate and display both the correct location for the displayed point *and* the erroneous location, as shown in the picture below.

[55] This was tested not because it was a large error source, but rather because its behavior as described in Chapters 5 and 6 seemed complex enough to warrant at least a simple check.

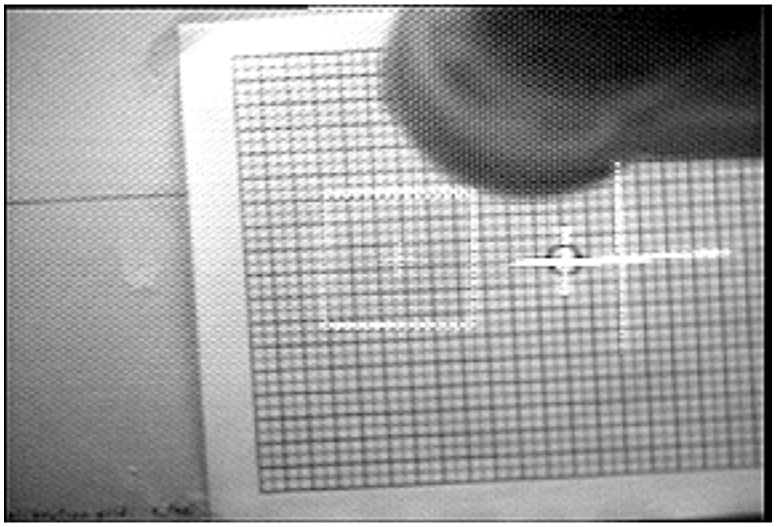


Figure 7.4.2. Camera image of virtual and real scene

Since the system has modeled both P and P*, it can compute the vector between them to derive the linear registration error. Since the location of the eyepoint is also known, we can compute the angular error, the lateral error, and the depth error. Finally, the camera at the eyepoint gives a reality check on the whole process and allows for more precise measurement of the lateral error in World or Image space than could be done by a human observer in the STHMD.

In general, my goal was to try to find the worst-case observable error and to compare it with the error bound for each error type and to verify both that the predicted value was an upper bound on the error and that it was not overly pessimistic. Thus, most of the experiments are geared toward finding system configurations in which the error is at its worst, rather than trying to find typical values. Because of the large number of parameters, it was not feasible to cover every possible range of parameters for every equation; instead, I varied the size of the input errors (*e.g.,* the magnitude of the tracker's translational error $||\mathbf{v}_{S'\_S}||$) of each type and used typical values for the other parameters (*e.g.,* the distance from the head to the tracker, $||\mathbf{v}_{T\_S}||$).

7.4.1. Static Head-Tracking Error

I will begin with static head-tracking error since the behavior of the dynamic error is described by the same equations. Doing it this way enabled me to verify the general behavior for the static case (which is considerably easier to measure) and then to use a simpler experiment for verifying the dynamic case.

The general expression for the linear registration error due to head-tracking error is given by

$$b_{head\_track} = \|\mathbf{v}_{T'\_T}\| + \|\mathbf{v}_{S'\_S}\| + 2\cdot\left|\sin\frac{\text{Ø}_{T'\_T}}{2}\right|\cdot\|\mathbf{v}_{T\_S'}\| + 2\cdot\left|\sin\frac{|\text{Ø}_{T'\_T}| + |\text{Ø}_{S'\_S}|}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \quad (5.4.3.2)$$

After calibrating the system, I introduced both rotational and translational error into T_{W_T} and T_{T_S} in order to verify this equation.

I began with the fairly simple case of pure translational error in T_{W_T} and T_{T_S}. In the worst case, the translation error vectors are in the same direction and are normal to the line of sight, and the lateral error is roughly equal to the linear error. The table below shows a few representative measurements which show this behavior.

World-Tracker error $\|\mathbf{v}_{T'\_T}\|$	Tracker-Sensor error $\|\mathbf{v}_{S'\_S}\|$	Calculated linear error b_{calc}	Linear error bound (Eqn 5.4.3.2) b_{head_track}	Calculated lateral error s_{calc}	Observed lateral error s_{obs}
0	5	5.0	5.0	5.0	6
5	0	5.0	5.0	5.0	5
5	5	10.0	10.0	10.0	12
5	10	15.0	15.0	14.9	16
10	10	20.0	20.0	19.9	23

Table 7.4.1.1. Translational head-tracking error; all distances in mm

In the table, the first two columns give the magnitude of the input errors, the b_{calc} column gives the calculated linear error (*i.e.*, the calculated distance from P to P*), the b_{head_track} gives the bound from Equation 5.4.3.2 (*i.e.*, the worst-case distance from P to P*), the s_{calc} column gives the linear error as calculated by the system, and finally the s_{obs} column gives the observed lateral error seen by the camera. Since the error vectors are chosen to be worst cases, the calculated linear error bounds are all equal to that predicted by the model, and the observed lateral errors match the calculated lateral errors to within 1-3 mm.

The more interesting and less intuitive case is that of the interaction in the rotational error terms. The following table shows the worst-case behavior for one set of measurements.

Angular World-Tracker error $\|Ø_{T'\_T}\|$	Angular Tracker-Sensor error $\|Ø_{S'\_S}\|$	Calculated angular error $Ø_{calc}$ (deg)	Calculated linear error b_{calc}	Linear error bound b_{head_track}	Calculated lateral error s_{calc}	Observed lateral error s_{obs}
0.0	0.0	0.0	0.0	0.0	0.0	1
1.0	0.0	1.3	15.1	16.9	14.0	14
2.0	0.0	2.5	30.3	33.7	27.9	27
3.0	0.0	3.8	45.4	50.6	41.6	41
0.5	2.0	2.2	24.8	25.8	24.4	23
1.0	2.0	2.8	32.2	34.3	31.3	30
1.5	2.0	3.5	39.7	42.7	38.3	39
2.0	2.0	4.1	47.3	51.1	45.2	46
2.5	2.0	4.7	54.7	59.5	52.0	55
3.0	2.0	5.4	62.3	67.9	58.9	59

Table 7.4.1.2. Worst-case rotational head-tracking error

In this case, the translation errors are zero and the rotation errors are given in the first two columns. The Tracker-Sensor and Sensor-to-point distances ($\|\mathbf{v}_{T\_S}\|$ and $\|\mathbf{v}_{S\_P}\|$) were set to around 500 mm (specifically, 467 mm and 498 mm) to simulate typical operating conditions. The first two columns give the amount of angular error introduced into the World-Tracker and Tracker-Sensor transforms, and the calculated angular error (based on the system's knowledge of the actual transform values) is given in the $Ø_{calc}$ column, and the last four columns are as in the previous table.

In the first four rows of the table, I varied the angular error in the World-Tracker transform ($Ø_{T'\_T}$) from 0˚ to 3˚ while holding the angular error in the Tracker-Sensor transform ($Ø_{S'\_S}$) equal to zero. In the bottom six rows, I set $Ø_{S'\_S} = 2˚$ and varied $Ø_{T'\_T}$ from 0.5˚ to 3˚. The worst case behavior only occurs when $\mathbf{v}_{T\_S'}$ and $\mathbf{v}_{S\_P''}$ are aligned, which corresponds to the user looking away from the tracker at the point. In this case, the observed and calculated errors closely match that predicted by Equation 5.4.3.2. Note that the observed lateral errors in this case are generally within about 10% of the modeled linear errors (in the b_{head_track} column) for these worst-case situations.

7.4.2. Delay-Induced Registration Error

I next performed a simple test to show the behavior of delay-induced registration error. The equations that describe this error's behavior are straightforward and well understood, and the parameters' values have already been measured in other studies (head velocities were measured in Section 7.2.2 and delay values in the UNC system were measured by Mine [Mine 93]). I therefore kept this experiment very simple, since its goal was more to illustrate and confirm the behavior of a known phenomenon rather than to discover any new behavior.

The method used was similar to one described in [Bryson & Fisher 90]: A video camera records both the movement in the real scene and the delayed movement in the virtual scene. Timing accuracy is therefore done in 1/30th second video frames (or, at best, 1/60th second video fields), but this is reasonable for a simple test such as this one.

A top view of the setup is shown in Figure 7.4.2.1. After calibrating the system, I moved the STHMD/camera/head setup along the table top parallel to the calibration grid. Parallel motion was enforced by pressing the Styrofoam head base against a bar clamped to the table, as shown in the figure. The output of the camera in the STHMD head was recorded on digital (D2) tape, from which the delay values, velocities, and registration errors were measured and recorded.

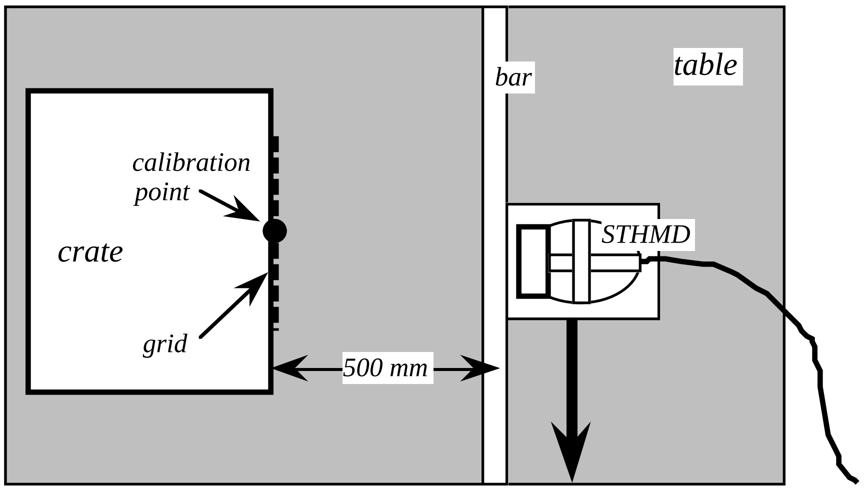


Figure 7.4.2.1. Delay error measurement setup

The data consists of three runs at three different speeds; the average velocities for each were 23 mm/s, 52 mm/s, and 75 mm/s. Speed was controlled by hand; I attempted to move the head as smoothly as possible from beginning to end. The linear velocities in the three runs are typical for this application, with peak velocities for the last run reaching 140 mm/s. I did not attempt to simulate the maximum velocities observed in the user study because of the mechanical constraints imposed by the Faro arm and because of the amount of flex in the STHMD itself.

In each run, I began with the virtual point aligned with the real point at the left edge of the field of view of the camera and moved the head in the *-x* direction until the points reached the right edge of the field of view. Since the motion was one-dimensional, the equation for b_{delay} is quite simple:

$$\mathrm{b}_{\mathrm{delay}} = \|\dot{\mathbf{v}}_{\mathrm{head}}\|\Delta \mathrm{t} \tag{6.3.2.3.1}$$

where $\dot{\mathbf{v}}$ in this case is just $(\frac{\mathrm{dx}}{\mathrm{dt}}, 0, 0)$. Since the error vector should be perpendicular to the line of sight, $b_{delay} \approx s_{delay}$. The delay value Δt used for computing b_{delay} was taken to be the time between the initial movement of the head and the time the virtual point began to move back toward its modeled location. The average delay (as measured at the start of each run) was about 9 frames, or 300 ms. Most of the delay in this system is due to the Faro arm, which does not seem to be built for low latency since it is marketed as a digitizer.[56] The data for the three runs is plotted in the figures that follow and gives the velocity, the predicted linear error b_{delay} and the observed lateral error s_{obs}. All velocities are in mm/s and all errors are in mm.

[56] Faro does not quote a latency figure for the Metrecom. During these tests, the Faro update rate was about 25 Hz, the baud rate was 19,200, and the data packet size for each update was 74 bytes (which implies 39 ms of latency just for serial transmission). If we use Mine's latency values for Pixel-Planes (55 ms) and for the LCD display (17 ms), we can infer that the Faro's latency is about 190 ms. I suspect that the high latency is due to internal filtering designed to improve accuracy and repeatability.

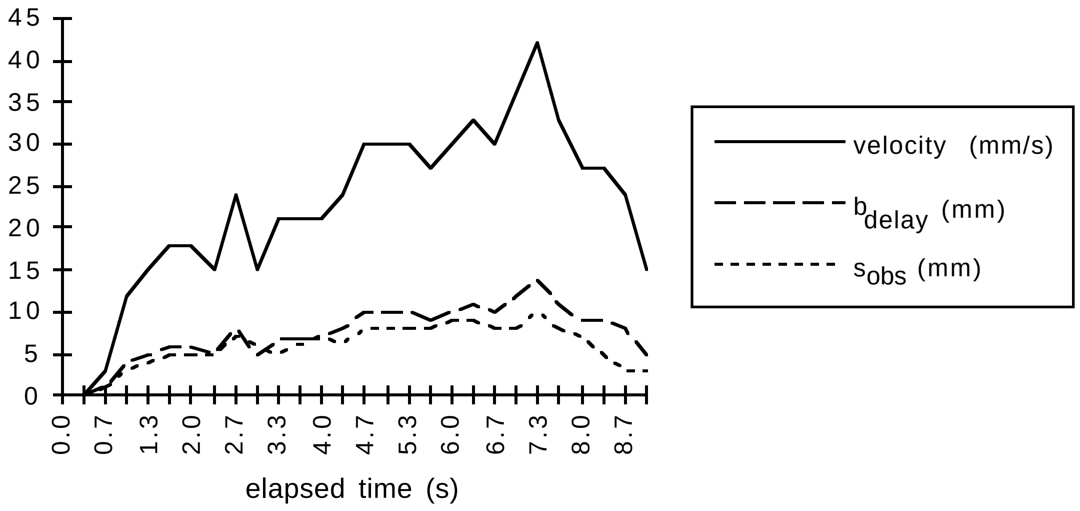


Figure 7.4.2.2. Slow case: average velocity = 23 mm/s

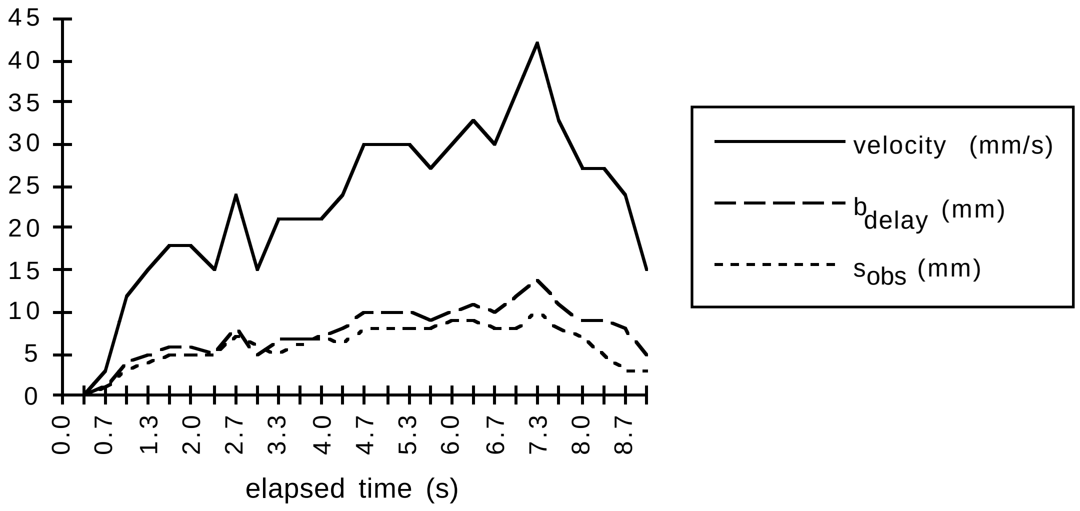


Figure 7.4.2.3. Medium velocity case: average velocity = 52 mm/s

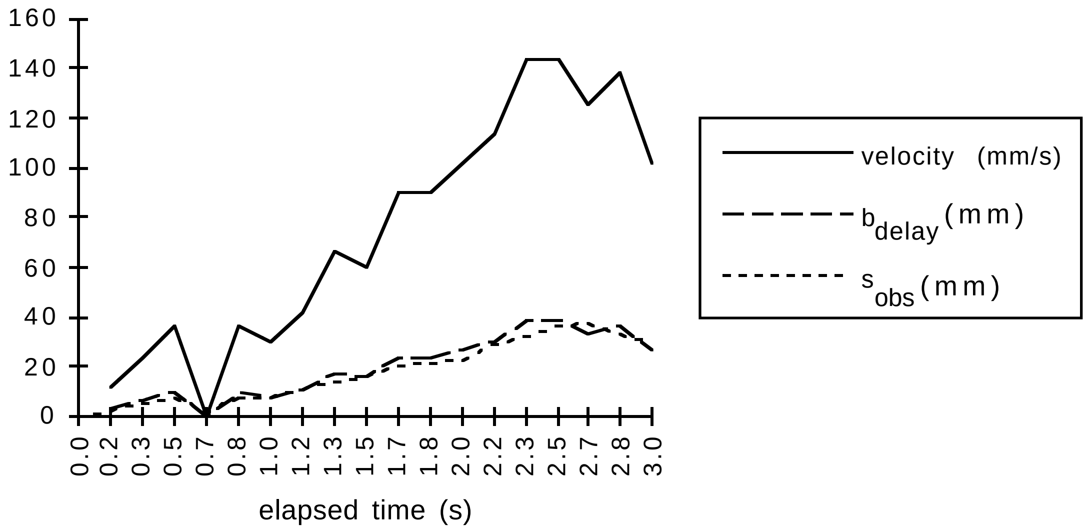


Figure 7.4.2.4. Fast case: average velocity = 75 mm/s

The real linear velocity of the head was measured from the videotape by comparing the position of a fixed crosshair in image space against the grid marks. s_{obs} was simply the distance between the virtual image point location and its modeled location (the center of the grid). The modeled and observed errors generally match well; the small discrepancies can be attributed to measurement error[57] and variation in the delay amount, since the load on the Unix host was not controlled.

7.4.3. Optical Distortion

In Section 7.3, I describe how the coefficient of optical distortion, *k*, for this system was measured. What remains is to demonstrate the effect of this error source on the net registration error.

The behavior of lateral distortion error is described by Equations 6.5.1.1 and 5.7.9.2.10:

$$s_{display} \approx q \cdot \left(1 - \frac{z''}{d'}\right) \tag{6.5.1.1}$$

$$q_{dist} = k\, m^3\, r_G{}^3 \tag{5.7.9.2.10}$$

In the absence of other errors, Q' represents the linearly magnified point, which gives

[57] Because the images did not change much from frame to frame, I only took measurements on every fifth frame for the medium and fast cases, and every 10th frame for the slow case.

$$\mathbf{v}_{I\_Q'} = m \cdot \mathbf{v}_G \quad (7.4.3.1)$$

This allows us to write q_{dist} in the following more convenient form:

$$q_{dist} = k\,(m\,r_G)^3 = k\,r_{I\_Q'}{}^3 \quad (7.4.3.2)$$

where $r_{I\_Q'} = \|\mathbf{v}_{I\_Q'}\|$. The lateral distortion error is then

$$s_{dist} \approx k\,r_{I\_Q'}{}^3 \cdot \left(1 - \frac{z''}{d'}\right) \quad (7.4.3.1)$$

In this experiment, I used the same setup as in previous sections except that the STHMD was cocked on the Styrofoam head so that the camera was pointed at one edge of the STHMD's field of view where the distortion is at its worst (the camera's field of view is only about 20° and is therefore not large enough to capture the entire virtual image of the LCD screen). I then moved the STHMD so that the image of the calibration point in the middle of the grid was in the periphery of the virtual image of the screen where the distortion was at its worst. I first measured the within-image distortion error q_{obs} and then the resulting lateral error s_{obs}. Because the optical distortion had already been measured, the system had the information necessary to calculate the expected values for both q_{dist} ands_{dist}, which are denoted by q_{calc} and s_{calc}. The following plot gives the calculated and observed within-image error (q_{calc} and q_{obs}):

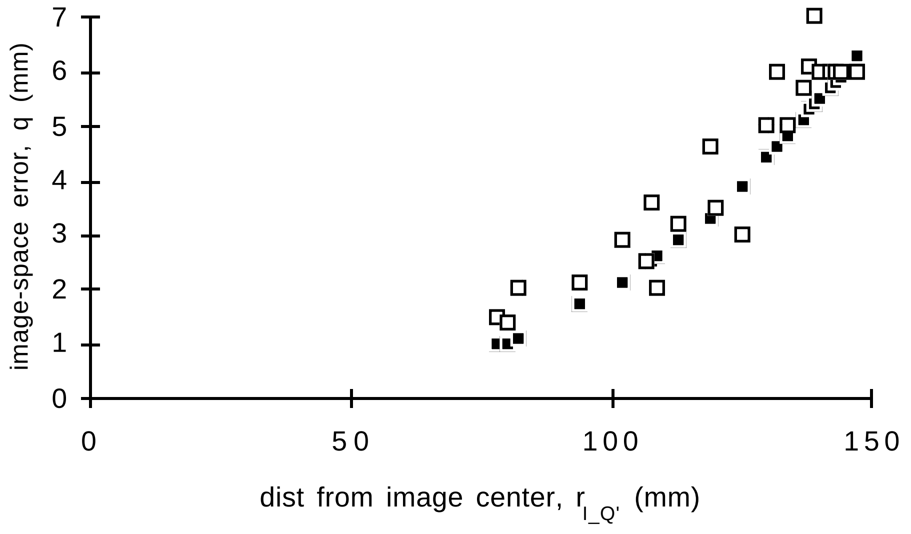


Figure 7.4.3.1. q_{calc} (solid dots) and q_{obs} (hollow dots) vs. $r_{I\_Q'}$ (all values in mm)

Note the expected cubic curve for both sets of points. The absolute difference between the calculated and observed values is less than a millimeter for most data points, with a mean of 0.6 mm. The next plot shows the calculated and observed lateral errors (s_{calc} and s_{obs}).

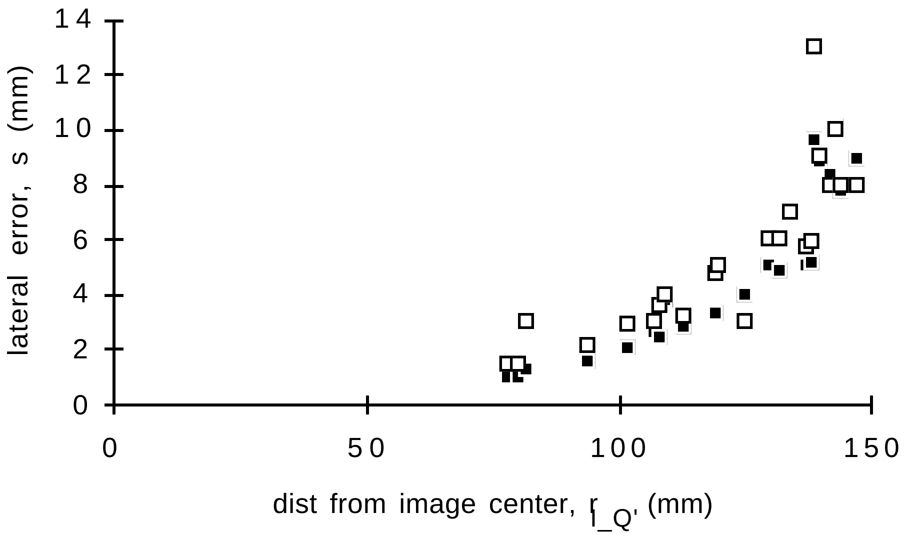


Figure 7.4.3.2. s_{calc} (solid dots) and s_{obs} (hollow dots) vs. $r_{I\_Q'}$; all values in mm

Here, the cubic pattern is still evident, but less pronounced. The reason for much of the variation is that the lateral error depends not only on q_{dist}, which is cubic, but also on the z coordinate of the point, which varied in this case because I moved the STHMD around within the working volume (which changed z, since it is measured relative to I). The mean difference between the calculated and observed s values was 0.8 mm.

7.4.4. Viewing Error

Even though viewing error is not likely to be a major error source, it seemed worthwhile to verify the expression for lateral viewing error with a simple experiment.

The expression for the lateral viewing error is

$$s_{view} \approx e \cdot \frac{|z''|}{d'} \qquad (5.6.3.3)$$

where e is the magnitude of eyepoint error vector $\mathbf{v}_{E'\_E}$.

The assertion that the viewing error goes to zero in the image plane (where z'' = 0) was already verified in the calibration process (Section 7.3). In keeping with the approach used in other sections, I varied the error magnitude (in this case, *e*) and set the other parameters to typical values: in this case, |z"| = 369 mm; d' = 501 mm.

A common source of viewing error is the variation in the way the STHMD fits on the user's head from session to session. To simulate this, I digitized the approximate location of the camera's entrance pupil with the Faro digitizer, clamped the Styrofoam head to the table top and then moved the STHMD relative to the head and recorded the distance between the real point P on the grid and the displayed point P* (as in previous experiments). The figure below shows the results.

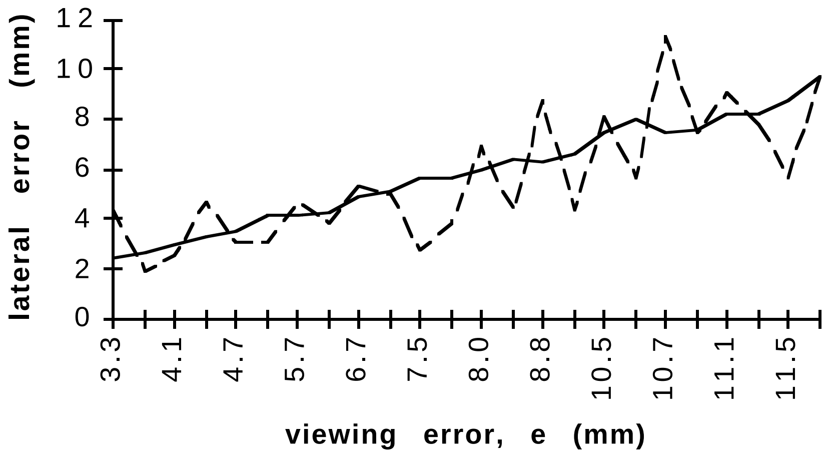


Figure 7.4.4.1. Calculated (solid line) vs. observed (dashed line) lateral viewing error

The data is somewhat noisy, but follows the expected pattern of linear variation with increasing values of *e*. Almost all of the deviations from the predicted values for s_{view} are within 1-2 millimeters, which is the average residual error in the system under normal operation. The extreme values of *e* correspond to rather contorted positions of the STHMD with respect to the head and are therefore not meant to be representative of normal use; I used them only to verify the error behavior for large values of *e*. In general, the system was fairly robust to misalignments of the STHMD on the head, which confirms the earlier assertion that viewing error should not be a major error source in most systems.

7.5. Summary

This chapter details the experiments done to arrive at input error bounds for tracker jitter and head velocities, describes the calibration techniques used for the prototype system, and the experiments used to double-check the model's results.

The tracker jitter test showed that the Fastrak's outputs are quite stable for working distances of less than 500 mm (with sigma values of less than 0.5 mm and 0.05˚), reasonably stable for 500 - 1000 mm (sigma values less than 1.2 mm and 0.15˚), and unstable for distances exceeding 1400 mm (sigma values in excess of 6 mm and 0.4˚). The head velocity experiments show that most of the velocities are below 500 mm/s and 50˚/s, with average velocities of 164 mm/s and 20˚/s. The working distance ranged from 250 mm to 650 mm, with an average of 430 mm.

The calibration work described in Section 7.3 gives a set of techniques for accurate calibration of an STHMD system. After calibration, the system achieves static registration of 1-2 mm, which is the best of which I am aware. The error model was quite useful during this process, since it can be used to determine which parts of the system need to be calibrated carefully and which ones can be approximated. In particular, I found that the parameters whose values are difficult to measure are often those for which precise calibration is not necessary. For example, the *z* coordinate (*i.e.,* the depth) for the Image coordinate system is difficult to measure with precision, but the net registration error is fairly insensitive to error in this parameter. In contrast, error in the *x* or *y* coordinate induces registration error directly and is therefore easy to detect and correct. The final insight was that direct measurement of system parameters is an important first step in the calibration process, but fine-tuning the parameter values with the user in the system is an important final step for accurate calibration.

Finally, Section 7.4 describes a set of experiments for verifying the most important components of the error model with a real system. The general approach was to use the calibrated system and then to deliberately introduce errors and measure their effects. In almost all cases, the observed errors were within 1-2 mm of the predicted errors.

8. Future Work

In the previous chapters, the error model was derived and used to analyze the current state of the UNC STHMD system. This chapter looks more broadly at the problem of registration error. It begins with a forecast for future systems—a discussion of what sort of performance will be required in order to realize a system that can register real and virtual objects to within a millimeter. This forecast highlights the most difficult problems in achieving good registration, which leads to the discussion in Section 8.2 on future work. The chapter finishes with a summary of the dissertation as a whole.

8.1. Forecast for future systems

In Chapter 6, I examine the amount of registration error in an AR system which uses current technology. Another good use for the error model is to define what level of performance is required of each part of the system in order to achieve a certain level of registration. As an example, take the case of surgery planning. As noted earlier, our surgeon, Dr. Davis, estimates that for a system to be useful, the registration would have to be within 1 mm. We can use the error model to get an estimate of the amount of improvement each part of the system needs in order to reach this goal.

I will now examine each of the error sources discussed in this document in order to determine the level of performance required in order to meet registration goals of 1 mm and 0.1 mm. The 1 mm bound indicates the performance required for each component if it is the only error source; the 0.1 mm bound gives an idea of how difficult it would be to get an order-of-magnitude improvement in each component. The next section will offer suggestions on how to achieve improvements in some of the system components.

1. **Acquisition error:** In Section 6.2, I estimate the error in the acquisition process to be on the order of one-half the voxel diagonal. For a voxel size of 1x1x2, this amounts to an error of 1.2 mm. To reduce this error to 1 mm (under the assumptions of Section 6.2), the voxel size would have to be reduced to 0.82 x 0.82 x 1.6 mm (a factor of 1.2). For a registration error of 0.1 mm, the voxel size would have to be reduced to 0.082 x 0.082 x 0.16 mm (a factor of 12).

2. **Alignment error:** Since the RMS error in the alignment process converges to the RMS error in the virtual dataset, there should be no additional error in this stage if enough landmarks are used.

3. **World-Tracker error:** For the case cited in Section 6.3.1, we have

$$b_{W\_T} = \|\mathbf{v}_{T'\_T}\| + 2\cdot\left|\sin\frac{\text{Ø}_{T'\_T}}{2}\right|\cdot\left(\|\mathbf{v}_{T\_S'}\| + \|\mathbf{v}_{S\_P''}\|\right) \qquad (6.3.1.1)$$

   with $\|\mathbf{v}_{T\_S'}\| + \|\mathbf{v}_{S\_P''}\| = 1000$ mm.

   If we split the error budget evenly between the angular and translational terms for simplicity, the error bounds for the 1 mm case are $\|\mathbf{v}_{T'\_T}\| \leq 0.5$ mm and $\text{Ø}_{T'\_T} \leq 0.029^\circ$ and the error bounds for the 0.1 mm case are $\|\mathbf{v}_{T'\_T}\| \leq 0.05$ mm and $\text{Ø}_{T'\_T} \leq 0.0029^\circ$ (or about 10 arcseconds).

   If we assume (as was done in Chapter 6) that the limiting factor in this part of the calibration is the tracker, then the 0.1 and 1 mm cases would require 2.6× and 26× improvement over the Fastrak's specified translational accuracy (1.32 mm) and 5.2× and 52× improvement over the angular accuracy (0.15˚).

4. **Tracker-Sensor error:** The static case for tracker error in T_{T_S} is not as demanding as that for World-Tracker error. From Chapter 6 we have

$$b_{static} = \|\mathbf{v}_{S'\_S}\| + 2\cdot\left|\sin\frac{\text{Ø}_{S'\_S}}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \qquad (6.3.2.1)$$

   which (under the same assumptions as above) requires accuracy of 0.5 mm of translation error and 0.057˚ for the 1 mm case, and 0.05 mm of translation error and 0.0057˚ for the 0.1 mm case. This requires the same improvements in translational accuracy as above and factors of 2.6× and 26× for rotation.

   The dynamic case is considerably harder. The simple expression for delay error alone is

$$b_{delay} = \|\dot{\mathbf{v}}_{head}\|\Delta t + 2\cdot\left|\sin\frac{\dot{\text{Ø}}_{head}\Delta t}{2}\right|\cdot\|\mathbf{v}_{S\_P''}\| \qquad (6.3.2.3.1)$$

   For the 1 mm case and the maximum head velocities of 500 mm/s and 50˚/s and $\|\mathbf{v}_{S\_P''}\| = 500$ mm, the required delay value Δt is 1 ms; if we use the average head velocities, the value for Δt is 3 ms. The worst-case head motions would require a factor of 55 speedup over the average delay in Pixel Planes 5 and a factor of 17 over the Pixel-Planes 5 low-latency rendering system, and this is assuming perfect synchronization with the display devices.

For the 0.1 mm case and the maximum head velocities of 500 mm/s and 50˚/s and $\|\mathbf{v}_{S\_P''}\|$ = 500 mm, the required delay value Δt is 0.1 ms, which is one third of the transmission time for a single Fastrak report on a fast parallel interface (IEEE 488). If we use the average head velocities, the value for Δt is 0.3 ms. The worst-case head motions would require a factor of 550 speedup over the average delay in Pixel Planes 5 and a factor of 170 over the Pixel-Planes 5 low-latency rendering system. (Other approaches to this problem are discussed in the next section.)

5. **Viewing error:** The approximate expression for lateral viewing error is

$$s_{view} \approx e \cdot \frac{|z''|}{d'} \qquad (5.6.3.3)$$

If we assume that the image plane is centered in the patient's head at about 500 mm (d' = 500 mm) and that the system does not adjust it dynamically according to gaze direction, then the range of *z* values is ±100 mm, and the 1 mm case implies *e* ≤ 5 mm (which is the estimated value used in Chapter 6), and the 0.1 mm case requires *e* ≤ 0.5 mm (a 10× improvement).

6. **Image alignment error:** The general expression for display error is

$$s_{display} \approx q \cdot \left(1 - \frac{z''}{d'}\right) \qquad (6.5.1.1)$$

and the approximate contribution to *q* for image misalignment is

$$q_{im\_align} \approx \|\mathbf{v}_{I'\_I}\| + 2 \cdot \left|\sin\frac{Ø_{I'\_I}}{2}\right| \cdot \|\mathbf{v}_{I'\_Q'}\| \qquad (6.5.2.1.1)$$

Let us assume z" = -100 and d' = 100 as above, use $\|\mathbf{v}_{I'\_Q'}\|$ = 200 mm (the maximum image-space radius in the UNC STHMD) and split the error evenly between translation and angular terms.

For 1 mm of registration error, we arrive at $\|\mathbf{v}_{I'\_I}\| \leq 0.42$ mm and $Ø_{I'\_I} \leq 0.12˚$. In the 0.1 mm case, we get $\|\mathbf{v}_{I'\_I}\| \leq 0.042$ mm and $Ø_{I'\_I} \leq 0.012˚$ (or 43 arcseconds). Note that for both the linear and angular terms, within-image-plane error vector in the 0.1 mm case would subtend about 30-35 arcseconds, which is near the limit of normal human visual acuity (but still above the limit for vernier acuity).

7. **Aliasing error:** The error term in screen space is

$$\|\mathbf{v}_{G'\_G}\| \le \frac{\sqrt{2}}{2} h = \frac{\sqrt{2}}{2} \frac{g_x}{p_x} \qquad (5.7.8.1.3)$$

Assuming linear magnification, this gives

$$q_{alias} = m \cdot \frac{\sqrt{2}}{2} \frac{g_x}{p_x}$$

Using the magnification and screen size of the UNC STHMD shows that for the 1 mm case, NTSC resolution is sufficient; for the 0.1 mm case, the resolution of the display would have to be increased to 2,790 x 2,092 (a factor of 4.4 vs. NTSC). Of course, antialiasing could also be used to reduce this, probably at the expense of increased latency.

8. **Display-device nonlinearity:** The screen-space error here is

$$\|\mathbf{v}_{G'\_G}\| \le \sqrt{\eta\, g_x^2 + \eta\, g_y^2} = \eta \sqrt{g_x^2 + g_y^2} \qquad (5.7.8.2.1)$$

Assuming linear magnification as before, the 1 mm case would require no improvement for the Tektronix CRTs discussed in Chapter 6. In the 0.1 mm case, the accuracy of the display device would have to be 0.056% (a factor of 3 improvement) and 0.02% for the LCDs in the UNC STHMD.

9. **Distortion:** The expression for distortion as an error percentage is

$$q_{dist} = \rho \cdot \|\mathbf{v}_{I\_Q''}\| = \rho \cdot \frac{w_x}{2} \qquad (6.5.2.4.1)$$

In the 1 mm case, the distortion would have to be reduced to 0.5%, which is a factor of 13 better than the 6.5% distortion in the current system. In the 0.1 mm case, the distortion would have to be reduced to 0.05%, which is a factor of 130 better than the 6.5% distortion in the current system. Correcting elsewhere in the graphics pipeline would have to be figured into the delay error calculations.

10. **Lateral color:** The expression for *q* due to lateral color is

$$q_{LC} = \frac{\alpha r_G + \beta k r_G^3}{3} \qquad (5.7.9.3.12)$$

In the 1 mm case, lateral color is not a problem since its current contribution is 0.28 mm.

In the 0.1 mm case, since the linear lateral color term can be corrected for by adjusting the image scale, we need only consider the cubic term. For the UNC STHMD, $\beta = 5.83$, $r_G = 34.2$ mm, and we find that to reduce the lateral color contribution to the lateral error to 0.1 mm, *k* must be reduced by a factor of 2.5. Since the previous step (# 9) requires a reduction of *k* by a factor of 276, the lateral color would be taken care of automatically (other things being equal).

This analysis shows clearly what the biggest problems are:

- Delay: The straightforward approach of trying to beat this problem by building faster systems seems doomed to failure for the foreseeable future: In the worst case, the system does not even have time to read the tracker before it goes over budget! The only way to combat this problem will be with predictive tracking. Other delay-reduction strategies will also be important; some of these approaches are discussed in the next section.

- Tracking accuracy: Performance improvements (over manufacturer's specifications) of at least 3× will probably be required and possibly as high as 50×. Future directions in tracking are also discussed in the next section.

- Distortion: Eliminating it in the optical design is difficult and usually adds weight and cost; the option of correcting it computationally only exacerbates the problem of delay-induced error.

In addition to these major problems, this budget analysis brings out another important point: errors which appeared negligible in the earlier analysis start to increase in importance once we attempt to attain a higher level of registration. For example, lateral color error (0.28 mm) seemed inconsequential in the original analysis (*e.g.*, when compared to a delay error of 60 mm), but it is actually over budget by a factor of 3 for the 0.1 mm case. I predict that the more we push these systems to get better registration, the more difficult problems we will find.

I further conjecture that small errors which might be difficult to detect by themselves may sum together to make large errors which *are* noticeable. Such residual calibration errors will be more difficult to correct, since their sources will not be as easy to identify as they were when the errors were large.

Finally, it is worth noting that tracker error, aliasing error, device nonlinearity, distortion, and lateral color can all theoretically be corrected or drastically improved by the appropriate use of filters and warping functions. The problem is that such operations take time, which is the quantity we can least afford to trade off. The next section discusses some approaches to the problems brought out by this analysis.

8.2. Future work

The error model presented here was tailored to a particular application (surgery planning) and as such was not thoroughly explored for different systems and different applications. In particular, this model could easily be expanded in order to analyze video STHMD systems, CAVE systems[58], and opaque HMDs. For video STHMDs, the model for viewing error would have to be changed, but much of the rest of the model would work as is. In CAVE systems, many of the problems of head tracking disappear (since the images are fixed in the environment), but the analysis of viewing error would be rather useful, especially since multiple users often view a scene which is only correct for the user wearing the head-tracker. Finally opaque HMDs do not have the strict requirements for registering real and virtual objects (since the real objects are not visible), but nevertheless suffer from the apparent swimming of virtual objects due to delay and tracker error, as well as the visual distortions from viewing error and optical distortion.

For the most part, however, the future work suggested by this dissertation is not in the area of extending the work presented here, but rather in addressing the problems that it describes. I will now briefly discuss what I consider to be promising future directions in the area of improving registration in augmented reality systems.

As we have seen, delay is the largest error source and seems likely to remain so for some time. There are at least four approaches to combating delay:

1. *Reduce the system delay*. This amounts to making the system components more responsive. An example of this approach is the work cited earlier by [Cohen & Olano 94]. Their low-latency rendering system combats latency by reducing the image generation time from 50-75 ms to 17 ms. The problem with this and other systems optimized for quick turnaround is that they lose the efficiency afforded by optimizing for high throughput. Cohen and Olano’s low-latency rendering system can only handle about 100-250 triangles in its current implementation. More work needs to be done on computer graphics architectures which are optimized for low latency and medium-to-high throughput.

   In a similar vein, more research on reducing tracker latency is warranted. Recent improvements in commercial magnetic trackers show that progress can be made: in

[58] CAVE systems use several rear-projection screens and head-tracked stereo glasses to immerse the user in a 10’x10’ virtual environment [Ghazisaedy et al. 95].

the past 4 years, the Polhemus and Ascension trackers have reduced their latency from 30-50 ms (for older models) to 10-20 ms (see Mine[93] for details).

2. *Just-in-time data.* Another way to fight latency is to feed head-motion data into the rendering pipeline at the latest possible moment. An early example of this is *beam racing* (used in flight simulators), in which pixel values are computed and stored into the frame buffer just before they are scanned out to the display. A more recent example is the system described in [Regan & Pose 94], in which the head orientation data is fed into the pipeline *after* rendering a larger view of the scene; the orientation data is then used to select which portion of the extra-large frame buffer to scan out. Another example is *just-in-time pixels* [Mine & Bishop 93], in which the system uses the most current estimate or measurement of head position in order to compute the value for each pixel (or scanline). A third example is a technique called *frameless rendering* [Bishop et al. 94], in which the system draws a randomly distributed subset of the pixels in the image at each frame which allows each pixel to be computed with the most recent head-motion data.

3. *Predict the future.* Promising work by [Azuma 95] shows that head-motion prediction with inertial systems gives a 5- to 10-fold increase in accuracy in an augmented reality system. His system achieved dynamic registration to within 4-5 mm for moderate head velocities. An important result from Azuma's work is that predicting does not allow us to relax the restraint that the system operate with quick turnaround. His results show that low latency is just as important for good prediction as it is for reducing delay error. In particular, it shows that prediction errors increase roughly as the square of the prediction interval, which means that prediction may not be effective for long delays (which for their data was > 80 ms). Thus, systems must still be optimized for low latency even when prediction is used. More work in the areas of head-motion models for prediction and sensors for assisting prediction will be essential for improving on this work.

4. *Video see-through with feedback.* A radically different approach is that used by [Bajura & Neumann 95]. In their video STHMD, they have the option of delaying the real-world imagery (not an option with optical STHMDs) to match the delay in displaying the virtual scene. They also warp the virtual scene based on landmark points in both scenes in order to compensate for other errors. This use of feedback will likely be an important component of both video and optical STHMD systems. Even though optical STHMDs cannot guarantee registration in the same way that

video STHMDs can, they can still benefit from feedback (via one or more cameras) on how the system is doing and how to improve registration on the next frame.

The first two approaches concentrate on reducing the end-to-end delay by reducing the component delays and by overlapping computations in the pipeline. The predictive tracking approach works in a different direction: it is a way of buying computation time by guessing where the head will be in the future. Because registration error is so sensitive to delay and because we will never have a system with instantaneous turnaround, predictive tracking will be essential for good dynamic registration.

As we have seen, tracking error plagues both run-time performance and system calibration procedures. At present, no single technology type (magnetic, optical, ultrasonic, mechanical, inertial, etc.) seems to be suitable for achieving the required accuracy, responsiveness, and wieldiness. An obvious future direction would be to use a hybrid tracking scheme which uses some combination of the technology types to achieve better performance than either alone. In addition to coupling different types of trackers, an obvious follow-on to Azuma’s work would be to add sensors for detecting velocities and accelerations as part of the tracker itself.

In a similar vein, more research is needed in the area of efficient methods for calibrating trackers both with respect to a laboratory reference frame and with respect to other trackers. These methods need to be robust in the presence of noisy data and should exploit knowledge of the volumes within which the tracker data is more reliable.

As noted earlier, the problem of correcting distortion optically is quite difficult, and correcting it computationally may make the registration error worse by introducing unacceptable delays. Recent work by [Watson & Hodges 95] shows promise because it takes advantage of texturing hardware in order to speed up the process. While it appears that this approach still introduces delay, it is a step in the right direction. More research needs to be done in the area of image processing hardware that can quickly predistort images without introducing distracting artifacts. Also, while [Kocian 88] mentions inducing barrel distortion in the CRT image to correct for optical distortion for military systems, this technique does not seem to have worked its way into commercial HMDs yet.

The area of calibration is still fertile ground for research. Most of the existing procedures for calibration are tedious and error-prone. One idea might be to make use of computer vision techniques coupled with an error model like this one in order to do much of the calibration automatically. That is, a dummy head with cameras at the eyepoints (or possibly more than two cameras) could be moved through a calibration environment and data from the cameras could be collected and used to derive optimal values for the system's parameters.

8.3. Conclusion

A summary of the major results and contributions of this work is given in Chapter 1; I will briefly re-summarize here.

Most of the major error sources are associated with the tracker in some way. The tracker is a critical component for making augmented reality work, and any error or delay in its data causes serious flaws in the illusion. The errors associated with the tracker are due to delay in displaying the tracker data (due to delay in the entire system), error in the tracker measurement, error in locating the Tracker CS in the World CS, and errors in tracker readings used for system calibration.

Another result of this analysis is that eye tracking is probably unnecessary if the eye's center of rotation is used as the center of projection, since it gives the correct projection for fixated points and small errors for non-fixated points. The last of the major error sources is optical distortion, which can cause large errors in the image periphery; unfortunately, inverting the distortion in the rendering process may cause more delay error than the warping error it corrects. Finally, there are several other small error sources, each of which may add a small amount of registration error.

The analysis shows that sub-millimeter registration is not likely any time soon (for optical STHMD systems without feedback) since there are many error sources on the order of a millimeter; but it does seem probable that we will achieve 1-5 millimeter accuracy for moderate head motions with the use of predictive tracking, synchronized display methods, and careful calibration. The progress toward sub-millimeter registration error will probably be asymptotic, with increasing effort and expense required to gain each small increase in precision.

A. Quaternion Basics

(Reproduced from [Robinett & Holloway 95])

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 multiplication, multiplication by a scalar, taking the norm, normalization, and inversion are given below in terms of quaternions *q* and *r*, and scalar α:

$$q*r = \left[\left(q_x, q_y, q_z\right), q_w\right]*\left[\left(r_x, r_y, r_z\right), r_w\right] = \begin{bmatrix} \begin{pmatrix} q_x r_w + q_y r_z - q_z r_y + q_w r_x, \\ -q_x r_z + q_y r_w + q_z r_x + q_w r_y, \\ q_x r_y - q_y r_x + q_z r_w + q_w r_z \end{pmatrix}, \\ -q_x r_x - q_y r_y - q_z r_z + q_w r_w \end{bmatrix}$$

$$(q*r)^{-1} = r^{-1} * q^{-1}$$

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

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

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

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

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.

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.

REFERENCES

Adams, Ludwig, et al. 1990. Computer-Assisted Surgery. *IEEE Computer Graphics and Applications.* May.

Adelstein, B, E Johnston, S Ellis. 1995. Dynamic response of electromagnetic spatial displacement trackers. Submitted for publication.

Allen, Dave. 1994. Tektronix, Beaverton OR. Personal communication. July.

Altobelli, David, R Kikinis, J Mulliken, W Lorensen, H Cline, F Jolesz. 1991. Intraoperative navigation in craniomaxillofacial surgery. *AAOMS.* September 25, p57.

Altobelli, David, R Kikinis, J Mulliken, W Lorensen, H Cline. 1991. Computer assisted three-dimensional planning for cranio-maxillofacial surgery: Osteotomy simulation. *AAOMS.* September 27, p89.

Azuma, Ron. 1995. *Predictive tracking for augmented reality*. PhD dissertation, University of North Carolina at Chapel Hill. TR95-007.

Azuma, Ron, and Gary Bishop. Improving static and dynamic registration in an optical see-through HMD. *Proceedings of SIGGRAPH '94.* Orlando, July 24-29, pp 197-204.

Bajura, Michael, and Ulrich Neumann. 1995. Dynamic registration correction in AR systems. *Proc IEEE Virtual reality annual international symposium* (VRAIS).

Bajura, Michael, H. Fuchs, R. Ohbuchi. 1992. Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient. *Computer Graphics.* Vol 26, No 2. July . 203-210.

Barrette, R.E. 1992. Wide field of view, full-color, high-resolution, helmet-mounted display. *Proceedings SID '92.* (Society for Information Display).

Bartlett, Scott, I Wornom III, and L Whitaker. 1991. Evaluation of facial skeletal aesthetics and surgical planning. *Clinics in plastic surgery.* 18:1. January.

Besl, P, and N McKay. 1992. A method for registration of 3-D shapes. *IEEE Trans. on Pat. Anal and Mach. Int.* 14:2.

Bishop, G, H Fuchs, L McMillan, E Zagier. 1994. Frameless rendering: Double buffering considered harmful. *Proceedings of SIGGRAPH '94.*

Bishop, Gary. 1992. No Swimming. UNC Department of Computer Science internal document.

Boff, KR & JE Lincoln. 1988. Comparison of perceptual capabilities across sensory modalities. Engineering Data Compendium: Human Perception and Performance. AAMRL, Wright-Patterson AFB, OH.

Breipohl, Arthur. 1970. *Probabilistic systems analysis*. Wiley & Sons. New York.

Bryson, Steve. 1992. Measurement and calibration of static distortion of position data from 3D trackers. *Proc. SPIE Vol. 1669: Stereoscopic displays and applications III.*

Bryson, Steve, and S Fisher. 1990. Defining, modeling and measuring system lag in virtual environments. *Proc Stereoscopic displays and applications II.* SPIE v1256.

Caudell, Thomas, D Mizell. 1992. Augmented reality: An application of heads-up display technology to manual manufacturing processes. *IEEE Hawaii International Conference on System Sciences.* January.

Chesnais, Pascal. 1988. A graphic/photographic arthroscopy simulator. Master's thesis, MIT.

Chi, Vern. 1993. Specifications for the Snail-Eyes color display. UNC Computer Science internal document.

Cohen, Jonathan, and M Olano. 1994. Low latency rendering on Pixel-Planes 5. UNC technical report TR 94-028.

Cutting, C, F Bookstein, B Grayson, L Fellingham, J McCarthy. 1986. Three-dimensional computer-assisted design of craniofacial surgical procedures: Optimization and interaction with cephalometric and CT-based models. *Plastic and reconstructive surgery.* June.

Deering, Michael. 1992. High resolution virtual reality. *Computer graphics.* 26:2:195.

Faro Technologies, Inc. 1993. Industrial Metrecom Manual for Model IND-01. Faro Technologies, Inc, Lake Mary, FL, 32746.

Feiner, Steven, Blair MacIntyre, and Doree Seligmann. 1993. Knowledge-Based Augmented Reality. *Communications of the ACM.* Vol 36, No. 7. July. pp 52-61.

Friets, Eric, J Strohbehn, J Hatch, D Roberts. 1989. A frameless stereotaxic operating microscope for neurosurgery. *IEEE Transactions on Biomedical Engineering.* 36:6. June.

Fuchs, H, G Bishop, PIs. 1992. Research directions in virtual environments. *ACM SIGGRAPH- Computer Graphics.* August. 26:3.

Fuchs, H, J Poulton, J Eyles, T Greer, J Goldfeather, D Ellsworth, S Molnar, G Turk, B Tebbs, L Israel. 1989. Pixel-Planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics: Proceedings of SIGGRAPH '89.* 23:3.

Fuchs, H, M Levoy and S Pizer. 1989. Interactive visualization of 3D medical data. *Computer*. August. pp 46-51.

Furness, T. 1986. The super cockpit and its human factors challenges. *Proceedings of the Human Factors Society.* 30, 48-52.

Galloway, Robert, R Maciunas, J Latimer. 1991. The accuracies of four stereotactic frame systems: An independent assessment. *Biomedical instrumentation and techonology.* Nov/Dec.

Ghazisaedy, M, D Adamczyk, D Sandin, R Kenyon, T DeFanti. 1995. Ultrasonic calibration of a magnetcic tracker in a virtual reality space. *Proceedings of Virtual reality annual international symposium.* pp 179-88.

Golub, Gene H., Charles F. Van Loan. 1983. *Matrix computations*. Johns Hopkins University Press. Baltimore.

Grinberg, V., G Podnar, M Siegel. 1994. Geometry of binocular imaging. *Proc. SPIE Stereoscopic displays and VR systems.* February. Vol. 2177.

Hecht, Eugene, and Alfred Zajac. 1974. *Optics.* Addison-Wesley. Menlo Park, CA.

Herman, Gabor. 1991. *Quantitation using 3D images.* In *3D imaging in medicine* (Udupa, J, and G Herman eds). CRC Press. Boca Raton, FL. pp. 146-161.

Hodges, LF, and ET Davis. 1993. Geometric considerations for stereoscopic virtual environments. *Presence*. Volume 2, Number 1.

Horn, Berthold. 1987. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am.*

Janin, Adam, D Mizell, T Caudell. 1993. Calibration of head-mounted displays for augmented reality applications. *Proc. IEEE Virtual reality annual international symposium (VRAIS)*.

Kalawsky, Roy. 1993. *The science of virtual reality and virtual environments*. Addison-Wesley. Wokingham, England.

Kaufman, Lloyd. 1974. *Sight and Mind*. Oxford University Press. Oxford, England.

Kocian, Dean. 1988. Design considerations for virtual panoramic display (VPD) helmet systems. *Proc. AGARD: The man-machine interface in tactical aircraft design and combat automation.*

Lastra, A. 1994. University of North Carolina. Personal communication.

Lee, Seungho H, and K Rao, Eds. 1987. *Cranial computed tomography and MRI.* (2nd edition). McGraw-Hill, New York, NY.

Longhurst, R. 1957. *Geometrical and physical optics*. Longmans. New York.

Lorensen, W, and H Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics.* 21:4.

Lorensen, W. 1990. Creating surfaces from volumes using marching and dividing cubes. Course notes for Tutorial on volume visualization algorithms. *Proc. first conference on visualization in biomedical computing.* Atlanta, GA.

Ma, Jintao, and Ian Hunter. 1993. Optical design for a head-mounted display. *Presence.* Vol. 2, No. 3. MIT Press.

Marsh, Jeffrey, and M Vannier. 1989. Three-dimensional surface imaging from CT scans for the study of craniofacial dysmorphology. *J. Craniofacial genetics and developmental biology*. 9:61-75.

Milgram, Paul and D Drascic. 1991. Enhancement of 3-D video displays by means of superimposed stereo-graphics. *Proceedings of the Human Factors Society 35th annual meeting.*

Min, P. and H Jense. 1994. Interactive stereoscopy optimization for head-mounted displays. *Proc. SPIE Stereoscopic displays and VR systems*. February.

Mine, Mark. 1995. Personal communication.

Mine, M, & G. Bishop. 1993. Just-in-time pixels. UNC technical report 93-005.

Mine, Mark. 1993. Characterization of end-to-end delays in head-mounted display systems. UNC Technical Report TR93-001. Chapel Hill.

Muthu, S.K. 1982. *Probability and Errors for the Physical Sciences.* Sangam Books Limited. Delhi, India.

Olano, Marc, J Cohen, M Mine, G Bishop. 1995. Combatting rendering latency. *Proceedings of 1995 Symposium on Interactive 3D Graphics*. Monterey, CA. April 9-12.

Polhemus, Inc. 1992. 3SPACE Fastrak user's manual. November. Polhemus Inc, PO Box 560, 1 Hercules Drive, Colchester, VT, 05446.

Poole, Harry H. 1966. *Fundamentals of display systems*. Macmillan and Co. London.

Reading, R.W. 1983. *Binocular Vision*. Butterworth Publishers. Woburn, MA.

Reinhart, William. 1992. Gray-scale requirements for anti-aliasing of stereoscopic graphic imagery.   *Proc Stereoscopic displays and applications III*. SPIE v 1669.

Robinett, W, and R Holloway. 1994. The visual display transformation for virtual reality. *Presence*. 4:1.

Robinett, Warren & JP Rolland. 1991. A computational model for the stereoscopic optics of a head-mounted display. *Presence.* 1:1.

Rolland, J. P., D. Ariely & W. Gibson. 1995. Towards quantifying depth and size perception in virtual environments. *Presence.* 4:1.

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.

Scherr, Sol. 1982. *Video and digital electronic displays: A user's guide*. John Wiley & Sons. New York.

Scherr, Sol. 1970. *Fundamentals of display system design*. Johny Wiley & Sons. New York.

Schmandt, Christopher. 1983. Spatial input/display correspondence in a stereoscopic computer graphic work station. *Computer Graphics*. July.

Shinners, Stanley M. 1967. *Techniques of system engineering.* McGraw-Hill. New York.

Southard, D. 1994. Viewing model for stereoscopic head-mounted displays. *Proc. SPIE Stereoscopic displays and VR systems.* February.

State, Andrei, David T. Chen, Chris Tector, Andrew Brandt, Hong Chen, Ryutarou Ohbuchi, Mike Bajura, and Henry Fuchs. 1994. Case Study: Observing a Volume-Rendered Fetus within a Pregnant Patient. *Proceedings of IEEE Visualization '94.* edited by R. Daniel Bergeron and Arie E. Kaufman. IEEE Computer Society Press, Los Alamitos, Calif. pp 364-368.

Sutherland, I. 1968. A head-mounted three dimensional display. *Proc. Fall Joint Computer Conference*.

Tannas, Lawrence E. [Ed]. 1985. *Flat-panel displays and CRTs*. Van Nostrand Reinhold Co. New York.

Taylor, John R. 1982. *Introduction to Error Analysis*. University Science Books. Mill Valley, CA.

Toennies, Klaus, J Udupa, G Herman, I Wornom, S Buchman. 1990. Registration of 3D objects and surfaces. *IEEE Computer Graphics & Applications*. May. p 52.

Turk, Greg, and M Levoy. 1994. Zippered polygon meshes from range images. *Proceedings of SIGGRAPH '94.* Orlando, July 24-29. pp 311-18.

Udupa, J and G Herman, Eds. 1991. *3D imaging in medicine*. CRC Press. Boca Raton, FL.

Vannier, M, C Hildebolt, D Gayou, and J Marsh. 1991. Introduction to 3D imaging. In *3D imaging in medicine* (Udupa, J, and G Herman eds). CRC Press. Boca Raton, FL. pp. 71-88.

Ward, M, R Azuma, R Bennett, S Gottschalk, H Fuchs. 1992. A demonstrated optical tracker with scalable work area for head-mounted display systems. *Proc. 1992 Symposium on 3D Graphics.* Boston. ACM, Addison-Wesley.

Watson, BA and LF Hodges. 1995. Using texture maps to correct for optical distortion in head-mounted displays. *Proc IEEE Virtual reality annual international symposium* (VRAIS).

Wojcik, W, and J Harris, Jr. 1991. Three-dimensional imaging of the musculoskeletal system. In *3D imaging in medicine* (Udupa, J, and G Herman eds). CRC Press. Boca Raton, FL. pp. 191-222.

Woodson, Wesley, and D Conover. 1964. *Human engineering guide for equipment designers*. University of California Press. Berkeley.