# A Real-time Optical 6D Tracker for Head-mounted Display Systems

*TR90-011*

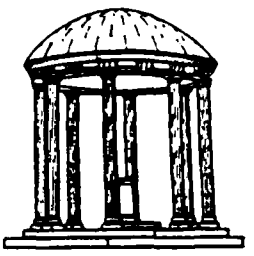
*March, 1990*



N00014-86-~~C~~-0680
-K-0680

*Jih-Fang Wang*





The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

# A Real-time Optical 6D Tracker for Head-mounted Display Systems

*by*

*Jih-Fang Wang*

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

1990



Approved by:

Advisor Henry Fuchs

Reader: Gary Bishop

Reader: Vernon Chi

**Jih-Fang Wang. A Real-time Optical 6D Tracker for Head-mounted Display Systems (Under the direction of Henry Fuchs)**


**abstract**

Significant advance has been made towards realistic synthesis and display of three-dimensional objects using computers during the past two decades. However, the interaction between human and computer-generated scenes remains largely remote through devices such as keyboards, mice, joysticks, etc. Head-mounted display provides a mechanism for much more realistic visualizing and interacting with computer-generated 3D scenes through the hand-eye-body coordination exercised daily. Head-mounted display systems require that the position and orientation of the user's head be tracked in real time with high accuracy in a large working environment. Current 6D positional tracking devices (3 translational and 3 rotational parameters) fail to satisfy these requirements.

In this dissertation, a new system for real-time, six-dimensional position tracking is introduced, studied, and documented. This system adopts an inside-out tracking paradigm. The working environment is a room in which the ceiling is lined with a regular pattern of infrared LED beacons which are flashing (invisible to the human eyes) under the system's control. Three cameras are mounted on a helmet which the user wears. Each camera uses a lateral effect photodiode as the recording surface. The 2D image positions of the flashing beacons inside the field of view of the cameras are recorded and reported in real time. The measured 2D image positions and the known 3D positions of beacons are used to compute the position of the camera assembly in space.

We have designed an iterative algorithm to estimate the 6D position of the camera assembly in space. This algorithm is a generalized version of the Church's method, and allows for multiple cameras with nonconvergent nodal points. Several equations are formulated to predict the system's error analytically, and the system's error is quantitatively studied and compared with the theoretical prediction. The requirements of accuracy, speed, adequate working volume, light weight and small size of the tracker are addressed. A novel approach to calibrate the positions of beacons automatically is also proposed.

A prototype was designed and constructed to demonstrate the integration and coordination of all essential components in the new tracker. This prototype uses off-the-shelf components and can be easily duplicated. Our results indicate that the new system significantly out-performs other existing systems. The new tracker prototype provides about 25 updates per second, and registers 0.1-degree rotational movements and 2-millimeter translational movements. The future full system will have a working volume about 1,000 $ft^3$ (10 ft on each side), and will provide more than 200 updates per second with a lag of less than 5 milliseconds by running on a faster processor.

We aim at developing a new tracking system which offers a large working volume, high accuracy in the estimated head position, fast update rate, and immunity to the electro-magnetic interference of the environment. Such a system should find applications in many 6D position-reporting and input tasks.

## Acknowledgements

I would like to express my gratitude to my advisor, Professor Henry Fuchs, for his guidance and support throughout the course of this research. This dissertation could never have come to be without him. I thank Professor Gary Bishop, Professor Frederick Brooks, Mr. Vernon Chi, Professor John Eyles, and Professor John Halton for their willingness to serve on my committee from the initial qualifying exam stage to the completion of this dissertation. Mr. Vernon Chi taught me analog hardware and optics, and provided many significant insights for this project. I thank him for that. I thank Professor Frederic Brooks for the opportunity to learn from him. Thanks go to Professors Gary Bishop and John Eyles for their attending the tracker meeting on a regular basis, and for their stimulating discussions and many constructive suggestions and criticisms.

The assistance of Mr. Robert Bennett and John Thomas of the Microelectronic Systems Laboratory was particularly important in constructing and trouble-shooting the experimental apparatus. Their help in system programming and hardware is deeply appreciated. Mr. Ron Azuma, who joined the project in the summer of 1989, provided helpful comments on my research as well as words of encouragement.

I gratefully acknowledge the financial support provided by the Office of Naval Research (grant N00014-86-K-0680), and by NIH Division of Research Resources (grant RR02172-05).

No expression of gratitude is adequate for the debt I owe my family, whose love and devotion never fail to encourage me and propel me through my course of study. Especially I thank my twin brother, Professor Yuan-Fang Wang, for his help throughout this project, and for his careful proofreading of this manuscript.

Finally, I thank my loving and caring wife Hsiao-Hung for her understanding and her faith in me. I dedicate this dissertation to her.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter, head-mounted display systems are introduced and their role in human/computer interaction is described. Important research problems pertinent to applying head-mounted display systems to facilitate human/machine interaction are identified. The particular issue which is the focus of this research, *tracking the position and motion of the user's head in space*, is studied in more detail. Configuration of our tracking system is also described. We conclude this chapter by defining the criteria of an ideal tracker, and outline the structure of the remainder of this thesis.

## 1.1 The Head-mounted Display Systems

Significant advances have been made towards realistic synthesis and display of three-dimensional objects using computers during the past two decades. It is now a common practice to generate complicated 3D scenes with hidden surfaces removed, and visible surfaces realistically lighted and smoothly shaded. However, the interaction between human and computer-generated scenes remains largely remote through devices such as mice, joysticks, rotary dials, tablets, etc. These devices do not allow a user to interact directly with the virtual environments created by a computer.

An ideal interactive paradigm was proposed by Ivan Sutherland [Sut65] in 1965. The concept of the *Ultimate Display* is that the host computer controls the existence of objects in a virtual world. Computer-generated chairs could be sat upon, and computer-generated bullets would be fatal. In this system, users experience and interact with computer-generated objects just as they do with real physical objects. Unfortunately, the Ultimate Display concept is a goal far from attainable even with today's technology. However, we are now in a position to investigate certain feasible approximations to the Ultimate Display. Head

mounted display presents such an approximation.

For more than two decades, research effort has been expended to develop head-mounted display svstems for easy interaction with computers. The head-mounted display system developed at UNC [Hol87][CHB+89] has been used to allow a user to "walk-around" in a room-size environment. The walk-around concept, while not as powerful and realistic as the one advocated in the Ultimate Display, does provide a natural and effective man/machine interaction mechanism. The computer-generated images are presented on the screens of small television sets mounted in front of the user's eyes. As the user moves, his or her head position is constantly measured, and appropriate views of the 3D environment are displayed to provide illusion of a virtual world.

The head-mounted display provides a more realistic mechanism for visualizing and interacting with virtual 3D objects than do conventional methods of display and manipulation. It allows viewpoint selection through natural head and body movements. Since the method harnesses natural well-trained hand-eye-body coordination that is exercised daily, it may prove to be a more effective paradigm for human-machine interaction.

## 1.2 The Problems

A head-mounted display system consists of three major components: a display engine, a 6D position tracking subsystem, and a helmet. The head-mounted system developed at UNC uses the Pixel-Planes 4 machine [FGH+85]—which is capable of generating thirty thousand Gouraud shaded polygons per second—as the display engine. The helmet is equipped with two color-liquid-crystal television sets, each with a two-inch-diagonal display screen. The user's head position is tracked by a Polhemus position tracker. The Polhemus tracker consists of a source which is fixed at a certain location inside the working environment, and a sensor is attached to the helmet. The sensor detects a low-frequency magnetic field generated by the source. Based on the polarization and the orientation of the magnetic field, the sensor's position and orientation relative to the radiating source are determined.

Much improvement needs to be done before this head-mounted display system can be used for practical human/computer interaction. The display engine needs to be made even faster, and the LCD television sets need to have a higher resolution and a wider field of view. As for the tracker, the Polhemus has a slow update rate and suffers from serious lag (latency). The lag, which is the time between the user making a certain movement and that movement being registered and reported by the Polhemus, can be as long as 120 milliseconds [CHB+89]. One should note that the problems of update rate and latency are different. For example, one can have a tracker with a fast update rate that still reports

events which occurred far back in time. The Polhemus also has a limited working range (the user has to be within a short distance from the radiating source for the sensor to detect the emitted waves). Finally, the function of Polhemus is affected by magnetic perturbations in the environment. Interferences come from metallic furniture in the room, and possible radiating sources such as TV sets, computers, and terminals. Hence, the tracker needs to be made faster, more accurate, and to work in a larger range.

## 1.3 The Criteria

It is not difficult to identify the characteristics that an ideal tracking device should possess:

- a fast update rate,
- low latency,
- high accuracy,
- a large working volume, and
- the ability to report both position and orientation.

An ideal tracker should also

- impose little restriction on the user,
- impose little restriction on the environment,
- be compact and rigid, and
- be light-weight and comfortable to the user.

As evident from the previous work on 3D position trackers (Section 2.1), it is difficult for any single system to meet all the requirements.

## 1.4 The Proposed System

In this section, we present the configuration of the system we propose to build, and describe the structure of a prototype we constructed to prove our design concept. Note that there are two systems discussed here: one is the *proposed*, or *full scale* system; the other

is the *prototype* , or *bench-top* system. In this dissertation, a bench-top prototype is built which implements a subset of the proposed system.

The goal of this research is to design and construct a new tracker to be used with head-mounted display systems. The working environment is a room where the ceiling is lined with a regular pattern of beacons flashing under the system's control. Three cameras are mounted on a helmet which the user wears; these cameras record the 2D image positions of flashing LED beacons. Lateral-effect photodiodes are used as the recording surfaces of the cameras so that the 2D positions of the LEDs can be measured in real time. A new algorithm is designed to infer, from the 2D image positions of the flashing beacons, the 6D position and orientation of the camera unit. Figure 1.1 depicts the configuration of the proposed tracking system.

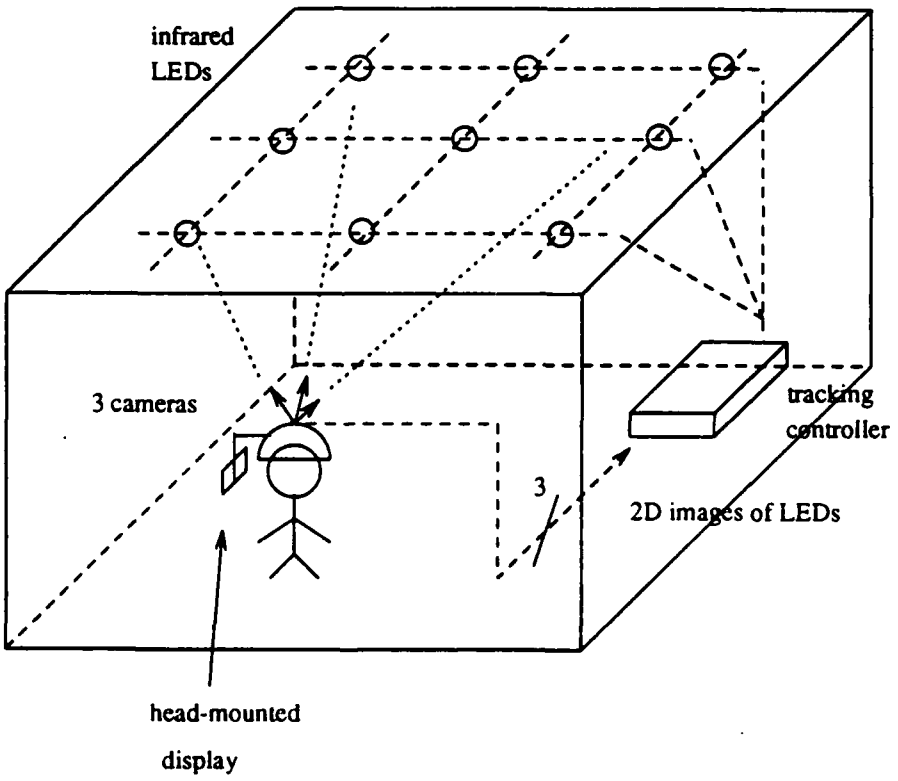



Figure 1.1: The new tracking system

A bench-top prototype of the proposed system has been designed and built. The prototype aims at proving the conceptual correctness of our design, and demonstrating that a fast update, low latency, and a large working volume can be achieved using this new tracker. The bench-top prototype is made of three cameras mounted on a small helmet.

The three cameras are used to observe three infrared LEDs affixed to the ceiling. Although the prototype system implements only a small part of the proposed system, it nonetheless demonstrates the integration and coordination of all essential components in the new design.

The proposed system provides a large working volume about 1000 $ft^3$ (10 ft on each side), high accuracy in the computed head position (error less than 0.1° in rotation and 2 mm in translation), a fast update rate (> 200 Hz), low latency ($\sim$ 5 millisecond), and immunity to electro-magnetic interference in the environment.

The remainder of this thesis is organized as follows: In chapter two, we survey the state-of-the-art in motion tracking and discuss the fundamental working principles of the new tracker. Several algorithms for tracking 6D motion are presented in chapter three, and the particular tracking algorithm which we designed is formulated. The system's error is analyzed theoretically and compared with the data gathered from a simulation program.

A prototype tracker was designed, built, and tested. Chapter four discusses the design and implementation of the prototype system. In chapter five, performance of the prototype is quantitatively measured and analyzed. Finally in chapter six, we discuss the future research plan to achieve a full scale system.

# Chapter 2

# Tracking Devices and Methods

We begin this chapter with a survey of existing position-tracking systems. The *inside-out* tracking paradigm which we adopt in our system is then presented and compared with the conventional *outside-in* tracking paradigm adopted by most other systems. Several important components in our system are described here. Details of the system design and implementation will follow in later chapters.

## 2.1 Overview of Existing 3D Position Tracking Systems

In this survey, we place our emphasis on systems that can track in six dimensions, which includes three positional (translational) and three orientational (rotational) degrees of freedom. We also include discussion of those systems that report only position information. Although the primary application of 6D tracking systems is in head-mounted display, such tracking devices have also found applications in interactive surface design and 3D modeling, and have been used as unconstrained 3D graphics input tools.

First, we discuss the general working principles of devices which provide 3D positional information. Commercial and experimental 3D position tracking devices have used acoustic, magnetic, mechanical, and optical methods to report 3D position. These systems are discussed in the following subsections.

### 2.1.1 Acoustic Systems

Acoustic ranging systems such as the *Lincoln Wand* [Rob66] use the time-of-flight principle to estimate the range of objects in space. A 3D acoustic tracking system uses at least three microphones. These microphones are placed perpendicular to one another in

space. 3D position of a sound-generating stylus is estimated using the time required for the sound wave to be recorded by the three microphones. Such systems are position recording devices. They do not sense orientation directly. The speed of sound also limits the update rate and the working volume. Finally, because the speed of sound varies as a function of the ambient air density, these systems have poor accuracy over a large working range.

### 2.1.2 Magnetic Systems

Polhemus Navigation Science manufactures the Polhemus 3D position sensor [Pol80] which is used in our current head-mounted display system. The Polhemus uses a three-axis magnetic-dipole source and a three-axis magnetic-dipole sensor. The three dipoles in the source are excited sequentially to produce magnetic fields. The sensor dipoles detect the polarization and orientation of the magnetic fields. Based on the recording of the sensor dipoles, both the position and orientation of the sensor relative to the source can be determined using a built-in microprocessor[RBS+79].

According to the specification [Pol80], the Polhemus provides a static position accuracy of 0.25 inch (6.35 mm), and an angular accuracy of 0.7 degree if the sensor is located between ±4 and ±28 inches (±0.1 – ±0.7 meters) from the source. Operation of the sensor up to ± 60 inches (±1.5 meters) away from the source is possible with reduced accuracy. Maximum output update rate is 60 Hz.

The main advantage of the Polhemus is that the sensor attached to the user is small and light-weight. Also, the operation of Polhemus does not suffer from the line-of-sight constraint—the sensor does not need to "see" the radiating source with no opaque obstacle in between—as most optical trackers do. However, the performance of the Polhemus is affected by any conducting materials present in the environment. In our Graphics Laboratory, we place all metal furniture and computers outside the working limits of the Polhemus to avoid the undesirable interference. However, there is no escaping the effect of the metal floor which can curve the Polhemus' space by as much as 30° near its spatial range, according to [CHB+89]. Further, the Polhemus has a limited working range, a slow update rate and suffers the lag problem. Experiments conducted in our head-mounted display project [CHB+89] indicate that the Polhemus can provide about 16 position updates per second, and the lag in response can be as long as 120 milliseconds.

### 2.1.3 Mechanical Systems

The first mechanical linkage head-mounted display system was built at the University of Utah [Sut68][Vic74]. The Argonne Remote-Manipulator (ARM) at UNC [Kil76] and the

Noll box [Nol71] also fall in this category. These types of systems have a limited working range due to the mechanical linkages attached to the user. It is also difficult to track several objects simultaneously with these kinds of systems.

### 2.1.4 Optical Systems

Commercial and experimental optical position tracking systems provide a new solution to the 6D tracking problem. Most optical systems use several light sources as beacons attached to the helmet the user wears. These light sources may be either blinking or constantly lit. These beacons are observed by cameras with known, fixed positions. The observed image positions of the beacons and the known positions of the cameras are used to compute the position of beacons, hence that of the user, in space. Note that for optical tracking systems with beacons, the line-of-sight constraint does apply. That is, in order for the sensors to register and compute the beacon position in space, they have to "see" the beacon with no opaque obstacles obstructing the line of sight. This restriction is alleviated by the natural environment trackers (discussed later in this section) which do not use beacons.

Below we introduce optical tracking systems. They are divided into beacon trackers and natural environment trackers, and are listed with commercial systems followed by experimental ones in order of their publication.

#### 2.1.4.1 Beacon Trackers

SELSPOT [Wol74][LO74], is a commercial system consisting of camera-like units using lateral effect photodiodes as the recording surfaces (the properties of lateral effect photodiodes will be discussed in detail in section 2.2.2.1). The system detects a single light source and determines the 2D location where the light beam strikes the photodiode surface. A pair of these cameras can be used to estimate the 3D location of the light source using the stereopsis principle. The current system does not provide any software support for real-time measurement. The whole system, including two cameras, a control unit which controls up to 120 light sources, and an interface to an IBM personal computer, costs about $65,000. Each additional camera costs about $15,000.

OP-EYE [Uni81] is another commercial tracking system using lateral effect photodiodes. OP-EYE can track the motion of a single light source with an advertised resolution of 1 part in 4000 and an update rate of 5000 Hz—implying a lag of only 0.2 millisecond. Actual system performance appears much worse than that advertised. Like SELSPOT, it provides a limited working volume and do not sense orientation directly.

OPTOTRAK [Nor88] is a new system which claims to be much more accurate than all other optical tracking systems (with a minimum resolution of 1 part in 10,000). It uses one camera with two dual-axis CCD infrared position sensors. Each position sensor has a dedicated processor board to calculate the image position of the light source. Again, the triangulation principle is applied to recover the position of the light source in space. The camera weighs about 10 pounds, and the whole system costs $30,000 and each additional camera costs $10,000.

Twinkle Box [Bur73], a system developed by Burton, uses four fixed sensors to observe a set of sequentially blinking light sources attached to the user. Each sensor consists of a high-speed photomultiplier tube and a wide-angle lens. A rotating disk with slits around its periphery is mounted in front of each sensor. As the disk rotates, each slit sweeps out a field of view. When the photomultiplier detects a light beam, an electrical pulse is passed to the host computer. The position of the disk at the moment determines the angular position of the light source. Angular measurements from three different sensors are sufficient to compute the position of the light sources.

Fuchs, Duran and Johnson [FJ77] developed a system using multiple one-dimensional CCD array sensors. A sharp edge is placed in front of each CCD sensor. A light source held by the user casts a shadow through the sharp edge on each CCD array. The cast shadow position is used to estimate the position of the light source along a particular direction with respect to each CCD sensor—the direction of the plane which is defined by the shadow position and the position of the sharp edge in front of the sensor. The 3D position of the light source is recovered by intersecting planes from all CCD sensors. For a sensor with 256 CCD elements, the working volume is about 1 cubic meter with an accuracy of 6 millimeters. This system can infer orientation by time-multiplexing several light sources (flashing them in a fast sequence). This is a simple and robust system which requires no lens.

A position reporting system using the triangulation principle similar to the ones described above was constructed by Gary Bishop [FBP+82] at the Microelectronic Systems Laboratory (MSL) of UNC. In this system, two wall-mounted vidicon cameras are used to track the position of four lamps attached rigidly to antennae affixed to the helmet the user wears. The lamps flash in sequence. Images of the lamps are digitized and the centroid positions are estimated using an Ikonas graphics system. 3D positions of the lamps are computed using triangulation. The major drawback of this system is speed—digitizing video signals takes time. Hence, this system cannot report 6D position in real time (the update rate is about 2-3 per second [Bis84]). Also, the amount of data in a digitized image sequence is very large. For example, a black-and-white digital image of 500 by 500 resolution contains a quarter of a million bits of information. The whole memory bandwidth of the host computer is easily consumed by the data transferred between the cameras and the host.

Another system similar to the Twinkle Box using lateral effect photodiodes was developed by Vernon Chi and John Poulton [FBP+82] at the MSL. The user wears a helmet with a light source attached to it. This system estimates the 3D location of the light source, hence that of the helmet, in the same manner as the SELSPOT system. A polarization method is employed to sense the orientation of the helmet. The light source used is an unmodulated, high-intensity light bulb. The light bulb is surrounded by a cone cut from a single sheet of normal transparent polarizing material. The cone has the property that the direction of polarization of the light emerging from the cone is a linear function of the azimuthal angle of the cone. Angle of polarization is recovered by the photodiodes which view the target through a rotating disc made of polarizing material. Light intensity registered at the sensor is maximum when the disc's polarizer direction is aligned properly with that of the light source. The major problem with this system is that a very bright light source and a 100% duty cycle is needed to increase the signal-to-noise ratio. Such a high-intensity lamp requires a power cable or a battery pack. Some form of cooling is also needed. Also, the user will be distracted when the light source enters his/her field of view.

#### 2.1.4.2 Natural Environment Trackers

All the above surveyed optical trackers use some sort of beacons as the basis of tracking. They are not "natural" in the sense that they alter the environment by incorporating man-made tracking beacons. Further, as beacons are used, all these systems suffer from the line-of-sight constraint. Natural environment trackers do not require artificial beacons to guide the tracking. Instead, ambient light reflected from the working environment is recorded and the change in the ambient light pattern is used to guide the tracking process. In the following, we introduce two such systems.

Gary Bishop [Bis84][BF84] proposed a new scheme called *Self-Tracker* in which several 1D sensors are used for position reporting. Instead of mounting sensors on the wall like all the other systems do, sensors are mounted on the helmet. Custom VLSI circuitry is built to enable the sensors to report a shift in observed image pattern in real time. By clustering sensors with different orientation together and pooling information on image shifts from all sensors, 6D movements of the helmet can be derived by solving a set of non-linear equations. The salient features of the Self-Tracker are that it uses an inside-out tracking paradigm, and it does not use beacons, and hence does not have the line-of-sight constraint. However, the Self-Tracker might be confused by moving objects in the room. For example, if some of the sensors in the cluster happen to see a person walking by, the cluster might report that the user is moving even though he is stationary. Finally, the amount of data communication between the Self-Tracker and the host computer is much less than if 2D CCD cameras are used.

The full Self-Tracker system has never been built, so the performance of one cannot be empirically measured. Gary Bishop reported [Bis87] that the major problem with the Self-Tracker is likely to be the size, weight, and fragility of the sensor cluster. The helmet may be unwieldy with the mounting of chips, lenses, housing, and wiring.

John Tanner [TM84] constructed an analog VLSI circuit which reports the 2D motion of the observed image pattern. This chip contains an integrated photosensor array and has closely-coupled custom circuits to perform position computation and data extraction. The sensor detects 2D motion by measuring the change of local image intensity. Several such photosensors looking at different directions provide information to recover 6D position and orientation of the observer's head. Again, since a complete system has not yet been built and put into use, the speed and accuracy of such a system remains to be proven.

## 2.2 The Proposed 6D Tracking System

From the above survey, it should be obvious that the requirements outlined in Section 1.3 are hard to satisfy by a single tracking mechanism. However, of all the tracking mechanisms, optical methods seem to possess the most potential. Optical tracking is appealing because it is relatively insensitive to environmentally induced distortion, has a large working volume, and can be made fast and accurate. In the following, we will describe our design adopting the optical tracking paradigm.

### 2.2.1 Inside-out vs. Outside-in Tracking

From Section 2.1.4.1, we learned that most commercially available optical tracking systems place the sensors, which are heavy and bulky, at fixed locations. These sensors are used to observe light emitted from the beacons attached to the helmet which the user wears. Such schemes are termed *outside-in* tracking. Natural environment trackers introduce two novel concepts: no artificial beacons are needed, and an *inside-out* tracking configuration is used. In an inside-out configuration, sensors are placed on the helmet, which is in constant motion. The sensors register the position of beacons which are fixed in the environment. This inside-out tracking scheme was first proposed by Vernon Chi of the UNC MSL. It was suggested that a tracking device can be designed in three dimensions using a concept similar to that of the Optical Mouse [Lyo85]—with sensors directed at a room environment instead of a specially marked surface. The idea was first implemented by Gary Bishop in the Self-Tracker.

The outside-in tracking method, although intuitively simple and appealing, places a

much higher demand on the camera resolution. To compare the required resolution in an outside-in and an inside-out scheme, consider a room 4 meters wide on each side. In an outside-in tracking scheme, in order to cover a large working volume without using a lot of cameras, each camera must have a wide field of view. For example, if we use only two cameras, then each camera must cover at least 4 $m^2$ at a distance of 4 meters away. Suppose several light sources are affixed on antennae mounted on the helmet which the user wears. The base line separation of the light sources is 0.5m. A 0.1° rotation by the user moves the light sources by less than 1mm. In order to detect this 1 millimeter movement at 4 meters, the camera needs at least a 1 part in 4000 (1mm/4m) resolution. This resolution is difficult to achieve by either CCD cameras or lateral effect photodiodes.

In an inside-out tracking scheme, many light sources, or beacons, are used, hence, a camera needs to cover only a small field of view. In a small viewing field, a slight rotational movement induces a large shift of the beacon image on the photodetector surface. Hence to detect the same 0.1° rotation, a much less resolution is needed. For example, if a camera uses a 50mm lens and a recording surface of 1 $cm^2$, then the field of view of the camera at 4 meters away is only 0.8x0.8 $m^2$. A camera with a 1 part in 800 resolution can detect a 1mm movement easily.

### 2.2.2 System Overview

The system we propose to develop is based on the inside-out tracking configuration. We attempt to combine the salient features from several optical trackers surveyed above, namely the inside-out tracking paradigm and the use of beacons, in our system. The goal is to provide a practical and flexible solution to the 6D tracking problem. The system aims to provide reliable performance over a large working volume and should be suitable for use as an unconstrained 6D input device. However, one should note that we do not claim this system to be the ultimate solution to the 6D tracking problem. For one thing, our system still relies on beacons with known positions for a sensor to register its position in space. The placement of beacons does restrict the possible arrangement of the working environment. Further, the line-of-sight constraint still applies.

In this system, all beacons will be placed on the ceiling. This configuration limits the user's movement. For example, it is not possible for the user to tilt his/her head forward 90°, as the camera would point at the wall. But by keeping the beacons on the ceiling, the working environment is not rigidly fixed, and the line-of-sight constraint can be more easily satisfied.

Our proposed system uses three cameras with lateral effect photodiodes as recording surfaces. We use infrared LEDs as beacons. Beacons are placed in a regular pattern on the

ceiling. The user wears a helmet which mounts three cameras. Three LEDs selected by the system are flashed when they are inside the cameras' fields of view. The 6D position of the camera assembly is recovered by correlating the observed image positions and the known 3D positions of the beacons.

Below we introduce the components used in the system, namely the lateral effect photodiodes as sensors and the infrared LEDs as beacons.

#### 2.2.2.1 Lateral Effect Photodiodes

The advent of position-sensitive photodiodes [Edl57] makes possible the design of high-performance real-time tracking systems. A lateral effect photodiode features a large photo-sensitive surface, usually square in shape, on which the $x$, $y$ location of the centroid of an illuminance can be estimated. This is done by measuring the ratio of the photo-currents flowing from the electrodes on each of the four sides of the photosensitive surface.

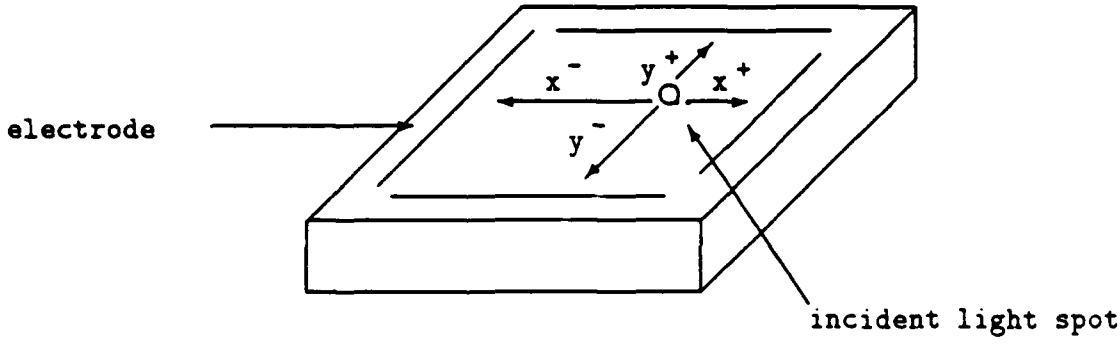



Figure 2.1: Lateral effect photodiode

Specifically, when a light ray strikes the surface of a photodiode, photoelectric charges are generated. The magnitude of the charges is proportional to the energy carried by the incoming ray. The electric charges flow through the resistive surface and are collected by the electrodes on the four sides of the photodiode. Since the resistivity of the surface is uniform, the photocurrent collected by an electrode is a function of the distance between the incident light position and the electrode. It is possible to calculate the $x$, $y$ position of the incident light beam on the photodiode surface using the following formulas:

$$x = \frac{x^+ - x^-}{x^+ + x^-}$$
$$y = \frac{y^+ - y^-}{y^+ + y^-}. \qquad (2.1)$$

where $x^+, x^-, y^+$ and $y^-$ denote the photocurrents collected by the electrodes at the four edges.

Using lateral-effect photodiodes as the recording surfaces has the following advantages over conventional CCD arrays: lateral-effect photodiodes provide faster response and higher positional resolution, have no dead-zone over the recording surfaces, and provide an accurate positional reading even if the light spot is out of focus (or blurred). Finally, there is no need to calculate the centroid of the recorded light spot. However, photodiode surfaces usually have a spatially non-linear characteristic, and thermal noise and dark current can seriously degrade the accuracy in the positional reading. Detailed analysis of the performance of photodiodes will be presented in later chapters.

#### 2.2.2.2 Beacons

The beacons used in this system must satisfy several requirements. First, they must be easily distinguishable from the environment (by blinking or high luminance), and emit light in the infrared—so as not to distract the user. Second, the beacons should have a wide emission angle. Third, the beacons must be inexpensive since many are used. Beacons chosen for this implementation are infrared light emitting diodes (LEDs). Properties of the LEDs are discussed in Section 4.2.

# Chapter 3

# An Optical Inside-out Tracker

In this chapter, we discuss the design criteria of the new 6D optical tracker. The requirements of accuracy, speed, adequate working volume, light weight and small size are addressed. In an inside-out tracking configuration with cameras and beacons, a mechanism for computing 6D position of the camera assembly from the observed position of beacons is needed. Various algorithms to accomplish this computation are surveyed. The particular tracking algorithm which we adopt in our system is then discussed in more detail. The system's error is studied, and several equations are formulated to predict the system's error analytically. Error predicted by the analytic formulation is compared with the results obtained using a simulation program. Since beacons are used and accuracy of their placement affects that of the system, a self-calibration method which automatically calibrates the position of beacons is proposed.

## 3.1 Design Criteria

The criteria used to judge the performance of a 6D tracking system were outlined in Section 1.3. Before we start the design and construction of a prototype system, the desired performance of the prototype—evaluated against the criteria outlined in Section 1.3—has to be stated. In the following subsections, we address the issues of accuracy, speed, working volume, weight and size of the tracking system.

### 3.1.1 Accuracy and Speed

An obvious requirement is for the 6D tracker to report the user's head position that accurately agrees with the user's true head position. Furthermore, the system should be

responsive, i.e, should be sensitive to small changes in the user's head position, and report such changes in real time (small latency), and as often as possible (a high update rate). This leads us to define the following three terms:

**Accuracy** *The agreement between the true head position*[1] *and the calculated head position*

**Resolution** *The minimal amount of head motion that can be registered by the tracker*

**Responsiveness** *The number of updates per second and the lack of latency*

As discussed later in this chapter, the resolution of the system depends on the resolution of the photodiode, whereas the accuracy of the system also depends on how accurately the beacons are placed. The resolution of the prototype system aims at being able to report 5mm in translational movement and 0.4 degree in rotational movement. The resolution corresponds roughly to a shift of 2 pixels in the displayed image—assuming that the user is looking at a target 1 meter away with a 90 degree field of view, and an image resolution of 512 by 512 pixels. Also, by a novel camera self-calibration process, we can reduce the measurement error in beacon positions, thus improving the accuracy of the system. We can achieve about 0.8 degree of rotational accuracy and 1cm of positional accuracy with this camera self-calibration process (Section 3.4.2).

Our experience indicates that in order to maintain an illusion of smooth motion in a graphic animation sequence, an update rate of at least 10 to 20 frames per second is required. Also, the latency problem, which is an important issue in all interactive applications, must also be addressed. For this prototype, the target update rate is 30 Hz. the latency was targeted to be not longer than one update period, or 33 milliseconds.

### 3.1.2 Working Volume

Limited working volume is a major drawback of the Polhemus tracker currently used in our head-mounted display system. The Polhemus has a working volume which is spherical in shape and about 0.7 meter radius. The working volume can be extended to a 1.5 meter radius sphere with reduced accuracy. This working volume is adequate for applications such as simulating the combat situation display in a cockpit where the pilot has limited movement. However, such a range is too restrictive for any interesting maneuvering or walk-around in, say, a room-sized environment. With the new tracker, a much larger working range can be achieved. We aim at a room of size $(3m)^3$ for this prototype.

[1] Here the term implies 6D position

### 3.1.3 Size, Weight and Moment of Inertia

Gary Bishop stated in his thesis [Bis84] that the size of the tracker should not exceed that of a grapefruit, and the preferred size is no larger than an orange. Lacking any theoretical study, we will use his observation as our size requirement. The camera is 4x4x6.5 $cm^3$ in size and weights 138 grams. The lens is 5cm in diameter and 5cm in length, and weights 181 grams. The current prototype with multiple cameras and lenses weight nearly 1 kg. Hence, our prototype fails short in satisfying the size and weight requirements. Besides, a 1 kg camera assembly on the helmet considerably raises the center of gravity of the head-mounted display, and makes it harder for the user to tilt his/her head forward and back and from side to side. Finally, in order for the user to rotate his/her head freely, we also have to worry about the moment of inertia, not just the weight of the head-mounted display system. The current EyePhone head-mounted system [VPL89] we use has a 2 kilograms counter balance at the back of the user's head. It should be possible to distribute the weight of the tracker and the video display screens to reduce the the moment of inertia.

In this current prototype, we do not intend to completely solve the above problems. As the technology advances, photodiodes with better resolution can reduce the size and weight of both the cameras and lenses. Also, the full-scale version of the prototype will likely use holographic lenses which are compact and light-weight. This would reduce the size of the assembly, and cut down the weight of the whole system significantly. Then, instead of mounting the tracker on top of the helmet, we could mount the tracker at the back of the user's head to act as the counter balance to the LCD TVs which are in front of the user's eyes. So for the full scale system, the center of gravity will be lower and the moment of inertia should not exceed that of the current EyePhone system we use.

## 3.2 Algorithms for Inferring 6D Position

For an outside-in tracking scheme, the 3D position of a beacon is routinely computed using triangulation where at least two cameras are used. The position of the beacon's image on the image plane of a camera determines a line along which the beacon must lie. The 3D position of the light source is then located by intersecting at least two such projection lines in space.

Position recovery is different in an inside-out configuration. Here, cameras are mounted on top of the helmet which is in constant motion, so their position and orientation are unknown. Besides, cameras are used to observe different beacons so the stereopsis principle doesn't apply. Instead, the 6D position of the camera assembly is computed by correlating several observed image positions (3 to 6) and the known 3D positions of the beacons through

a projection equation.

Given the position $B_i = [B_{i1}, B_{i2}, B_{i3}, 1]$ of several beacons in homogeneous world coordinates, and the image locations $b_i = [b_{i1}, b_{i2}, 1]$ of the beacons under a perspective projection, the goal in both outside-in and inside-out tracking configuration is to find a transformation matrix $M_{4x3}$—which represents the projection process—such that $M$ satisfies the following constraint:

$$[B_{i1}, B_{i2}, B_{i3}, 1]M = [b_{i1}, b_{i2}, 1]. \tag{3.1}$$

In an inside-out tracking configuration, several algorithms have been proposed to solve this problem. Since there are twelve unknowns in $M$, and each observed beacon provides two independent equations based on Equation 3.1, at least six beacons are needed to solve a set of twelve linear equations for estimating $M$. This method is described in [PP86]. The method is computationally expensive, for it requires the inversion of a 12 by 12 matrix. Further, the coefficient matrix may be ill-conditioned, depending on the configuration of beacons in space. Experience from implementing this algorithm shows that to avoid degeneration in the computation, one has to make sure that the six observed beacons do not lie in the same plane (e.g., ceiling).

Since a lateral effect photodiode is used as the recording surface, only one beacon can be turned on inside the field of view of the camera at any time. Hence, beacons are flashed sequentially. During the period of time between the flashing of the first beacon and the flashing of the sixth beacon, the user may have moved. This movement will introduce an error in the recovered 6D position of the camera. Hence, one important criterion in judging the feasibility of a tracking algorithm is the number of beacons the algorithm needs to observe to compute the 6D position of the camera. Below we introduce several algorithms which use fewer beacons to solve for the transformation matrix $M$.

Ganapathy introduced a variation of the above method which needs only four coplanar beacons [Gan84]. He observed that the 3x3 submatrix at the upper part of the transformation matrix $M_{4x3}$ represents the rotational motion of the observer. If this 3x3 rotational submatrix $R$ is expressed as

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix},$$

then, this matrix must be a "proper orthonormal matrix" which satisfies the condition $R^{-1} = R^T$. This orthonormal property together with the property that the determinant of $R$ is unity (no scaling) imply that

$$a^2 + b^2 + c^2 = d^2 + e^2 + f^2 = g^2 + h^2 + i^2 = 1$$
$$ad + be + cf = dg + eh + fi = ag + bh + ci = 0$$

$$
\begin{aligned}
i = ae - bd \quad & g = bf - ec \quad h = cd - af \\
a = ei - fh \quad & b = fg - di \quad c = dh - eg \\
d = hc - bi \quad & e = ai - gc \quad f = bg - ah.
\end{aligned} \tag{3.2}
$$

By imposing the above constraints, he showed that only four beacons are needed in determining uniquely the 6D position of the camera—provided that the four beacons are on the same plane. This method provides a simple analytic solution which is computationally efficient. Unfortunately from our simulation, it was found out that this method contains a lot of degenerate cases, and does not perform well when input data are noisy.

Another method proposed by Earl Church [Chu45] was first used in aerial photogrammetry. Church's method determines the position of a camera from an aerial photograph by locating three known landmarks in the photograph. Since only three beacons are used, it can be shown there is no linear solution [PP86], and an iterative method has to be employed to derive the 6D position of the camera.

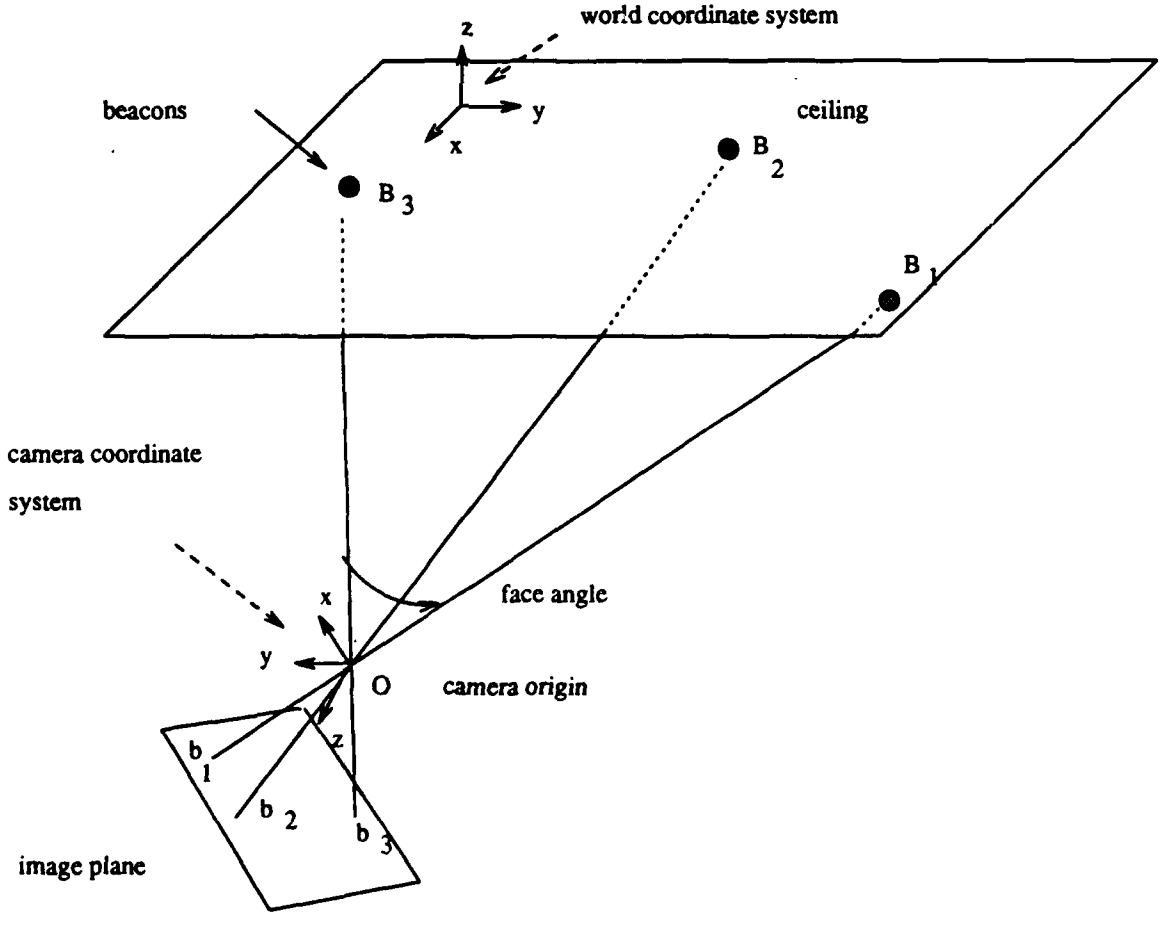



Figure 3.1: Church's algorithm

Figure 3.1 shows how Church's method may be applied in our tracking environment. Two coordinate systems are defined here. The world coordinate system is defined on the fixed room environment, while the camera coordinate system is defined by using the nodal point of the camera lens as origin and the x-y plane taken as parallel to the image plane of the camera. In Church's method, three beacons are observed by the camera. Any two of the three observed beacons form a *face angle* with the camera origin. These face angles can be computed using either the 2D image position or the 3D space position of the beacons. Denote the position of the three beacons in space as $B_1, B_2$ and $B_3$, and the positions of their images as $b_1, b_2$ and $b_3$ as in Figure 3.1, then we have

$$\begin{aligned} k_1 &= \cos(b_1 O b_2) \\ k_2 &= \cos(b_2 O b_3) \\ k_3 &= \cos(b_3 O b_1) \end{aligned}$$

$$\begin{aligned} K_1 &= \cos(B_1 O B_2) \\ K_2 &= \cos(B_2 O B_3) \\ K_3 &= \cos(B_3 O B_1) \end{aligned}$$

Church's solution is based on the condition that the face angle subtended by any two beacons in space equals the face angle subtended at their corresponding image projections. This condition states that

$$\begin{aligned} k_1 &= K_1 \\ k_2 &= K_2 \\ k_3 &= K_3 \end{aligned} \tag{3.3}$$

The face angles formed by the camera origin and the images can be calculated directly in the camera coordinate system. Further, the world coordinates of the three observed beacons are known. Hence, the only unknown is the location of the camera origin in the world coordinate system. Once the position of the camera origin is known, the orientation of the camera can be easily figured out as discussed later. In order to determine the current position of the camera origin, Church's method starts by guessing randomly a value for the camera origin in the world coordinate system. With this hypothesized camera origin position, we can form three face angles in the world coordinate system. These face angles will in general not match those computed from the image unless the hypothesized camera position is correct. An iterative search scheme is employed to estimate the correct 6D camera location by minimizing the difference between the corresponding face angles.

The iterative method works as follows: we are given the positions of three beacons $B_i = (X_i, Y_i, Z_i)$, and their images on the photodiode surface $b_i = (x_i, y_i, f)$, where $f$ is the

focal length of the camera lens and $1 \leq i \geq 3$. We then compute the face angles in the camera coordinate system as

$$\begin{aligned} k_i &= l_i l_j + m_i m_j + n_i n_j, \quad \text{where } j = (i+1) \bmod 3, \\ \text{where } l_i &= \frac{x_i}{\sqrt{x_i^2 + y_i^2 + f^2}}, \\ m_i &= \frac{y_i}{\sqrt{x_i^2 + y_i^2 + f^2}}, \\ n_i &= \frac{z_i}{\sqrt{x_i^2 + y_i^2 + f^2}}. \end{aligned}$$

If the hypothesized camera origin position is $H = (X_h, Y_h, Z_h)$, then based on $H$, we calculate face angles in the world coordinate system as

$$\begin{aligned} K_i &= L_i L_j + M_i M_j + N_i N_j, \\ \text{where } L_i &= \frac{X_h - X_i}{D_i}, \\ M_i &= \frac{Y_h - Y_i}{D_i}, \\ N_i &= \frac{Z_h - Z_i}{D_i}, \\ \text{and } D_i &= \sqrt{(X_h - X_i)^2 + (Y_h - Y_i)^2 + (Z_h - Z_i)^2}. \end{aligned}$$

These two sets of face angles will in general not match. Taking the partial derivative of the expression $(K_i - k_i)^2$ with respect to $X_h$, $Y_h$, and $Z_h$, we obtain an adjustment $(\Delta X_h, \Delta Y_h, \Delta Z_h)$ to $H$. The adjustment equations are:

$$\begin{aligned} \alpha_i \Delta X_h &+ \beta_i \Delta Y_h + \gamma_i \Delta Z_h = \Delta K_i = k_i - K_i \qquad (3.4) \\ \text{where } \alpha_i &= L_i I_i + L_j J_i \\ \beta_i &= M_i I_i + M_j J_i \\ \gamma_i &= N_i I_i + N_j J_i \end{aligned}$$

$I_i$ and $J_i$ are auxiliary values used in the formation of the coefficients $\alpha_i, \beta_i, \gamma_i$ in the adjustment equations.

$$I_i = \frac{k_i}{D_i} - \frac{1}{D_j}$$
$$J_i = \frac{k_i}{D_j} - \frac{1}{D_i}$$

The adjustment $(\Delta X_h, \Delta Y_h, \Delta Z_h)$ derived from Equation 3.5 is then added back to the hypothesized position $H$. Iteration stops when the adjustment amount is less than a preset threshold. In our system, we start the iteration using the user's latest head position as the initial guess of $H$, and iterate until $\Delta X_h^2 + \Delta Y_h^2 + \Delta Z_h^2 \leq 1^{-10}$ mm$^2$, or a maximum iteration

count is exceeded. The maximum iteration count is set to 10 in our implementation. Our experience shows in our application the Church's method usually converges in just one or two rounds, since we have a very good initial guess which is close to the true solution.

Church's method, due to the iterative procedure to find a final solution, doesn't guarantee convergence. It is possible for the method to converge to a wrong position, or not converge at all. From running the simulation , we learned that the method fails to converge when the tetrahedron formed by the three beacons (or the three beacon images) and the camera origin is ill-shaped (e.g., one face angle is too small or the three beacons are too close together). In order to make sure the method works, we need to select three beacons which are sufficiently far apart from one another.

After the current camera position is determined, orientation can be readily estimated. Suppose the 3x3 rotation matrix is $R$ with 9 unknowns, then from

$$(l_i, m_i, n_i)R = (L_i, M_i, N_i) \tag{3.5}$$

we solve a set of 9 linear equations to estimate $R$. Appendix B lists the program which implements the Church's method.

Fischler [FB81] solved Church's formulation analytically by applying geometric constraints to the tetrahedron formed in world coordinates using the three beacons and the camera origin as vertices. Referring to Figure 3.1 again, $O$ is the origin of the camera coordinate system and $B_1, B_2$, and $B_3$ are the beacon locations. Fischler's method first computes the lengths $\overline{OB_1}, \overline{OB_2}$, and $\overline{OB_3}$ based on the (known) lengths $\overline{B_1B_2}, \overline{B_2B_3}$, and $\overline{B_3B_1}$, and the (known) face angles $\angle b_1Ob_2$, $\angle b_2Ob_3$, and $\angle b_3Ob_1$ by solving the system of equations:

$$\begin{aligned}
|B_1B_2|^2 &= |OB_1|^2 + |OB_2|^2 - 2|OB_1||OB_2|\cos(\angle b_1Ob_2) \\
|B_2B_3|^2 &= |OB_2|^2 + |OB_3|^2 - 2|OB_2||OB_3|\cos(\angle b_2Ob_3) \\
|B_3B_1|^2 &= |OB_3|^2 + |OB_1|^2 - 2|OB_3||OB_1|\cos(\angle b_3Ob_1)
\end{aligned} \tag{3.6}$$

Once the sides of the tetrahedron are known, the position of the origin $O$ can be computed. The drawback of this method is that in order to solve Equation 3.6, we have to find the roots of a 4th order polynomial. This 4th order polynomial can be solved analytically [Edh30]. From our simulation, solving the 4th order equation in Fischler's method takes twice as long as the iterative steps in Church's method. If the input data are noisy, Fischler's method sometimes fails to compute any real solution.

Church's method was selected for our system because it requires the least number of beacons among all the methods we have surveyed. Church's method is quite robust even in the presence of errors (discussed in Section 3.3.6). Although Church's method requires iterations to converge to the correct solution, the convergence can be expedited if the initial

guess is close to the true location of the camera. In our application, we can always use the latest head position as an educated initial guess. Also, we can build the tracker in such a way to make sure convergence will not be a problem (Section 3.3.3).

## 3.3 System Design and Error Analysis

In this section, we discuss the type of errors which occur in the system and how they affect the accuracy in position reading. We also present methods to minimize these errors.

### 3.3.1 Error Sources

To quantify error in the system, we assume that Church's method, when given perfect input data, converges to the correct 6D position. That is, we supply as input to Church's method one set of face angles in the world coordinate system and another set in the camera coordinate system—all without any measurement error. If there is no error in the data acquisition process, Church's method will converge after a certain number of iterations and return with the correct camera origin in the world coordinate system.

Since Church's method does not introduce any significant computational error, any error in the computed 6D location will be due to the input errors propagating to the output [PW83]. As mentioned before in Church's method, the 6D location of the camera is computed by matching two sets of face angles. Hence, errors in Church's method come from the measurement errors in the two sets of face angles used as input to the algorithm.

The measurement error in the face angles formed by the camera origin and the projection of the beacons in the image plane is mainly due to the limited resolution of the photodiode and the nonlinear characteristics of the photodiode surface. Recall from Section 2.2.2.1 that the photodiode is an analog device which provides continuous reading of $x, y$ position across its surface. The resolution of the photodiode is limited by several factors, such as the material used and thermal noise inherent in all devices. Increasing the incident light energy increases the signal-to-noise ratio and hence gives a better resolution. Nonlinearity of the photodiode surface can be compensated by carefully calibrating the photodiode surface (discussed in 4.1.2).

The measurement error in the face angles formed by the camera origin and the beacons in space is mainly due to the positional uncertainty of beacons. Since a working system might contain hundreds or even thousands of beacons distributed over the ceiling, the problem of placing them accurately and measuring their position reliably are nontrivial. These problems

are addressed in Section 3.4.

Finally, the user's movement also induces error. Since beacons are flashed sequentially, the user's movement during the period from flashing the first beacon to flashing the third one will induce a certain amount of image shift on the photodiode surface. This problem is addressed in Section 3.5.

### 3.3.2 Errors Due to the Limited Resolution of Photodiodes

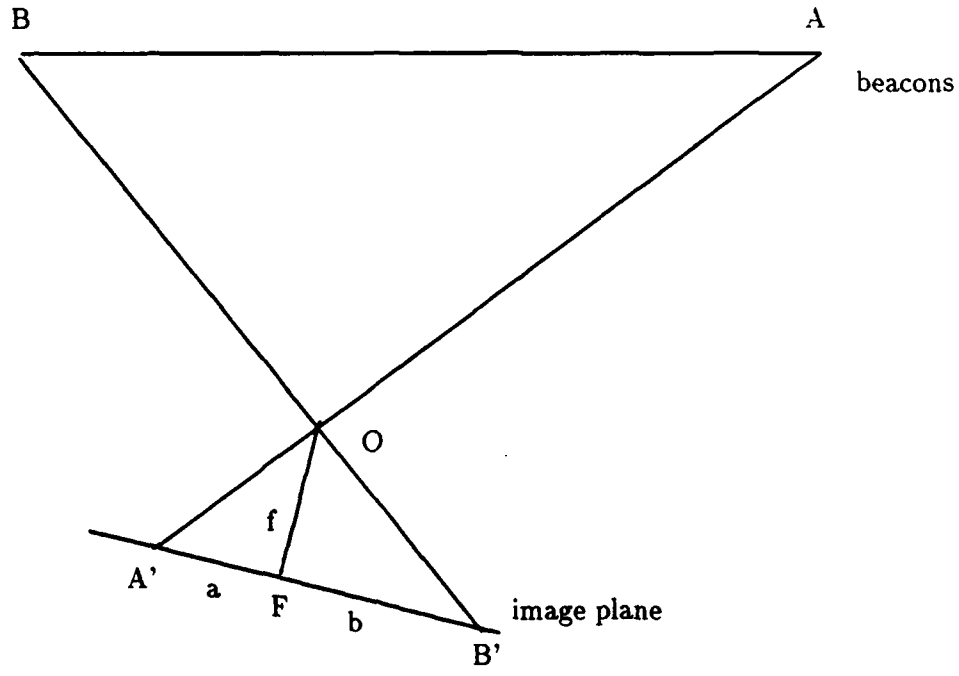



Figure 3.2: System error analysis

An analytic expression of the errors which are due to the limited resolution of the photodiode used is derived below. Figure 3.2 shows a plane which intersects the two beacons $A$ and $B$, and the nodal point of the camera $O$. The two beacons form image $A'$ and $B'$ on the image plane of the camera respectively. It can be seen that[2]

$$\begin{aligned} \theta &= \angle A'OB' = \angle A'OF + \angle B'OF \\ &= \tan^{-1}\frac{a}{f} + \tan^{-1}\frac{b}{f}, \end{aligned}$$

[2]The formulas derived in this section are for a restricted case where errors are only two-dimensional. That is, we consider only error in the plane formed by camera origin $O$, and beacons $A$ and $B$. Formulas for general cases are listed in appendix A

where $f$ is the focal length of the camera lens. From [PW83], we have

$$\begin{aligned}\varepsilon_\theta &= \frac{\partial\theta}{\partial a}\varepsilon_a + \frac{\partial\theta}{\partial b}\varepsilon_b + \frac{\partial\theta}{\partial f}\varepsilon_f \\ &= \frac{f}{a^2+f^2}\varepsilon_a + \frac{f}{b^2+f^2}\varepsilon_b + (\frac{a}{a^2+f^2} + \frac{b}{b^2+f^2})\varepsilon_f,\end{aligned}$$

where $\varepsilon_a, \varepsilon_b$ and $\varepsilon_f$ represent the absolute error bounds on the measurements of $a, b$, and $f$.

Assuming that a beacon's image location can be measured to $D/r$, where $D^2$ ($\text{mm}^2$) is the size of the photodiode, and the resolution of the photodiode is 1 part in $r$, then the maximum error in the measurement of $a$ and $b$ is half of $D/r$, or $\varepsilon_a = \varepsilon_b = D/2r$. Error in measuring the focal length is a constant, independent of different readings from the photodiode. We ignore this error term for now. Thus we have:

$$\varepsilon_\theta = (\frac{1}{a^2+f^2} + \frac{1}{b^2+f^2})\frac{fD}{2r}. \tag{3.7}$$

Equation 3.7 is the governing equation on the design of our tracking system. Equation 3.7 states that in order to minimize the measurement error of the face angles in the camera coordinate system, lenses with long focal length should be used, the separation between image points should be as far apart as possible, and the resolution of the photodiode should be as high as possible.

We found that these requirements impose conflicting demands on the system design. For example, lenses with long focal lengths have a relatively small field of view. In order to locate at least three beacons inside this small viewing field, beacons must be placed tightly together, which implies a lot of them are needed. How to place beacons and how to measure their positions accurately becomes difficult. Furthermore, closely-placed beacons introduce larger errors in the face angle measurements in the world coordinate system. Closely-placed beacons also make it harder to find three beacons whose images are widely separated in the image plane.

### 3.3.3 Single View vs. Multiple Views

In order to obtain a wide image separation of the projected beacon locations and to use a lens with long focal length, it was suggested by Vernon Chi that more than one camera should be used. For example, we can use lenses with long focal lengths for all cameras so that each camera covers only a small field of view to obtain high accuracy in the 2D position reading. These cameras can be oriented with mutually large angular separations to achieve wide image separations and a wide field of view of the combined assembly. Small translational movements are readily detected because of the small field of

view of the individual cameras, while widely separated cameras make detection of rotational movements easy. This idea is appealing because all mutually conflicting optical design criteria can be optimized independently instead of interdependently.

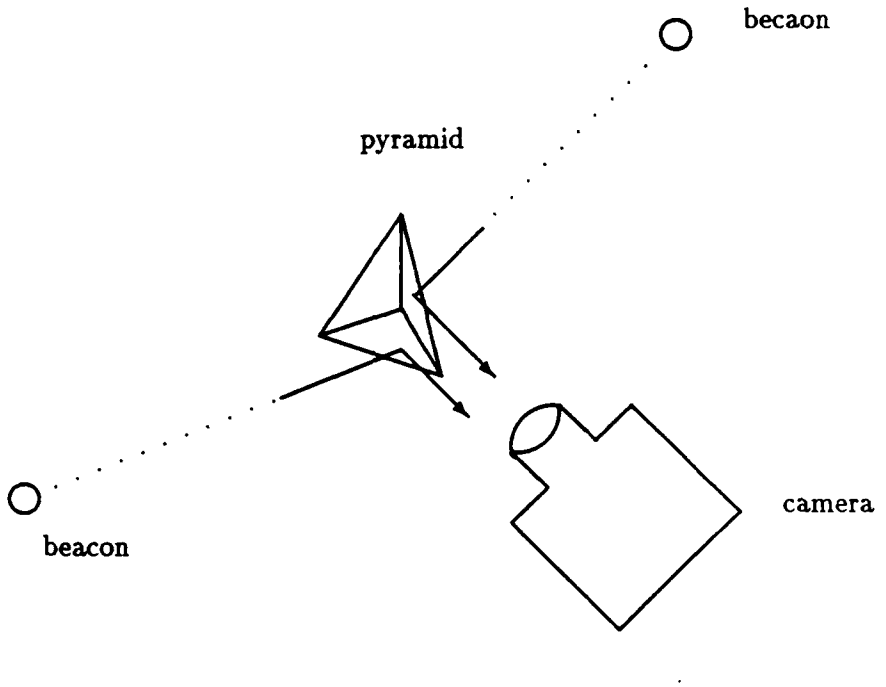



Figure 3.3: Realization of the multiple-view concept with pyramid

Other alternatives for generating multiple views are available. With a single camera, a pyramid made of highly reflective mirrors can be placed in front of the camera as shown in Figure 3.3. The pyramid reflects light from several different directions into the lens aperture. The difficulty with this design is that since each view takes only part of the lens, the image might shift if it is out of focus. Figure 3.4 shows the difference between a beacon that uses the whole camera lens and one that uses only part of the lens. The dashed lines show a view using only half of the lens aperture. Since the photodiode registers the centroid of the light spot, it can be seen that blurring (out-of-focus) introduces image shift in this case.

A similar concept is to place a rotating mirror in front of the camera and this rotating mirror reflects light from all directions. Rotation of the mirror must be carefully controlled and synchronized with the tracking system, which implies expensive hardware might be needed.

Another realization of multiple views is to stack several levels of partially-silvered mirrors in front of the camera as shown in Figure 3.5. Each view then uses the whole lens aperture without the image shift problem present in pyramid design. Further, there are no

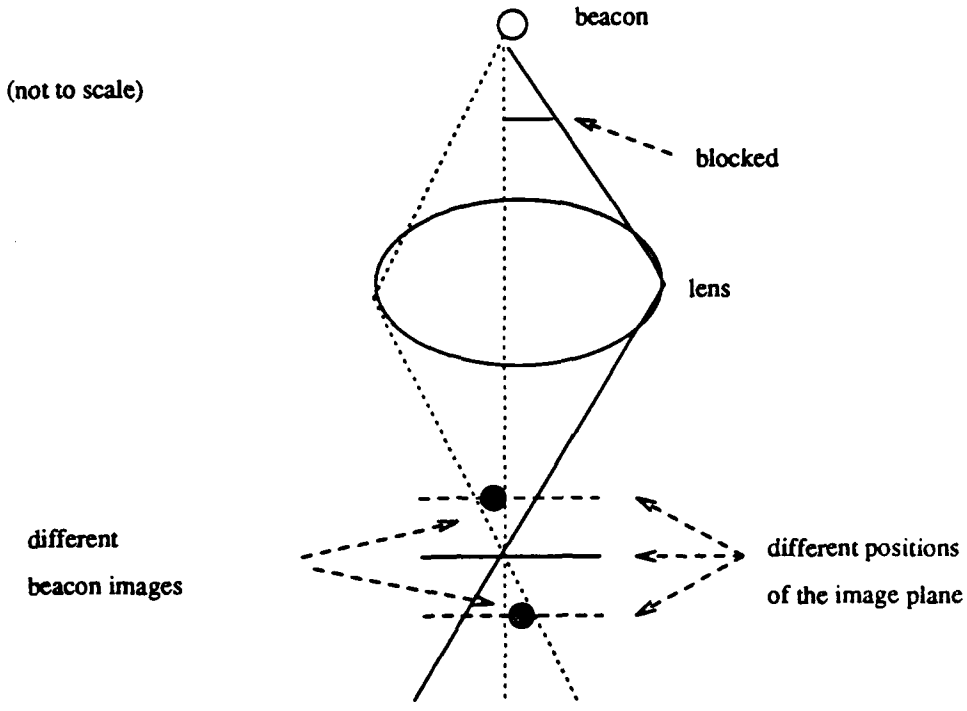



Figure 3.4: Image shift due to blurring

movable components as in the rotating mirror design. A $1/n$ silvered mirror causes $1/n$ of the light striking its surface to be reflected, while (n-1)/n of the light will pass through the mirror. From Figure 3.5, 1/3 of the light from the beacon on the left side of the camera will be seen by the len aperture, and (1/2)*(2/3) = 1/3 of the light from each of the other two beacons will be seen.

Holographic optics may prove to be an effective solution. According to [Tei88], a holographic lens can be made to observe in multiple (e.g., four) directions. The size of an F/0.5 50mm lens is only that of a silver dollar. This method is worth pursuing because it also solves the weight problem.

The method we adopt in this prototype is to use three cameras instead of one. These three cameras are set to an angular separation of 45° from the vertical direction, and are azimuthally separated equally from one another. Each camera uses a 50 mm focal length lens to observe a narrow angle of view ($\sim 12°$).

### 3.3.4 Estimating the 6D Position of the Camera Assembly

If multiple cameras are used, the 6D position of the whole camera assembly has to

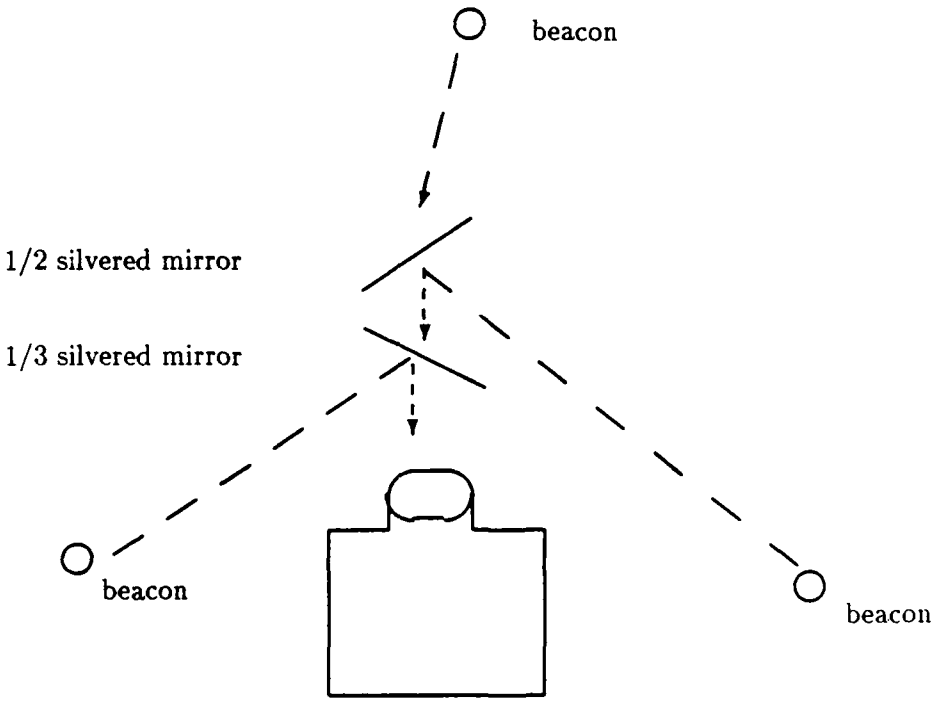



Figure 3.5: Realization of the multiple-view concept with partially-silvered mirrors

be estimated. This is in general a more difficult problem than if a single camera is used. With a single camera, light from different beacons always passes through the same nodal point and focuses on a single image plane. The nodal point of the lens can then naturally be chosen as the origin of the camera coordinate system in Church's method. If multiple cameras are used and each camera has its own nodal point, then more than one nodal point may exist in the system as the nodal points cannot normally coincide with each other. In order to understand the difficulty caused by the disjoint camera nodal points, we show two cameras with two distinct nodal points $O_a$ and $O_b$ looking at beacons $A$ and $B$ respectively as in Figure 3.6. These two cameras are separated by an angle and the two nodal points are separated by a fixed distance. By carefully measuring the camera positions, the vector $\overrightarrow{O_bO_a}$ in the camera coordinate system can be derived. Hence, the face angle $\angle aO_ib$ in the image coordinate system can be calculated by translating one of the nodal points (e.g. $O_b$) to merge with the other ($O_a$). In order to derive the corresponding face angle $\angle AO_aB$ in space, we need to translate $B$ by the same vector $\overrightarrow{O_bO_a}$. Thus we need to know the expression of $\overrightarrow{O_bO_a}$ in world coordinates system. However, the direction of $\overrightarrow{O_bO_a}$ in the world coordinate depends on the current head position which is unknown. Hence, Church's method does not apply when there are multiple cameras with distinct nodal points.

This problem has been attacked independently by John Halton [Hal89] and by the

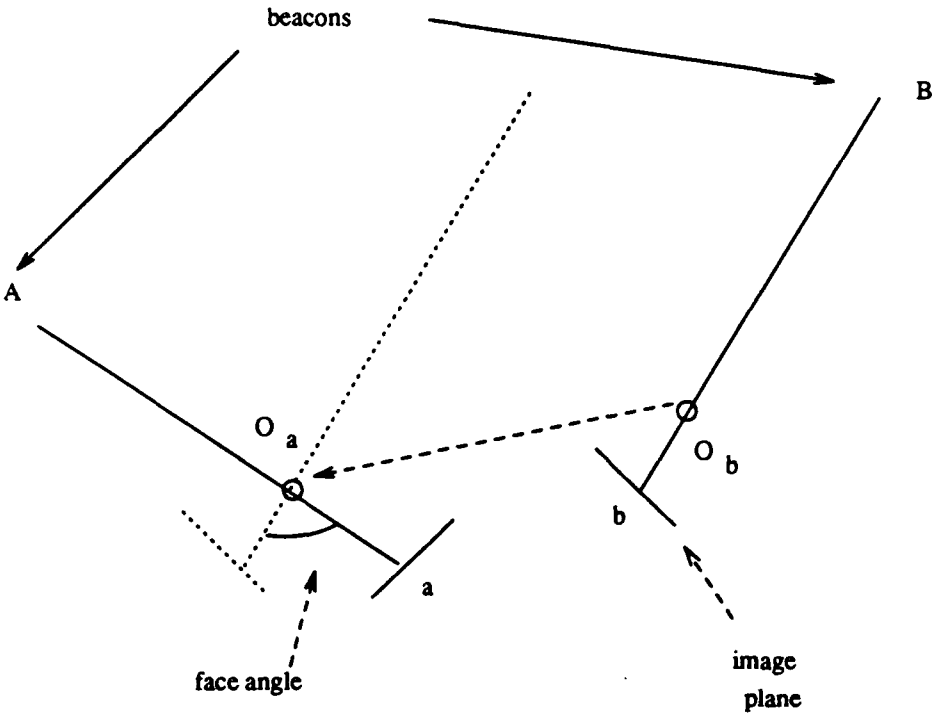



Figure 3.6: Problem with Church's method in multiple views

author. Two different solutions were formulated. Here we discuss my solution, which generalizes Church's method for multiple camera configurations. In the generalized Church's method, the origin of the camera coordinate system $O$ is arbitrarily chosen. Referring to Figure 3.7, $\overrightarrow{OO_a}$ and $\overrightarrow{OO_b}$ can be measured in the camera coordinate system, as can $\overrightarrow{O_a a}$ and $\overrightarrow{O_b b}$. We can thus derive the expressions of the angles $\angle OO_a a$ and $\angle OO_b b$

$$\angle OO_a a = \cos^{-1}\left(\frac{\overrightarrow{O_a a} \cdot \overrightarrow{O_a O}}{|O_a a|\ |O_a O|}\right)$$

$$\angle OO_b b = \cos^{-1}\left(\frac{\overrightarrow{O_b b} \cdot \overrightarrow{O_b O}}{|O_b b|\ |O_b O|}\right).$$

Now following Church's method, let's hypothesize the position of the virtual origin $O$ in the world coordinate, then we can compute the face angle $\angle AOB$ in the world coordinate system in terms of the hypothesized position of $O$. The face angle in the camera coordinate system is derived as follows. From the law of cosines, we get

$$|OA|^2 = |O_a O|^2 + |O_a A|^2 - 2\,|O_a A|\,|O_a O| \cos(\angle A O_a O). \tag{3.8}$$

From the above equation, we compute $\overline{O_a A}$, the same for $\overline{O_b B}$. Since we are solving a second order equation here, there might be two real solutions for $\overline{O_a A}$. However, if we choose the

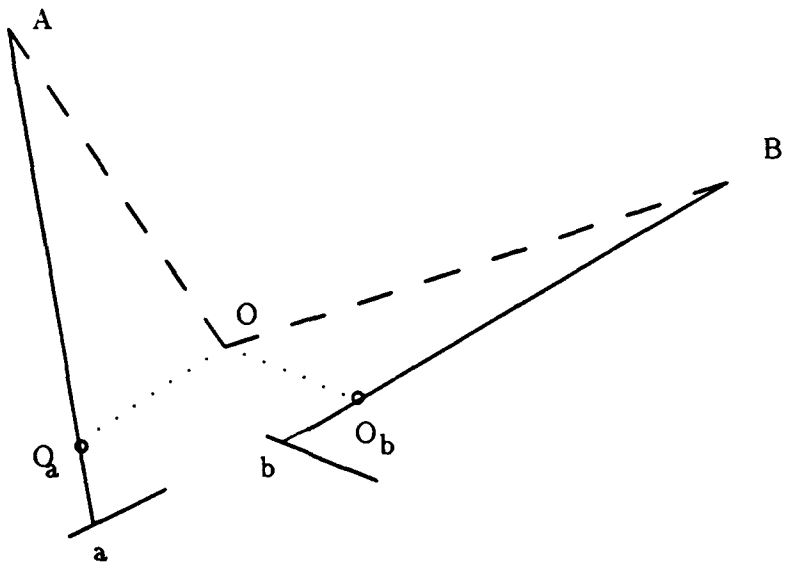



Figure 3.7: Generalized Church's method to suit multiple views

virtual camera origin $O$ in such a way that $\angle OO_aA$ is obtuse, then there can be only one real solution, since $\overrightarrow{OA} = \overrightarrow{OO_a} + \overrightarrow{O_aA}$, and

$$\overrightarrow{O_aA} = -\overrightarrow{O_aa}\,|O_aA| \,/\, |O_aa|. \tag{3.9}$$

Hence, we can derive $\overrightarrow{OA}$ in the camera coordinate system, and similarly for $\overrightarrow{OB}$. Then the face angle $\angle AOB$ in the camera coordinate system can be computed. Using the two sets of face angles as defined above with respect to the virtual camera origin $O$, the original Church's method again becomes applicable.

Compared with the original Church's method, the generalized version does introduce additional computational overhead in calculating the face angles. In the original method, the face angles in the camera coordinate system need to be calculated only once, whereas in the generalized method these angles have to be recalculated for each iteration. However, since $\overline{O_aO}$ and $\overline{O_aa}$ do not change from iteration to iteration, the computation of the face angles at each iteration involves solving $\overline{O_aA}$ using Equation 3.8 and $\overline{OA}$ using Equation 3.9 only. Our simulation shows only a small slow-down ($\sim$ 8%–12%) running the generalized method over the original method.

### 3.3.5 Errors Due to Inaccurate Placement of Beacons

As pointed out in the previous section, the error in computing the face angles in the camera coordinate system is largely due to the limited photodiode resolution, whereas the error in computing the face angles in the world coordinate system is largely due to error in the beacon position.

The number of the beacons used in the system depends on the field of view of the cameras. Lenses with shorter focal length cover a wider field of view. Since only one beacon needs to be observed by each camera if three cameras are used, fewer beacons are needed with shorter focal length lenses. However, to discern the details presented in the wide field of view, a photodiode of a much higher resolution is required. For example, consider two lenses, one with a focal length of 50mm and another 10mm. Assume that the size of the photodiode is 1 $cm^2$. In order to detect a movement of one millimeter at 2 meters away, the first photodiode needs to have a resolution of at least 1 part in 400 while the second one needs 1 part in 2000.

The resolution of commercially available photodiodes is usually in the range of 1 part in 1000 (discussed in Section 4.1.2), hence, it is not feasible to use very a wide field of view, which implies the system may then need to use several hundreds of beacons. It becomes difficult, if not impossible, to place them and measure their positions accurately.

The error in the face angles is computed using Figure 3.2 again. If the coordinates of $A$ and $B$ are $(x_a, y_a)$, and $(x_b, y_b)$, then

$$\angle AOB = \frac{\overrightarrow{OA} \cdot \overrightarrow{OB}}{|OA|\,|OB|} = \frac{x_a x_b + y_a y_b}{\sqrt{x_a^2 + y_a^2}\sqrt{x_b^2 + y_b^2}}$$

Let the maximum error in the positioning of beacons be $\varepsilon_p$, then $\varepsilon_x = \varepsilon_y = \varepsilon_p$. So we have

$$\begin{aligned} \varepsilon_\theta &= \left(\frac{\partial\theta}{\partial x_a} + \frac{\partial\theta}{\partial y_a} + \frac{\partial\theta}{\partial x_b} + \frac{\partial\theta}{\partial y_b}\right)\epsilon_f \\ &= \left(\frac{x_a + y_a}{|OA|^2} + \frac{x_b + y_b}{|OB|^2}\right)\varepsilon_p. \end{aligned} \tag{3.10}$$

We see that this error is directly proportional to that in the placement of beacons. Again large separation between the beacons seen by the camera decreases the error. Large separation can be achieved by using a photodiode with larger surface area, or using multiple views which are widely separated.

### 3.3.6 Computer Simulation Results

To verify the analytical prediction on the error bound and to help determine several important system parameters, we developed a simulation program. This simulation program allows us to selectively modify all the essential design parameters of the system, and study their effects on the performance of the system. The user can specify the environment parameters which include the size of the room, the number of beacons used, the separation between beacons, and the position of the user's head. The tracking method used can be either outside-in or inside-out. For outside-in tracking, the additional parameters which include the number of cameras and their position must be specified. For inside-out tracking, the user selects the number of views and their spatial relationship. Any one of the solution methods (direct solution, Ganapathy's method, Church's method, and Fischler's method) reviewed in Section 3.2 may be employed. Finally, the size and the resolution of the photodiode, the maximum error in the placement of the beacons, and the error of the focal length measurement can also be adjusted.

During the simulation, the user's head position can be set interactively or generated randomly by the simulation program. The simulation program then selects three beacons that are inside the field of views of the cameras based on the current head position (the user can also manually select these beacons). The solution method which the user chooses is used to derive the transformation matrix $M$ and to estimate the user's head position. The discrepancy between the computed head position and the correct one is then calculated and reported back to the user. The error in the estimation consists of two terms: one is a translational error which represents the distance between the true head position and the estimated one, and the other is a rotational error between the correct and inferred orientation of the user's line of sight. The user can also use a joystick connected to the Pixel-Planes 4 machine to simulate the head motion. The proper image sequence is generated and displayed to let the user verify the tracking process visually.

#### 3.3.6.1 Errors Induced by Limited Photodiode Resolution

To study the effects of the photodiode resolution on the performance of the system, we used a Monte Carlo method to randomly generate one thousand head positions in a room 4x4x2.5 $m^3$ in size. A three-camera configuration was used in the simulation. These cameras were set to an angular separation of 45° from the vertical direction, and azimuthally separated equally from one another. The focal length was set to be 50 mm for all cameras, and the placement error of beacons was set to be zero (we know the positions of beacons accurately). The generalized Church's method was used to recover the 6D position of the camera assembly. The effect of the photodiode resolution to final system error was determined by changing

the photodiode resolution from one run to the next, and measuring the difference between the correct head position and the one calculated with Church's method.

In Figures 3.8 and 3.9, the average translational and rotational errors (over a thousand runs) of the estimated head position as functions of the photodiode resolution are depicted. The results confirm Equation 3.7 and the curve is hyperbolic in shape. One can observe from these plots that the error in the estimated camera position remains fairly constant over a large range of the photodiode resolution. Also, the photodiode resolution required to achieve a desired accuracy is much coarser than the 1 part in 4000 which we originally expected to be necessary. The reason that we can tolerate a much poorer photodiode resolution is because in a three-camera configuration, each camera covers only a small field of view. Hence, movements inside this small viewing field can be detected with relatively poor photodiode resolution.

In the current system, we will use photodiodes with a minimum resolution of at least 1 part in 1000. With this resolution, we can achieve a resolution of less than 5mm in position error and less than 0.4 degree in orientation error in a room size environment, which is the requirement listed in Section 3.1.1.

#### 3.3.6.2 Errors Induced by the Misplacement of Beacons

To quantify the effect of the placement error of beacons, the simulation conditions in the previous simulation were used. However, the resolution of the photodiode was assumed to be infinite while the placement error of beacons was made to vary. Figures 3.10 and 3.11 depict the average errors (again, over a thousand runs) of the translational and rotational measurements induced by error in the placement of beacons. These results agree with Equation 3.10 and the error in the estimated position of the user's head increases linearly with that of the beacon positions.

In the current system, we aim at measuring the beacon position down to $\pm$5mm maximum error. This accuracy will enable us to achieve an accuracy of 0.8 degree in rotation movement and 1 cm in translational movement, which is the requirement listed in Section 3.1.1.

### 3.3.7 Combined Errors

The error in the final position report can be induced by either of these two effects. In Figures 3.12 and 3.13, the system error is plotted as a function of the photodiode resolution. Each curve in Figures 3.12 and 3.13 represents a different beacon placement error. As

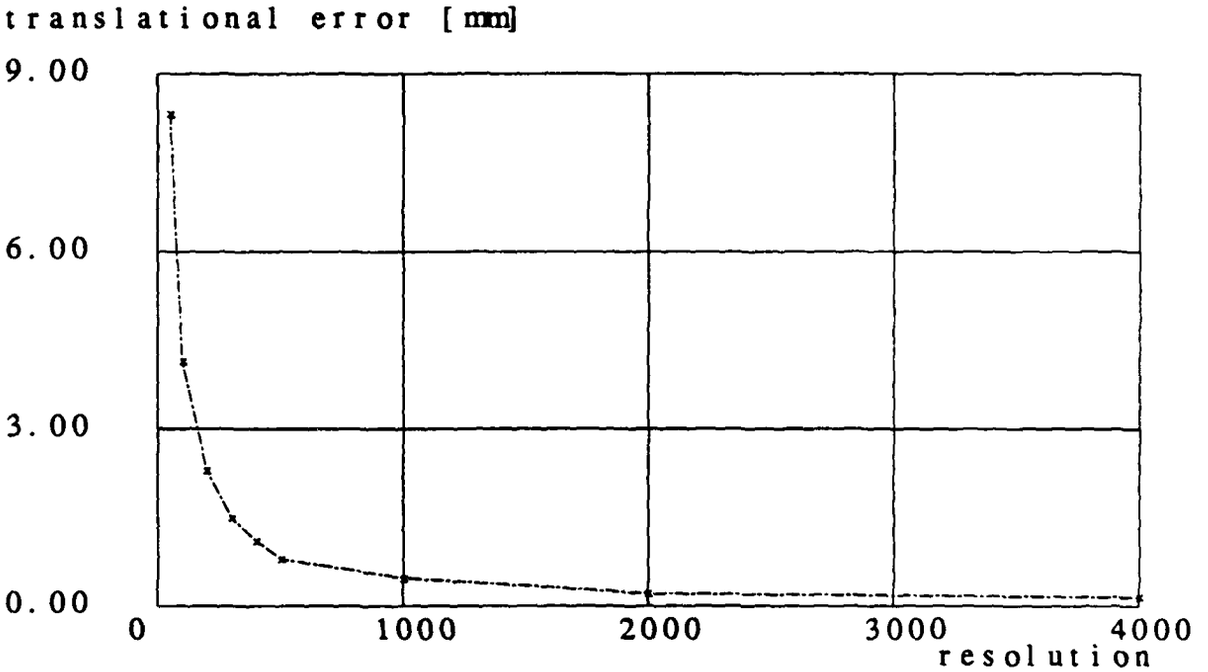



Figure 3.8: Translational error vs photodiode resolution

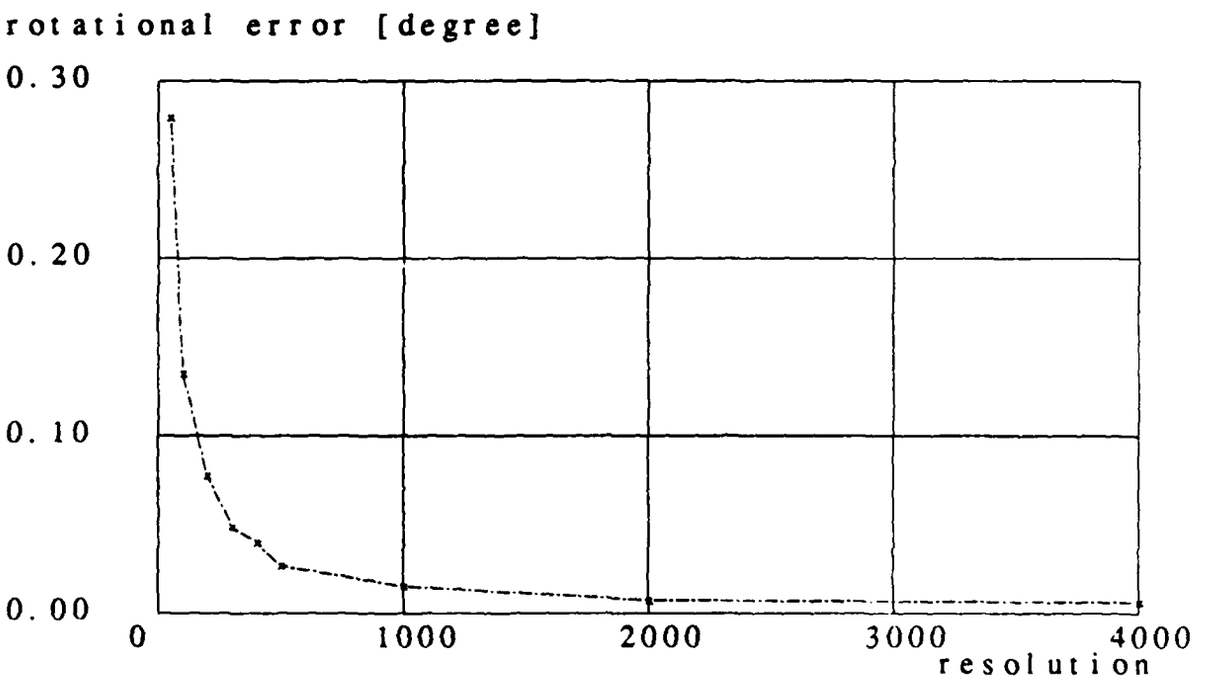



Figure 3.9: Rotational error vs photodiode resolution

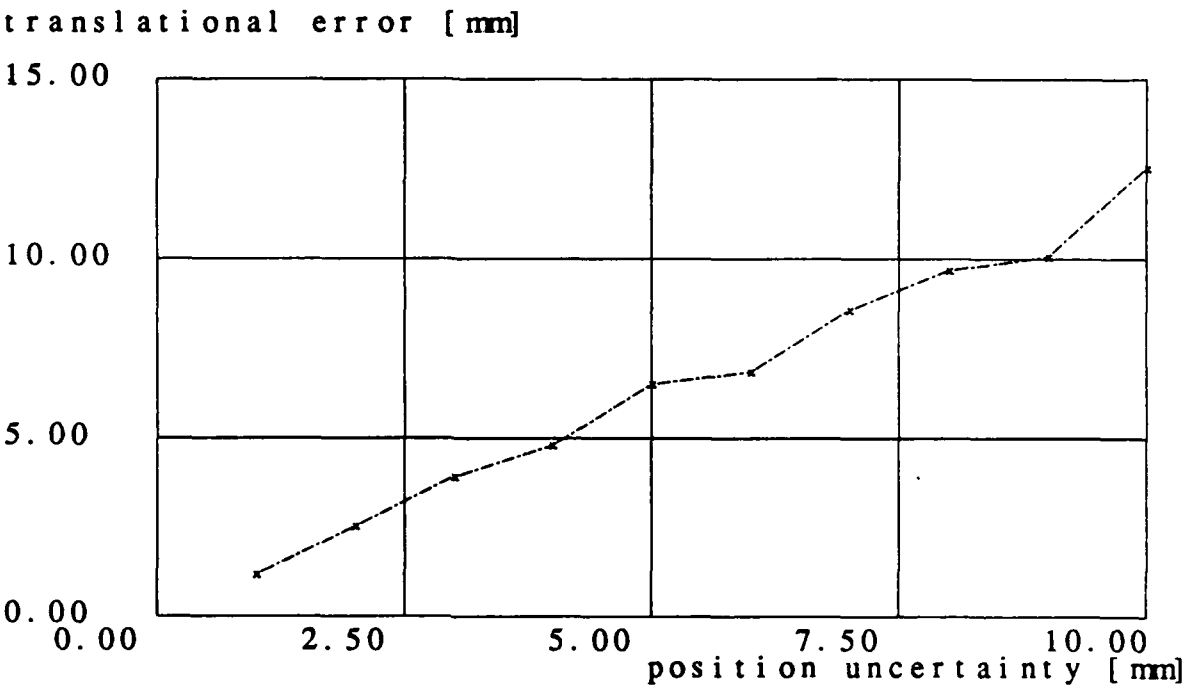



Figure 3.10: Translational error vs uncertainty in beacon position

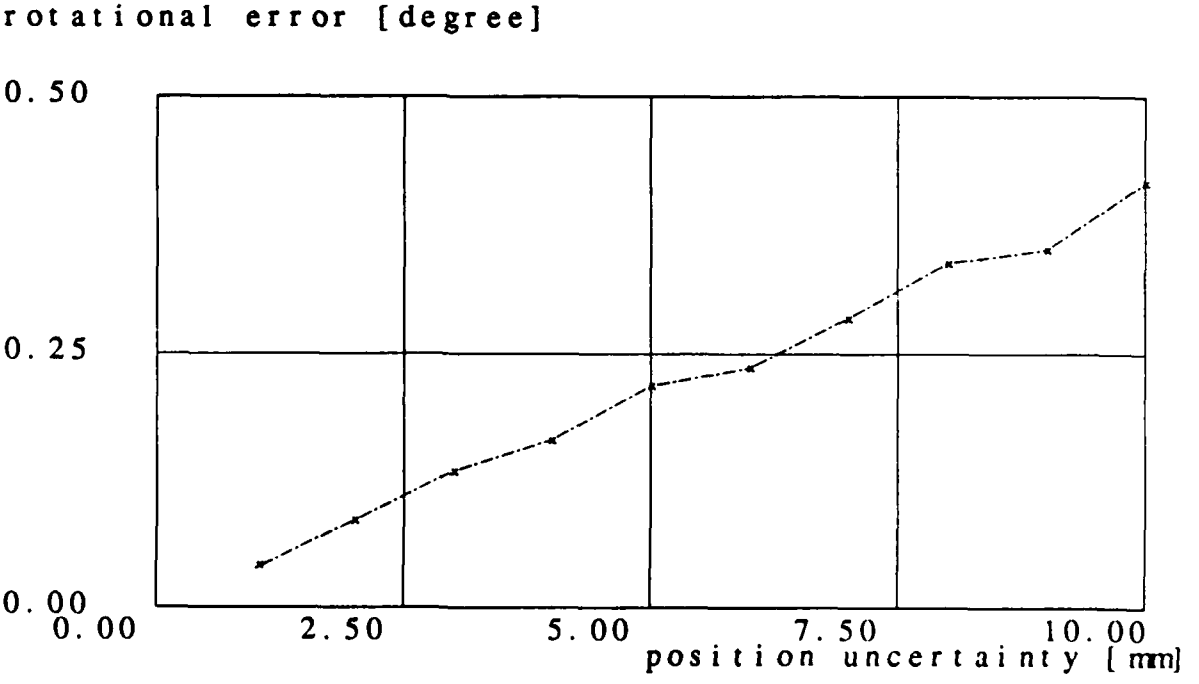



Figure 3.11: Rotational error vs uncertainly in beacon position

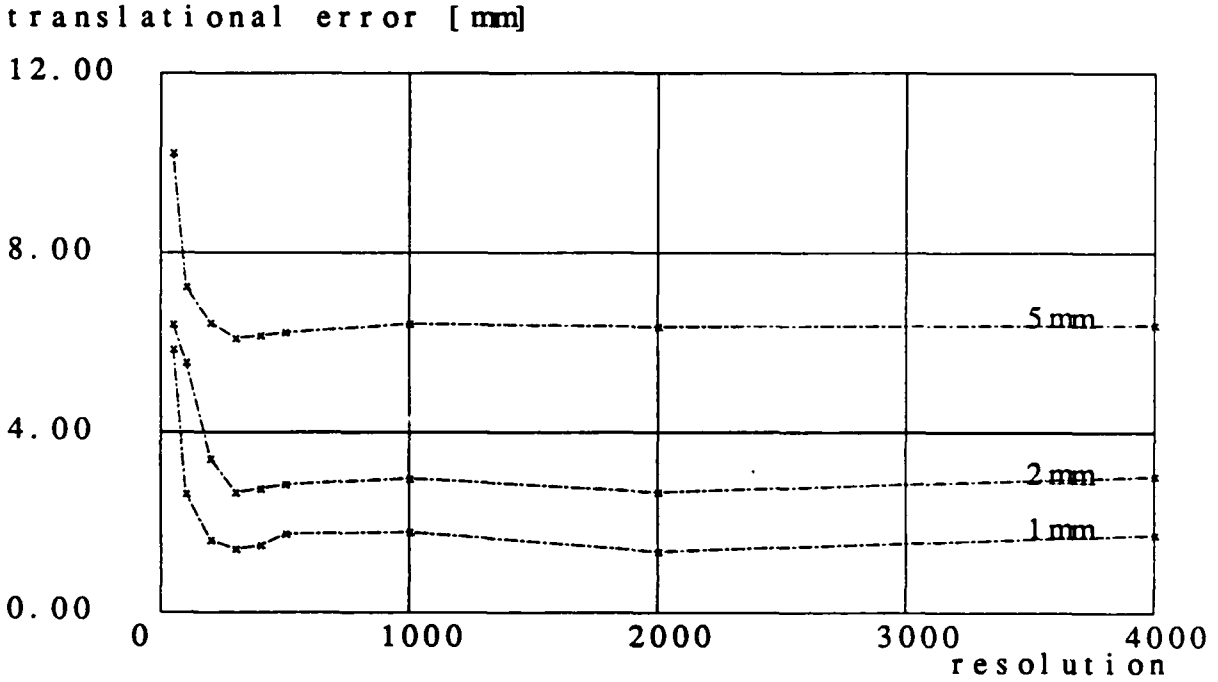



Figure 3.12: Translational error vs resolution with different position uncertainty

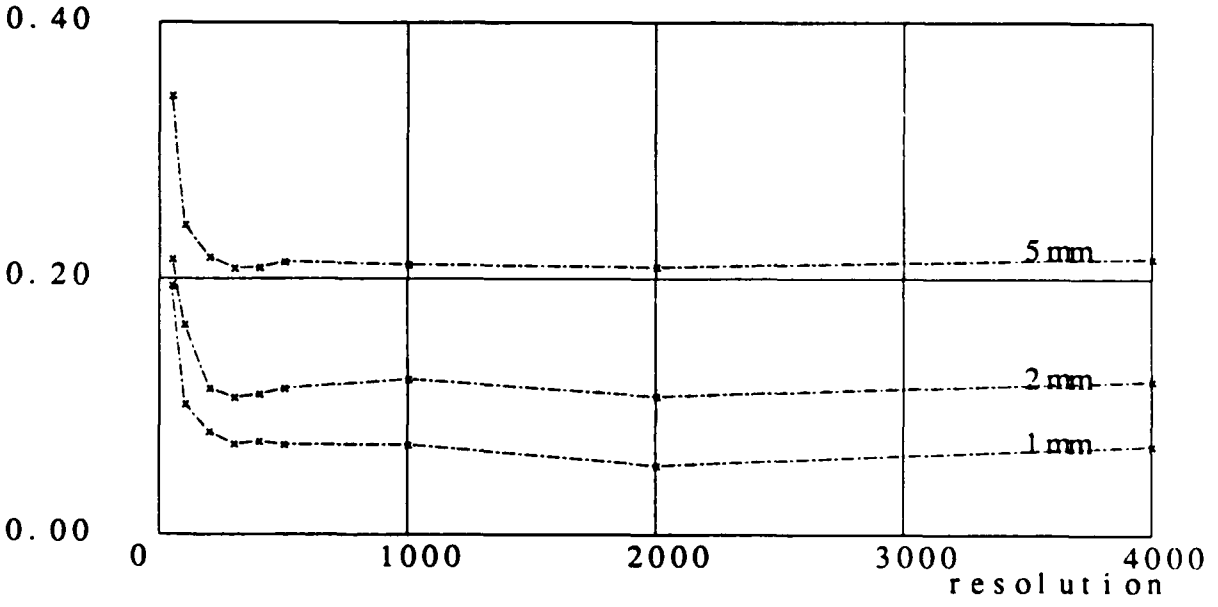



Figure 3.13: Rotational error vs resolution with different position uncertainty

can be seen from these plots, the system error is primarily induced by limited photodiode resolution— if the resolution is poor. If the resolution of the photodiode is above some crossover point (approximately 1 part in 200), the system error becomes dominated by error in the placement of beacons.

To understand how system errors dominate, consider the following example. If a camera with a 50mm lens and a 1 $cm^2$ photodiode recording surface is directed at a beacon 2 meters away as shown in Figure 3.14, a misplacement of the beacon by 5mm corresponds to a 0.125mm shift in the photodiode surface. A photodiode with a resolution of 1 part in 80 (10/0.125) will report the misplacement of the LED position. Hence, increasing the resolution of the photodiode will not increase the accuracy of the estimated image position because the error in beacon placement is an independent factor. Increasing photodiode resolution does increase the *resolution* of the system, which is a very important consequence that will be discussed in Section 3.4.2. The simulation results clearly suggest that the two error sources should be optimized together. It does not help to have a photodiode with high resolution unless we can place beacons more precisely.

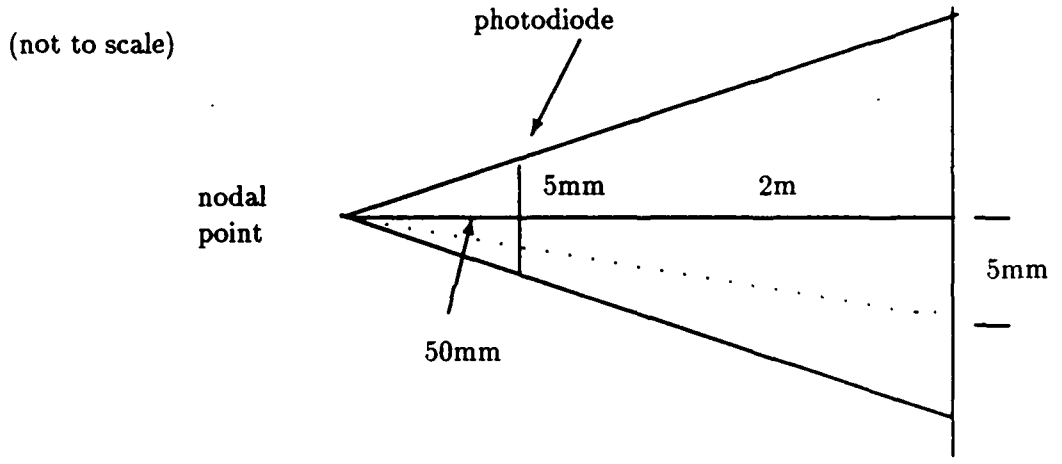



Figure 3.14: Comparing the two error sources

Now, examine Figure 3.14 more carefully. Suppose that we have a photodiode with a 1 part in 1000 resolution. Based on the above argument, we need to know the beacon position to ±0.4 mm (1cm/$r$*(50mm/2m)). This is a very strict requirement, and this requirement is difficult, if not impossible, to satisfy if several hundred LEDs are used in the systems and each individual LED has to be placed to that accuracy. We present a partial solution to this problem in the next section.

## 3.4 Self-Calibration of the Camera Unit

As pointed out in Section 3.1.1, there are two different criteria (*accuracy* and *resolution*) in judging the performance of a head-mounted display system. In this section, we propose two methods: *smoothing* and *camera self-calibration*, which can be applied to improve the accuracy of head-mounted systems.

In Church's method, beacons are used as the bases for inferring 6D head position. Errors in the positions of beacons will degrade the accuracy in the estimated head position. However, the misplacement of beacons will *not* affect the *resolution* of the system, which is governed by the photodiode used. In some head-mounted systems where the user's field of view is completely covered to display some virtual world environment (e.g. VPL EyePhone), then the user is not aware of his or her position in the real world. Hence, as long as the animation sequence depicts a smooth motion and the motion corresponds roughly to the head movement of the user, the user's perception will be of moving smoothly in a virtual world.

A simulation was performed to study the system error in such dynamic environments. Figures 3.15 and 3.16 shows the error between the calculated and true head position when the user moves along a smooth path across a virtual room. This path was a straight line 2 meters in length between two randomly chosen end points in the virtual room. The head was tilted from $-35°$ to $35°$ back to front and sideways, and rotated horizontally $180°$ along the path. The simulation was done under the same conditions as the previous simulations but with both error sources (photodiode resolution and beacon placement error) turned on. In this particular simulation, photodiode resolution was set to 1 part in 1000, and the maximum placement error of beacons was 5mm (Figure 3.15) and 2cm (Figure 3.16). Error curves for rotational error have similar characteristic shapes similar to Figures 3.15 and 3.16, but with different scales. The error curve resembles a square waveform with jumps between the plateaus. High frequency error components are easily noticeable from the plots. So these plots are characterized by high frequency noise and considerably more error with larger beacon placement error.

As the user moves about the room, if one beacon used in the previous sampling interval moves out of sight, a new visible beacon has to be selected to continue the tracking process. In Figures 3.17 and 3.18, we superimpose on Figures 3.15 and 3.16 vertical dashed lines indicating where the system switches beacons. It becomes obvious from Figures 3.17 and 3.18 that the switching of beacons causes the abrupt jumps in the error curve.

This effect can be accounted for by observing that in Church's algorithm, beacons are used as references to infer the current head position. Errors in the position of beacons

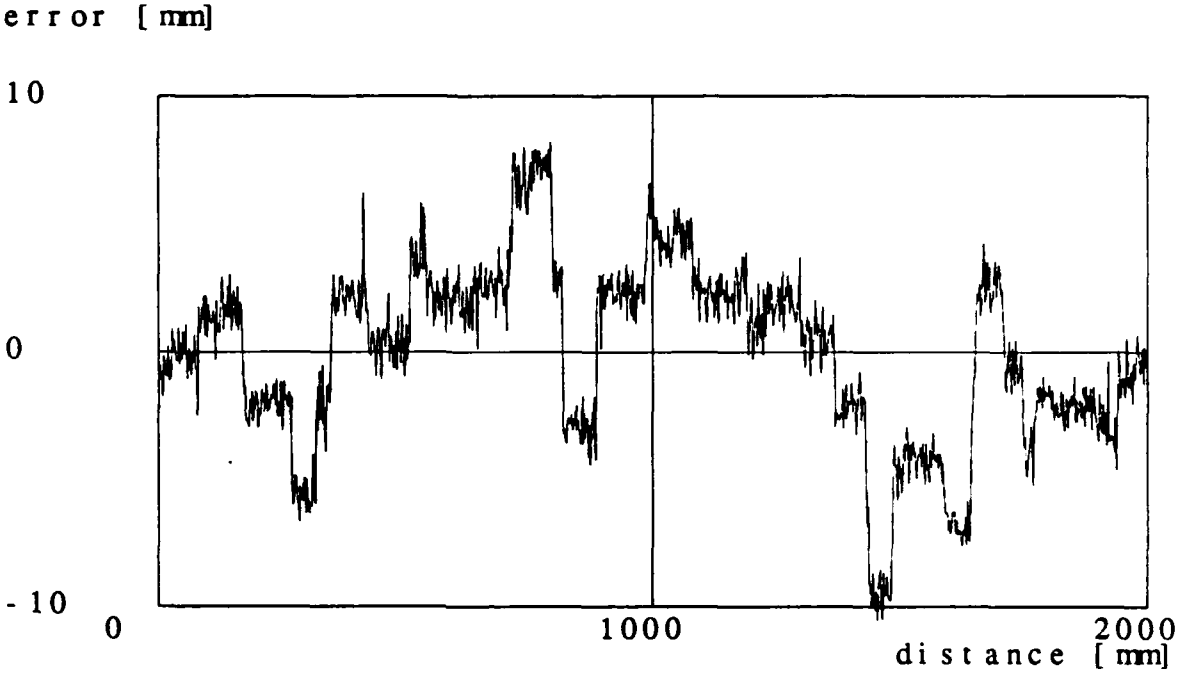



Figure 3.15: Error along a random path I: beacon position error = 5mm

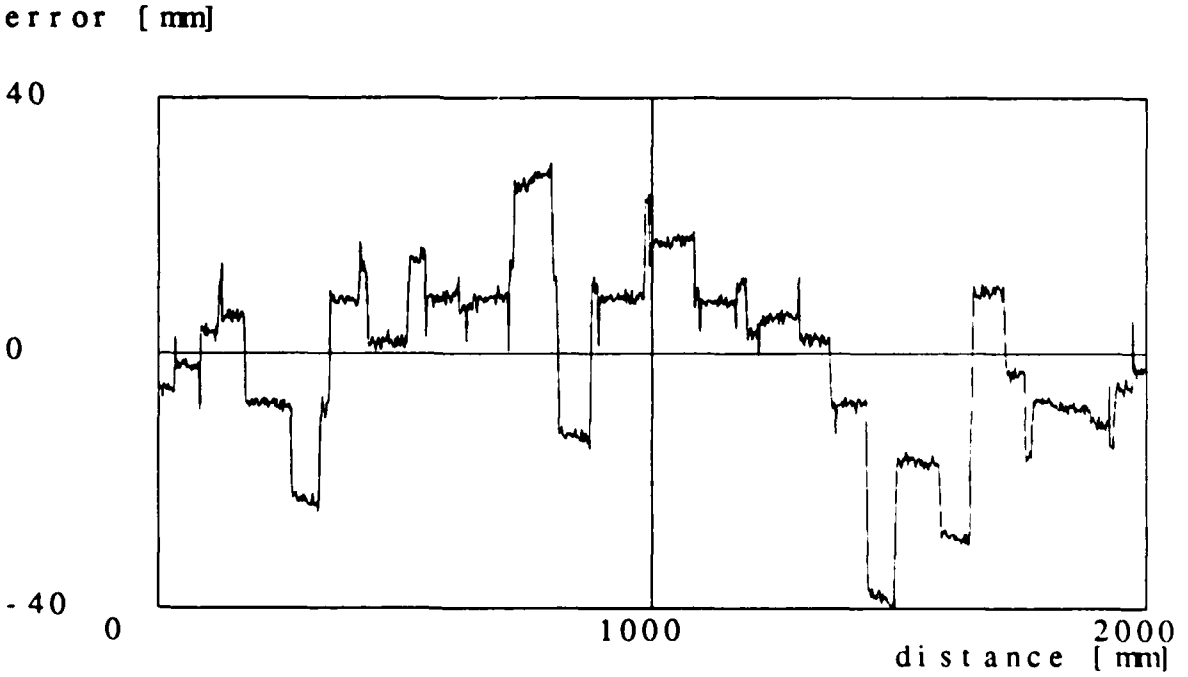



Figure 3.16: Error along a random path II: beacon position error = 2cm

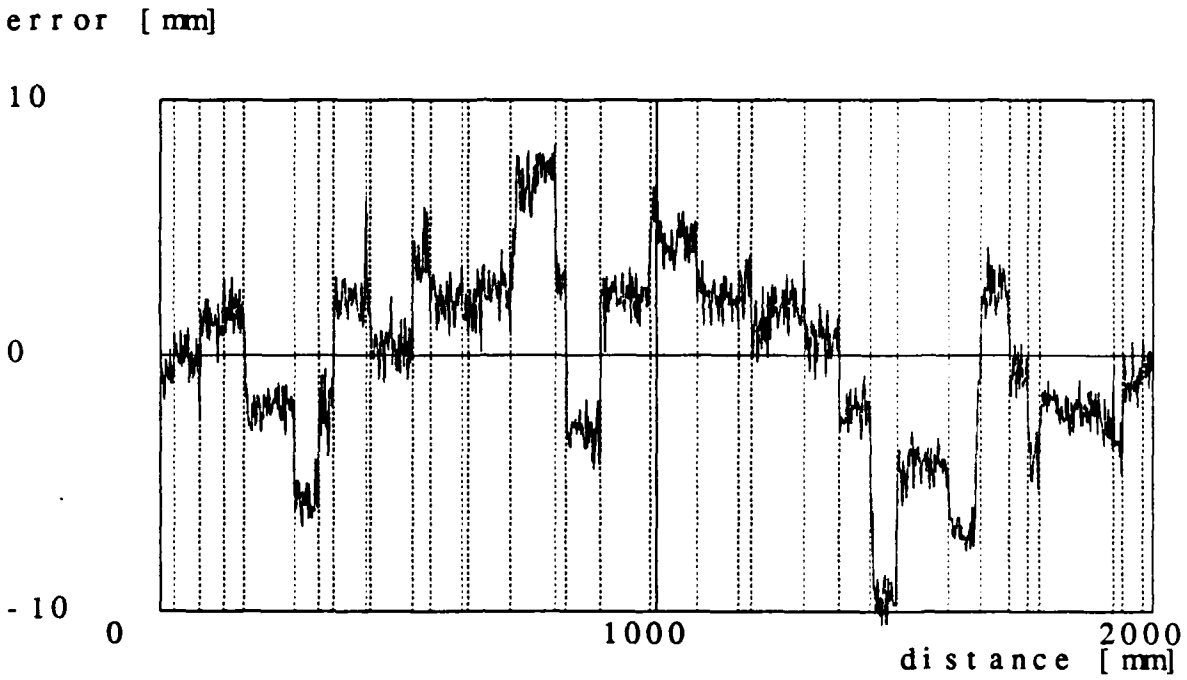



Figure 3.17: The system switches beacons at the dashed line positions I: beacon position error = 5mm

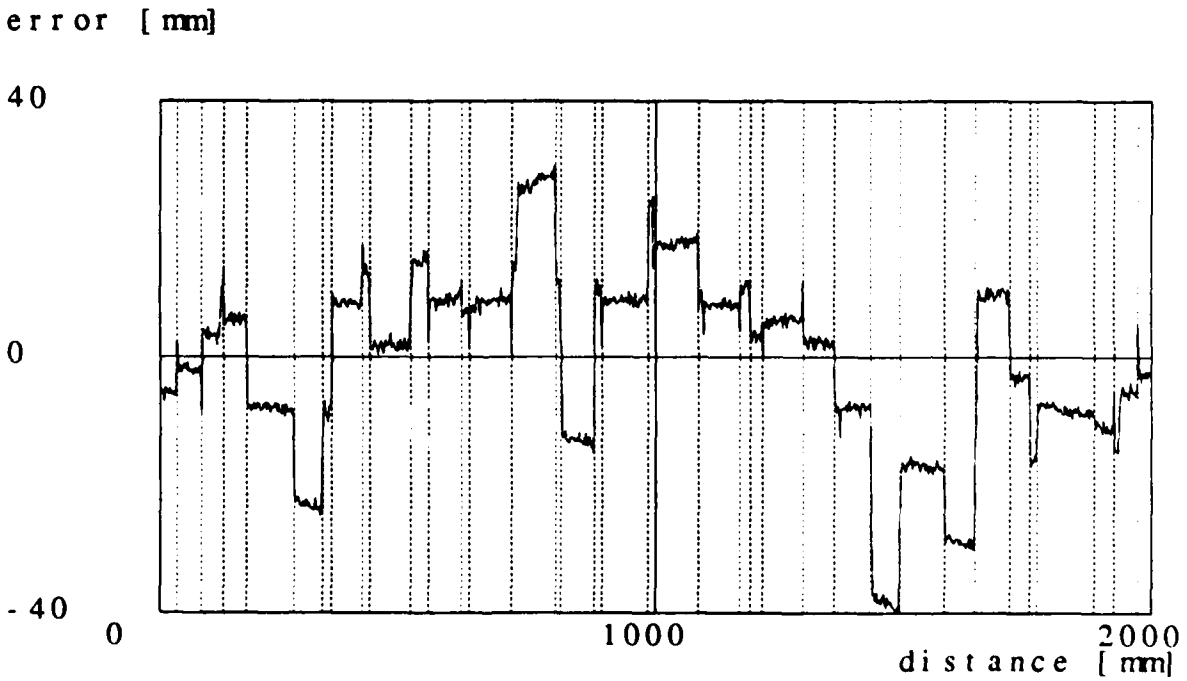



Figure 3.18: The system switches beacons at the dashed line positions II: beacon position error = 2cm

will introduce error in the estimated head position even though the photodiode used has an infinite resolution. That is, the *accuracy* of the system is limited by the placement error of beacons. When a different set of beacons are selected, a different error is introduced into the estimated head position, which explains the sudden jumps. On the other hand, limited photodiode resolution creates the "jittering" along the path. Jittering shows up as the high frequency components in the error curve. That is, the *resolution* of the system is limited by the photodiode resolution. Note that noise inherent in the photodiode readouts will also produce jittering, which is discussed in Section 4.1.1.

Referring to Figure 3.16 again, it is now desired to decrease the amplitude of the square wave (or "flatten" the curve) by placing the beacons more accurately. We can also smooth out the high-frequency error components by using a photodiode with higher resolution. Below, we introduce two methods: the first method tries to smooth the path by interpolating the head position along the path, This method is useful when the beacon placement error is small (Figure 3.15), or in virtual-world applications where accuracy is not a major issue (e.g. walking around a building during its design phase, or displaying molecular structures in a void environment without real-world objects as backdrop). The second method is to let the camera to automatically calibrate beacon positions. This method can be used when both resolution and accuracy are important. These two methods can also be used together to achieve even better results.

### 3.4.1 Smoothing

The sudden jumps in the error curves in Figures 3.15 and 3.16 will cause jumps or discontinuities in the displayed image sequences in the head set. The discontinuities can be made less noticeable by interpolating the path. That is, when the system switches from one set of beacons to another set of beacons, the system retains the information on the old head position and calculates the new head position using the new set of beacons. Now, instead of displaying the images based on the new head position, the system displays the images according to some weighted average of the old and new head positions. Effectively, we smooth the transition of the displayed image sequence. In Figures 3.19 and 3.20, smoothing is done using a simple weighted average of the old and new head positions. The weight of the new position increases after each head position updated. Although smoothing does get rid of the abrupt change in the display image sequence, the images can still "swim" around. The "swim" effect occurs when the system switches from one set of beacons to another set of beacons. Hence, even if the user is walking straight along a fixed direction, the displayed image sequence "zigzags" due to the smoothing effect.

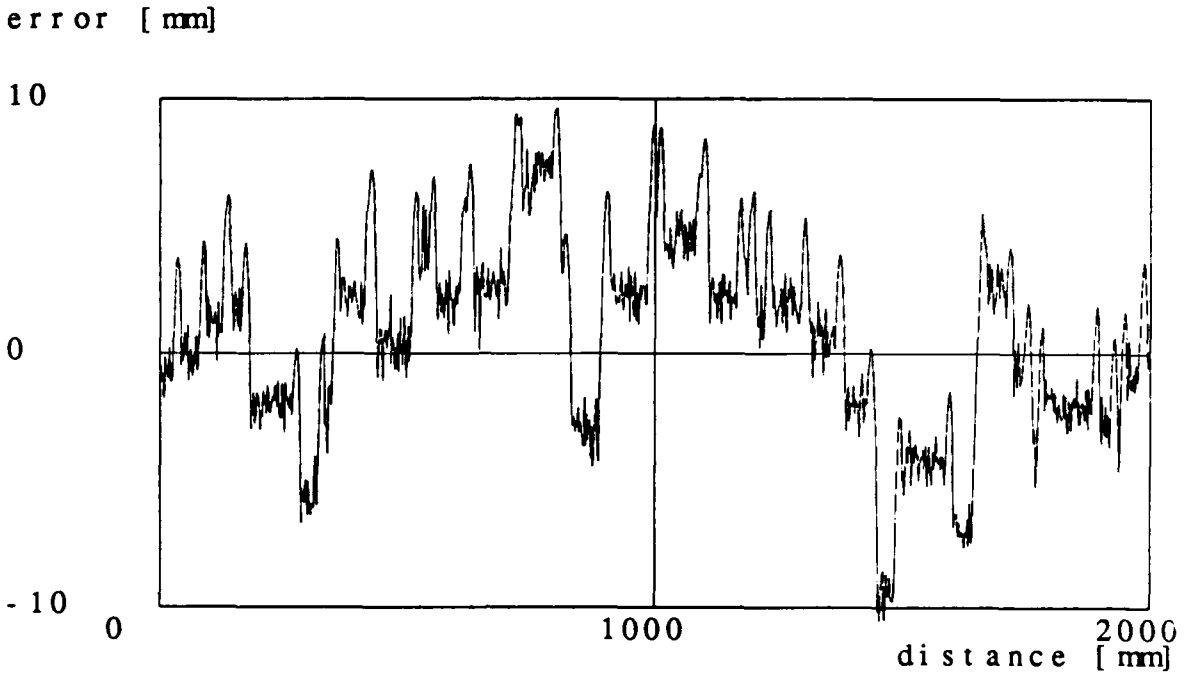



Figure 3.19: Error along a random path with smoothing I: beacon position error = 5mm

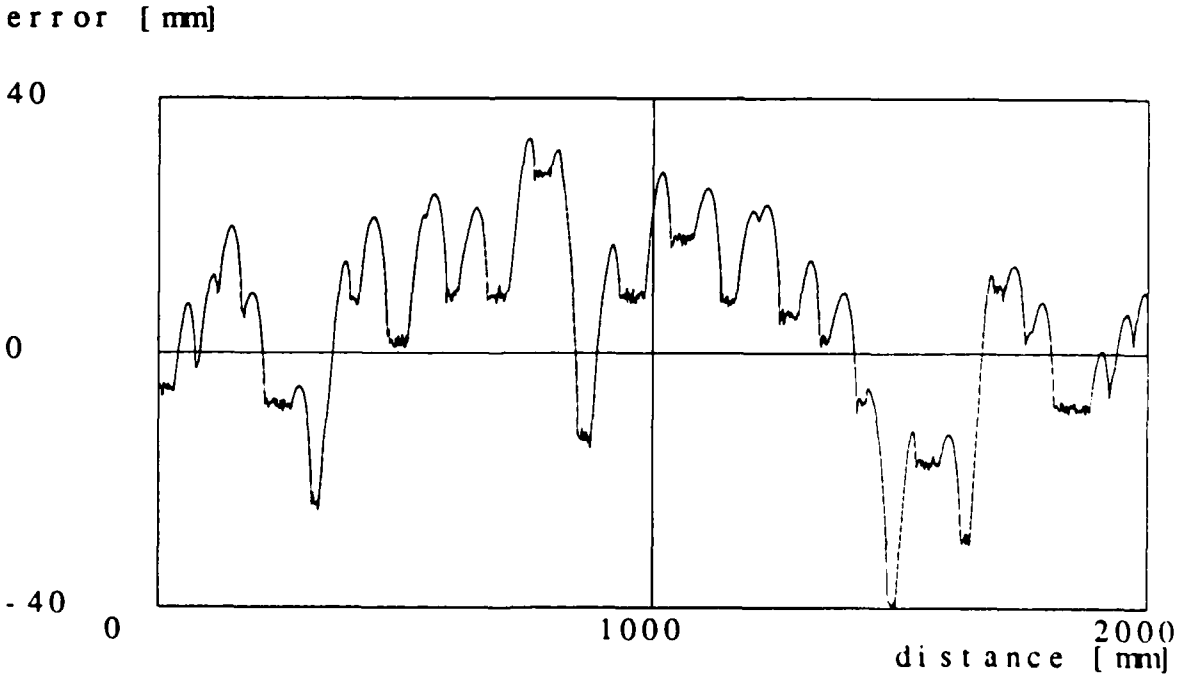



Figure 3.20: Error along a random path with smoothing II: beacon position error = 2cm

### 3.4.2 Camera Self-Calibration

Smoothing alone does not decrease the positional error, it only makes the error less noticeable. To decrease the error and to minimize the "swim" effect, it is necessary to "flatten" the error curve—not just to interpolate paths between jumps. In other words, we want to keep the amplitude at every point along the curve below some fixed threshold value. The basic idea is to use the higher resolution of the photodiode camera to automatically determine the locations of ceiling-mounted LEDs more accurately during the tracking process.

The camera self-calibration process assumes that all LEDs are affixed in a regular pattern on the ceiling, so their height is fixed. This self-calibration process uses a small number of beacons (the initial set), the positions of which are assumed to be precisely measured, and the beacons in the initial set form a coordinate system. The system infers the position of the camera assembly with a high accuracy using this initial set of beacons. Before the tracker moves to a new position and uses a new set of beacons for tracking (which might include those not in the initial set), the beacons surrounding those in the initial set are flashed. The 3D position of the flashed beacons can be computed using the known head position (inferred using the initial beacon set) and the 2D projection of the flashed beacons. The current head position and the 2D image of a beacon determine a line on which the flashing beacon must lie, and the fixed ceiling height gives us another constraint to fix the beacon position along that line, as shown in Figure 3.21.

Note that in an actual implementation, the LEDs will probably be mounted on circuit boards, so their positions on the boards will be quite accurate. It should be easy to place them with an accuracy of ±5mm with today's technology. Actually, we probably will be able to place them with error within 1mm on the boards, but we still have to worry about placement error between two adjacent boards, which can be much larger. So in the remaining analysis, we will use 2cm (Figure 3.16) as the bound on maximum placement error. We show that it is possible to reduce the beacon placement error through the self-calibration process as described below.

Using the initial set as a reference, the system determines accurately the positions of beacons not in the initial set. The position of these neighboring beacons can then be in the subsequent tracking process. The advantage of this automatic calibration process it that because the resolution of the photodiode is relatively high, we compute a position reading for those beacons much more accurately than we can do manual measurements. The error curve becomes "flattened" even though the curve might not lie anywhere close to zero if the placement error of those beacons in the initial set is large.

Camera self-calibration should be performed before the system is put in use. That

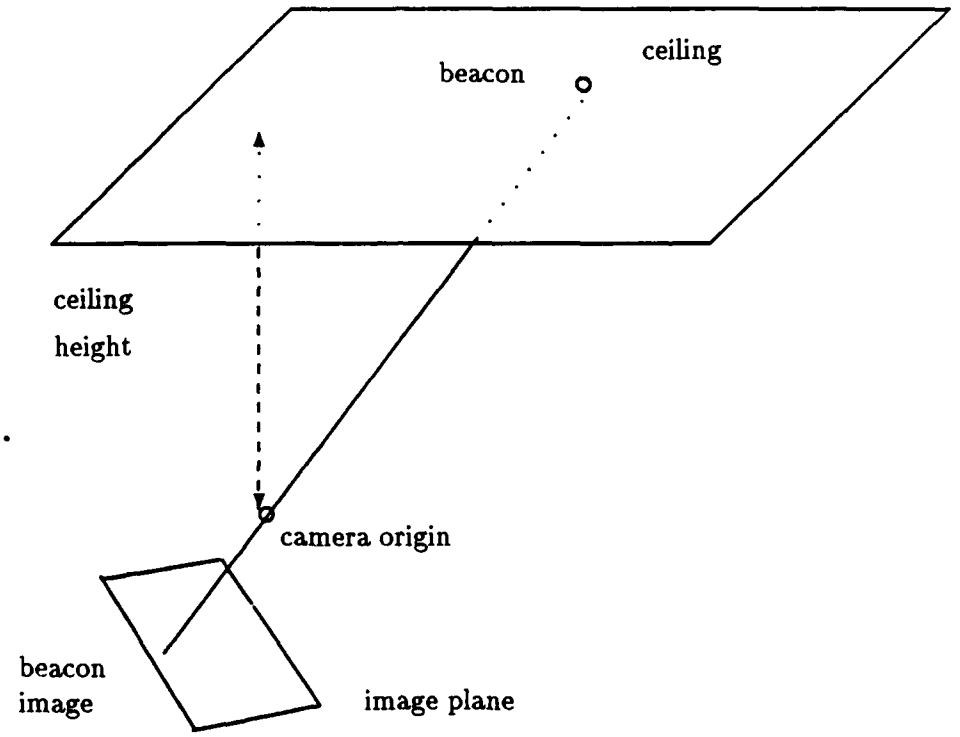



Figure 3.21: Camera self-calibration

is, after the system is set up and beacons are placed, the locations of a small set of beacons are measured. The self-calibration process is then adopted to calibrate positions of the rest of beacons. Further, since the sampling rate of our tracking system is very high (see later chapters), this calibration can be performed while the user walks slowly around the environment during an initial "training" run. The tracker will flash the beacons surrounding the current active ones and derives the positions of those flashed beacons based on the current head position. In Figures 3.22 and 3.23, we show the results of this camera self-calibration process. The simulation assumed a scenario where the user wearing the helmet was walking slowly around in a virtual room. As can be seen, although camera self-calibration does give us a smooth curve with no big jumps anywhere, it is not totally satisfactory due to the error accumulation problem. Error accumulation comes from the fact that since new LED positions are inferred based upon the current calculated head position, the error in the current head position tends to magnify itself through the process. The problem becomes more significant as the beacon placement error increases.

In order to solve this problem, some initial LED position information is needed as reference to keep the error under control, but the information doesn't have to be very accurate. The initial reference positions can be obtained by roughly measuring the beacon

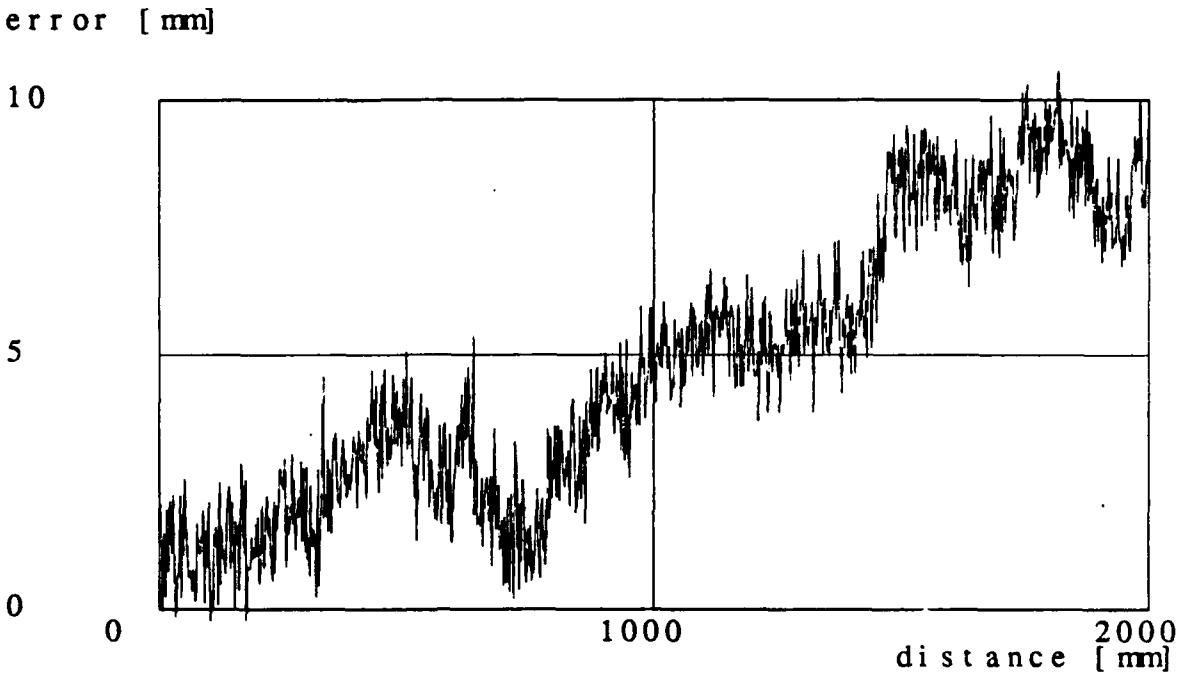



Figure 3.22: Error along a path with beacon under camera calibration with no knowledge of beacon positions I: beacon position error = 5mm

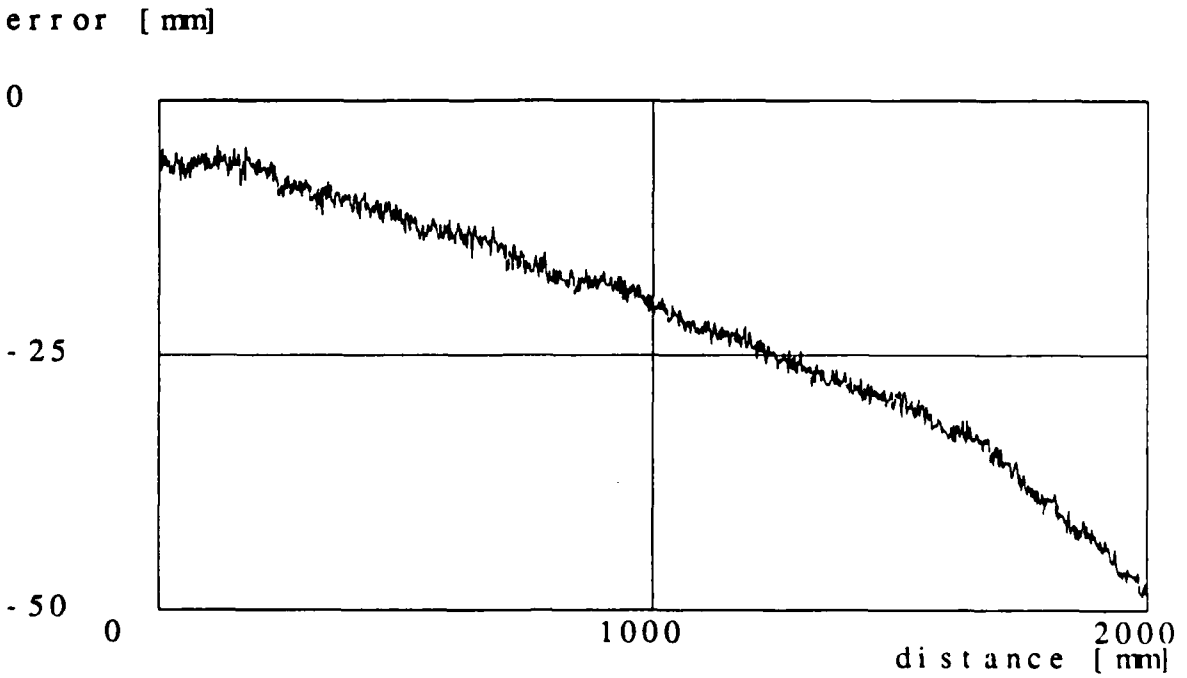



Figure 3.23: Error along a path with beacon under camera calibration with no knowledge of beacon positions II: beacon position error = 2cm

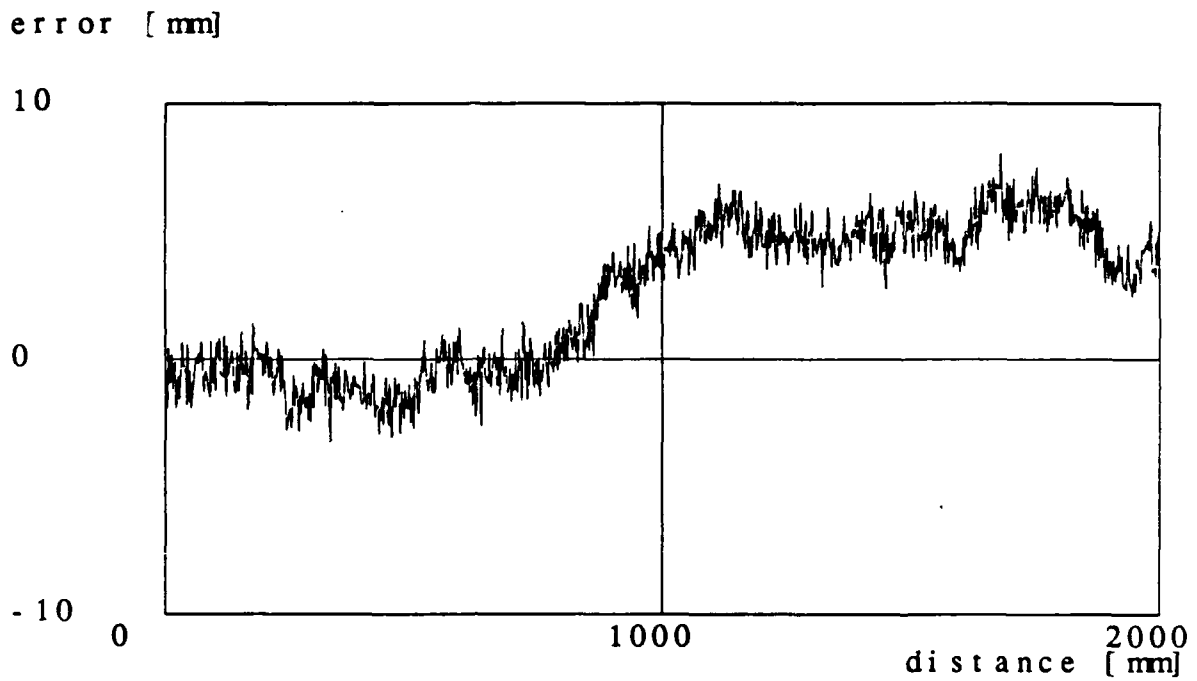



Figure 3.24: Error along a path with beacon under improved camera calibration with known beacon positions I: beacon position error = 5mm

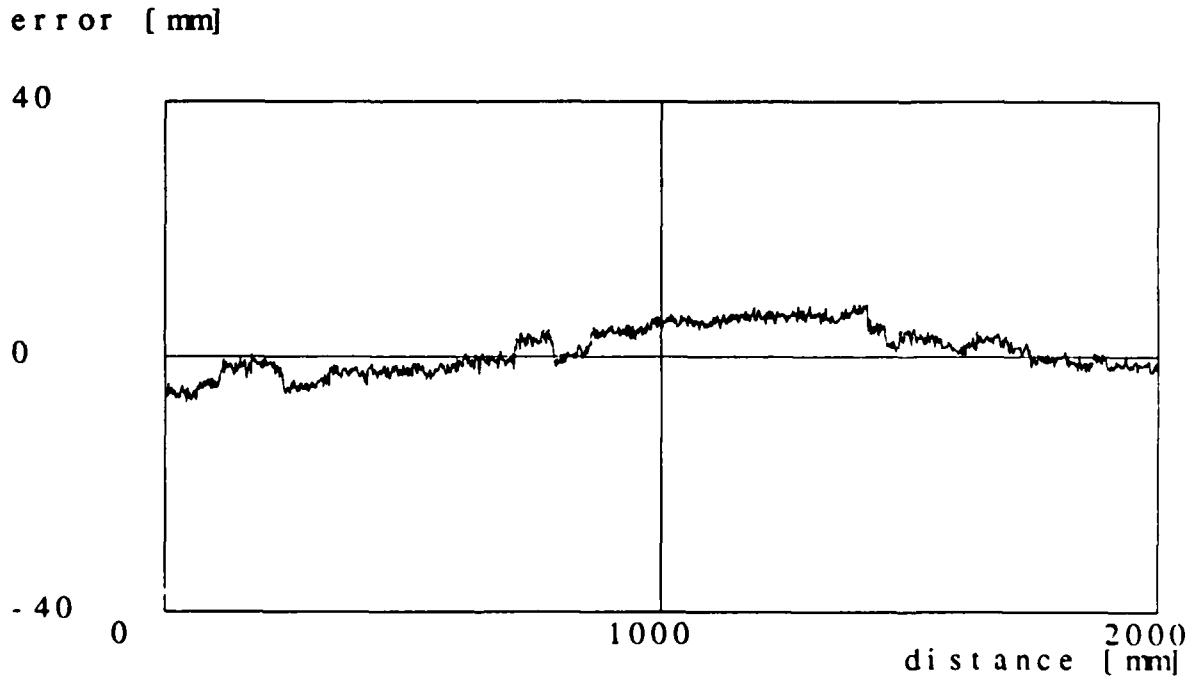



Figure 3.25: Error along a path with beacon under improved camera calibration with known beacon positions II: beacon position error = 2cm

positions when they are placed. In our system, the LEDs will be mounted on circuit boards, so their positions can be determined with a certain accuracy. If the maximum error in the initial LED placement doesn't exceed, say 0.5cm, then the beacon positions derived from camera self-calibration can be averaged with the original measurement. By adjusting the weights given to the original measurement and the data derived from self-calibration when averaging, we can prevent errors from accumulating. In Figures 3.24 and 3.25, we show the simulation result under the same situation as before, but the derived LED positions from camera self-calibration are averaged with the original measurement. The result is in the same scale as in Figures 3.15 and 3.16 so we can easily compare them. The curve shows that we can effectively flatten the error curve with camera self-calibration, even if there is a large error in the initial measured beacon positions.

## 3.5 Errors Induced by Motion

Finally, besides the errors studied above, the user's motion will also introduce error into the recovered position. This dynamic error comes from the fact that since a photodiode can only register the position of a single light source at a time, the three beacons used for the tracking process must be flashed sequentially. During the sampling period from flashing the first beacon to flashing the third beacon, the user might have moved a little. The movement will introduce some shift to the beacon images on the photodiode surfaces, which induces error to the recovered position.

In order to determine the error induced by the user's motion, we have to first determine how fast a person can move his/her head. According to the study done by Gary Bishop [Bis84], 70 degrees per second is the natural rate for most head rotations, 200 degrees should give us a good safety margin for normal rotations, and 750 degrees per second is the peak rate. We simulate the motion error under the following circumstance: the resolution of the photodiode is set at 1 part in 1000, the flashing between beacons is 100 $\mu$sec, and we simulate the user rotating his/her head at the speed of 70°, 200°, 360°, and 750° per second. Figure 3.26 shows the results. As can be seen, under normal motion (75° to 200°), the error induced by motion is very small. Besides, the error is not random, since the error curves lie at a fixed distance from the zero error line. This error can be corrected by motion prediction which is not covered in this thesis, or ignored for an opaque system like EyePhone.

The reason that only a very small amount of error is induced by motion is due to the high sampling speed. Since motion induces image shift, if we treat the image shift as error, then the error basically reduces the resolution of the photodiode. From Figure 3.8 and 3.9, we know that reduced photodiode resolution doesn't increase the final system error linearly. Instead, there is a threshold point below which the error increases sharply. In

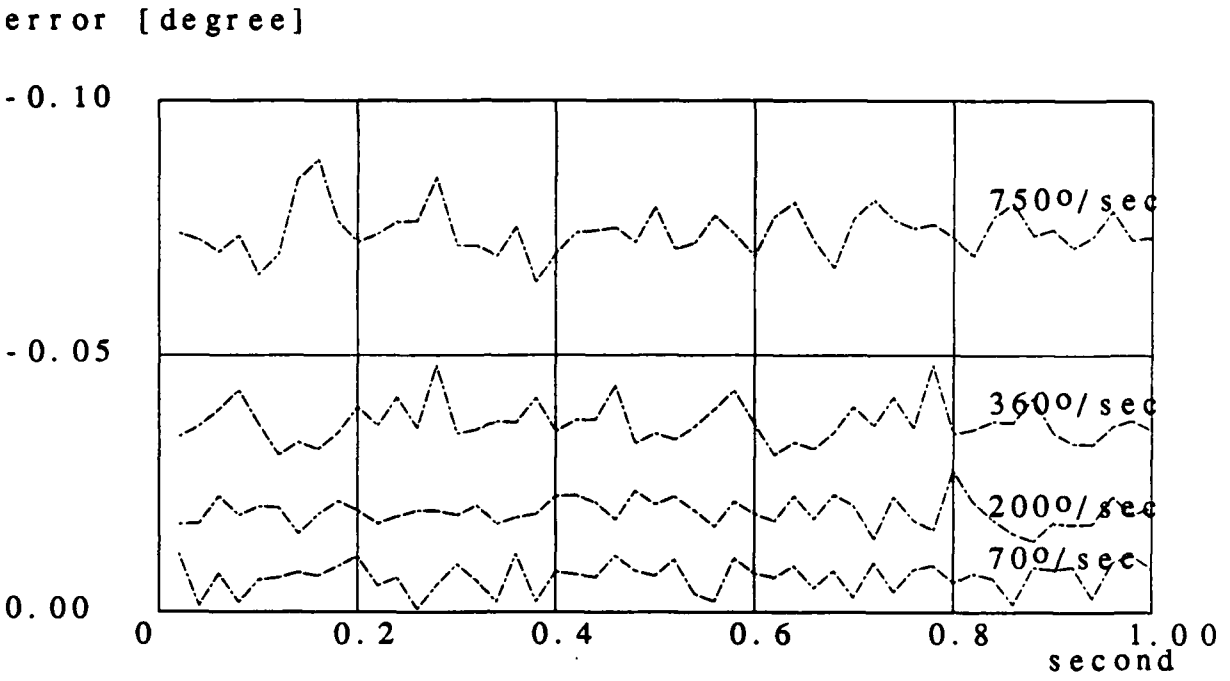



Figure 3.26: Errors induced by motion

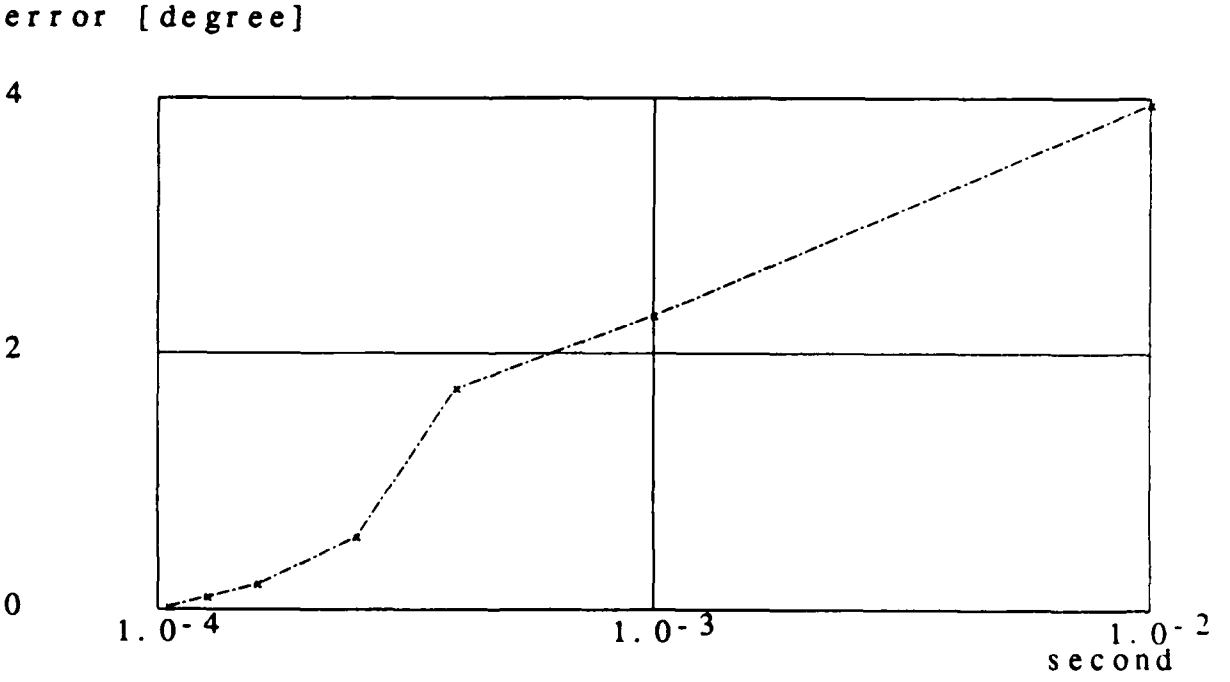



Figure 3.27: Errors under different sampling speeds

order tc determine the relationship between the sampling speed and system error, we also simulate a 200° per second motion under different sampling speeds. It can be seen clearly from Figure 3.27 that the sampling speed must be very fast, otherwise the image shift will become a problem. At the current sampling speed (100 $\mu$sec), the error induced by the user's motion is very small ($\sim$ 0.02°) and can be safely ignored under the criteria we set.

# Chapter 4

# The Bench-Top Prototype System

In this chapter, a laboratory bench-top prototype tracking system is described. This prototype system was constructed to prove the correctness of our design. Although limited in working volume, this prototype nonetheless demonstrates the integration and coordination of all essential components in the new tracker. The prototype shows that the 6D head position and orientation of the observer can be estimated if the positions of a sufficient number of beacons are observed, and that the tracking system can provide a fast update of the head position to maintain an illusion of a smooth motion, and a small lag so that stationary virtual objects will appear stable.

The prototype consists of three cameras on a mounting stage which provides two rotational (pitch and roll) and two translational (vertical and horizontal) degrees of freedom. The other rotational (yaw) and translational degrees of freedom are achieved by simply rotating the whole stage. Three infrared light emitting diodes (IR LEDs) are used as beacons of the system with one LED inside each camera's field of view. The outputs from the cameras are processed by signal processing circuitry which converts the analog signals to digital form. A parallel interface is used to connect the digital output from the circuitry to a general-purpose computer, which executes the generalized Church's algorithm for position recovery. The configuration of the prototype system is shown in Figure 4.1. In the prototype, all but the LED addressing circuitry (dotted lines in Figure 4.1) are implemented. Since there are only three infrared LEDs in this prototype, a simple multiplexer is used to select each individual LED (The final full scale system will contain several hundreds or even thousands of LEDs, and the addressing circuitry is still under design).

In the following sections, the components that make up the prototype—cameras, LEDs, photodiodes and the signal processing circuit—are described in more detail.

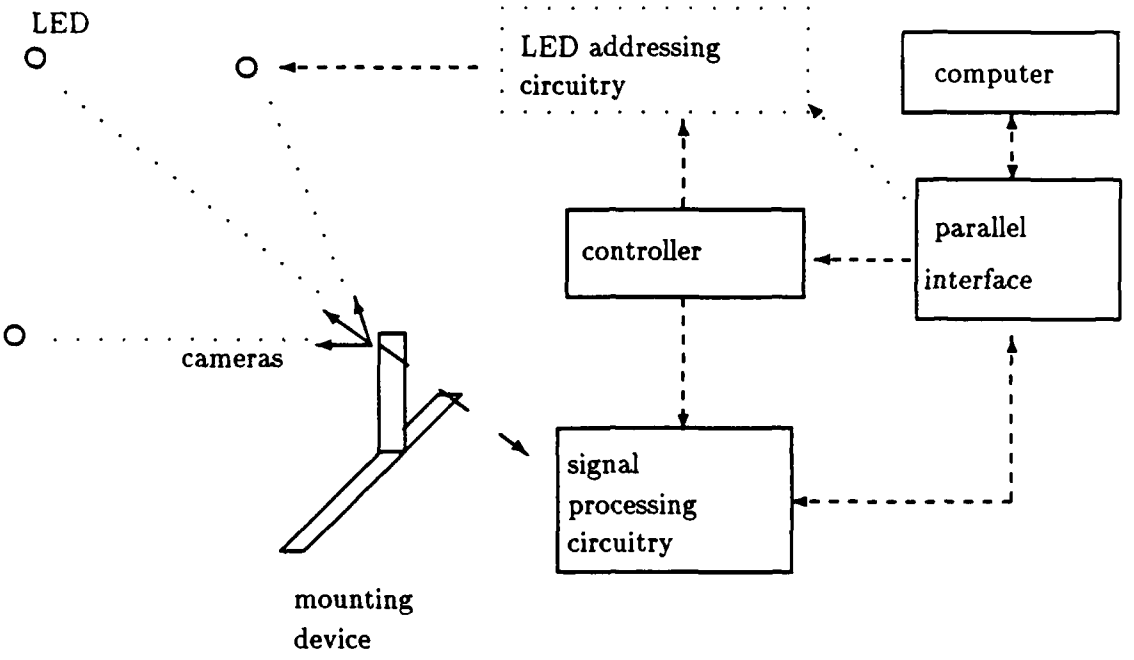



Figure 4.1: System Configuration

## 4.1 Camera

Three cameras are used. Each camera uses a lateral effect photodiode as the recording surface and contains a special-purpose circuit to amplify and process the output signals from the photodiode. The lateral effect photodiode used is a Hamamatsu S1880 pin-cushion type photodiode [Ham85]. Pin-cushion type photodiodes measure current output from the four corners of the photodiode surface, instead of from the four sides of the surface as the type described in Section 2.2.2.1. This type of photodiode has smaller dark current (described in Section 4.3), faster response, and smaller signal distortion along the circumference (described in Section 4.1.2).

### 4.1.1 Circuit Noise Analysis

Noise is inherent in all electronics devices. For photodiodes, noise limits the resolution, causes erroneous outputs, and induces jittering in the displayed images. Jittering is the small trembling motion in the displayed image even if the user of the head-mounted display remains perfectly still. Both the photodiode and the amplifier introduce noise which degrades the performance of the system. Hence, it is crucial to study quantitatively the

effects of the noise introduced by both the photodiode and the amplifier.

Generally speaking, there are many more varieties of amplifiers than there are photodiodes. Hence, it is possible to select amplifiers with the desired performance and low noise, such that the noise in the amplifiers is insignificant compared to the noise in the photodiode. The noise characteristics of the photodiode can then be studied and special filters such as Kalman filters [Bro83] can be designed to filter out the noise. In this project, we quantitatively measure and analyze the noise from the photodiode and amplifiers to ensure that the noise component from the photodiode dominates. Designing filters for cleaning up the signals is delayed until a later stage in the project if deemed necessary.

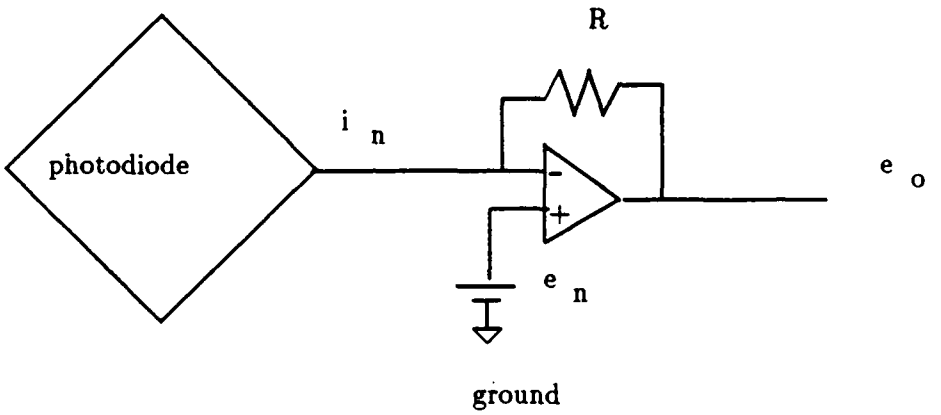



Figure 4.2: Circuit noise analysis

Figure 4.2 shows the schematic diagram of the photodiode/amplifier assembly. There are two noise sources, the photodiode and the amplifier. The vendor's specification states that the equivalent input noise voltage of the amplifier is several orders of magnitude larger than the noise voltage coming from the equivalent input noise current flowing through the resistor $R$. Hence in the study of the noise effect, the amplifier is connected to a noise voltage source at one of the amplifier's two input ports. Since the photodiode is basically a current device, it is represented by a noise current source in the analysis.

In order to measure the noise contributed by these two sources quantitatively, we first detach the photodiode from the amplifier and connect the amplifier inputs to ground. The output of the amplifier thus represents the noise characteristics inherent in the amplifier itself. The frequency spectrum of the output is plotted in Figure 4.3. It can be seen that the noise exhibits $1/f$ or flicker noise characteristics up to 15 kHz. This observation confirms the vendor's specification.

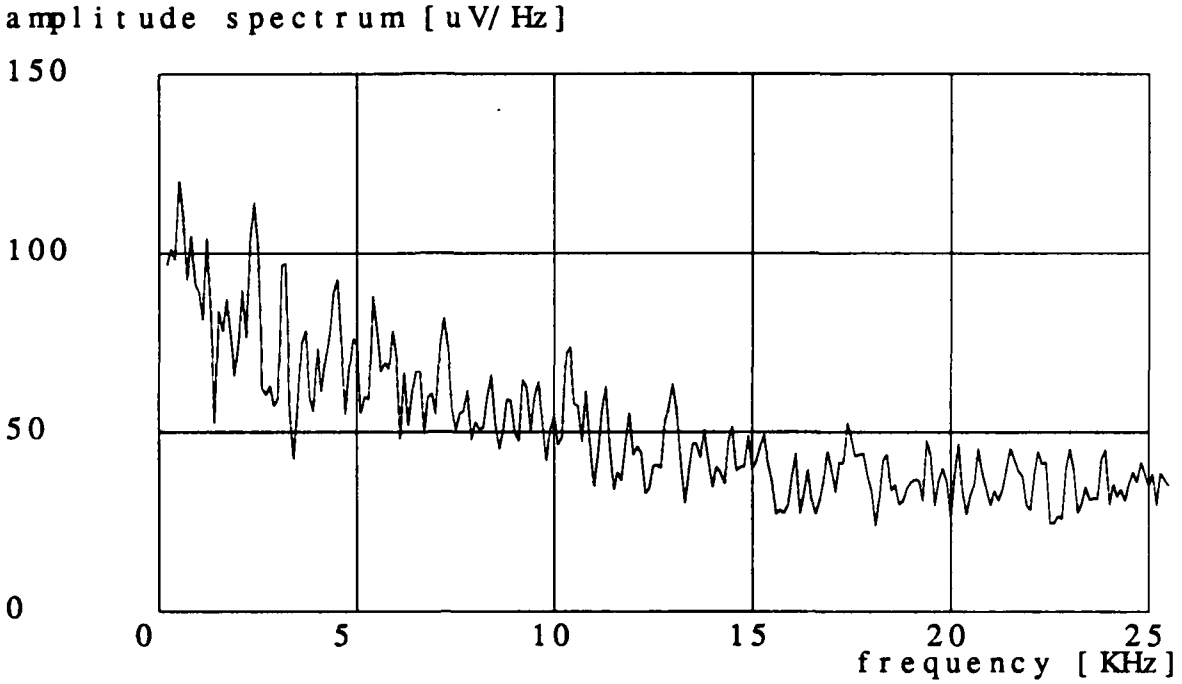



Figure 4.3: Noise spectrum of the amplifier

We then reconnect the photodiode to the amplifier. The current noise from the photodiode is studied. For an op-amp as depicted in Figure 4.2 we have

$$e_o = e_n - i_n R. \tag{4.1}$$

The contribution of the two noise sources at the output is calculated using the following procedure: The power spectrum of a signal $X(t)$ is defined to be the average of the periodograms of the signal, or

$$\text{Power Spectrum of } X = E\left[\frac{1}{T}|\mathcal{F}(X)|^2\right],$$

where $\mathcal{F}$ denotes the Fourier transform and $E(x)$ denotes the expected value of $x$. Assume that there is no correlation between $e_n$ and $i_n$, then the power spectrum of $e_o$ can be shown to be the sum of the spectra of $e_n$ and of $i_n$ weighted by $R^2$:

$$E\left[\frac{1}{T}|\mathcal{F}(e_o)|^2\right] = E\left[\frac{1}{T}|\mathcal{F}(e_n)|^2\right] + E\left[\frac{R^2}{T}|\mathcal{F}(i_n)|^2\right]. \tag{4.2}$$

In Figure 4.4, we plot the power spectrum of the noise signal from the amplifier alone and the power spectrum of the combined noise signal from both the amplifier and the photodiode. According to Equation 4.2, it is clear that the noise from the photodiode dominates.

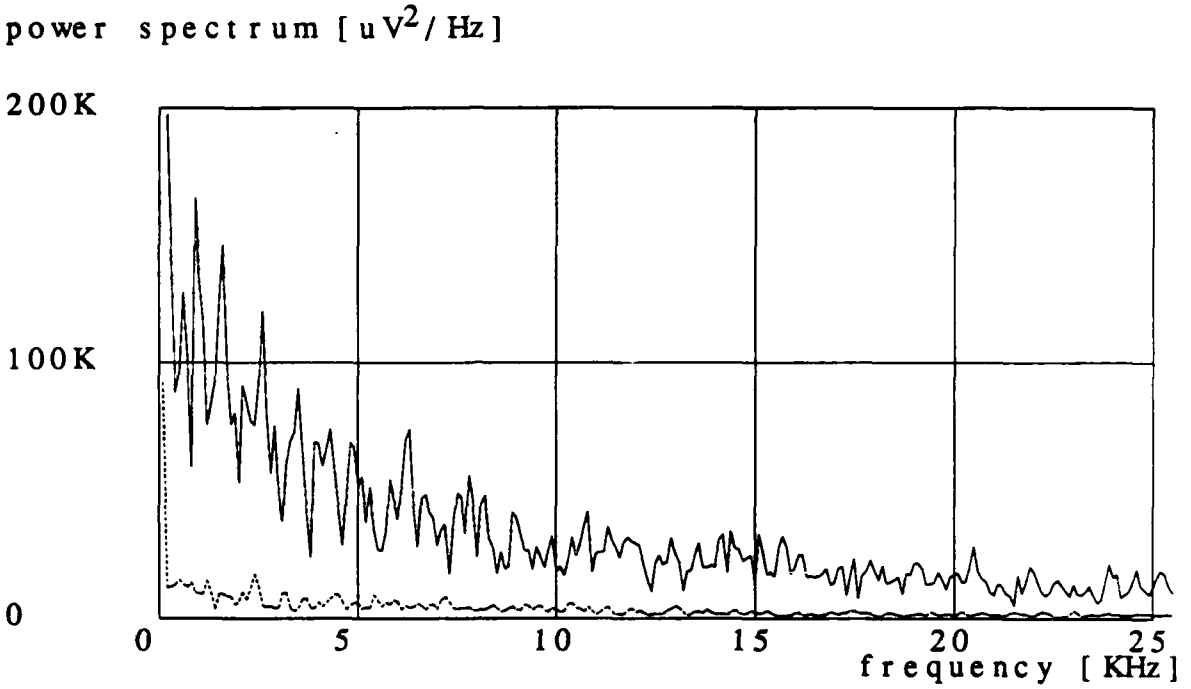



Figure 4.4: Power spectrum of the photodiode (upper graph) and the amplifier (lower graph)

### 4.1.2 Photodiode Resolution and Linearity

The resolution of the photodiode is governed mainly by its signal-to-noise (S/N) ratio. To increase the S/N ratio, either we increase the signal strength or we decrease the noise strength. We cannot do much to decrease the amount of noise inherent in a device unless we choose a different one. In this project, we tested two photodiodes made by United Detector Technology and Hamamatsu, and selected the one with higher resolution which is made by Hamamatsu. However, signal strength can be increased by using high power LEDs, low F-number lenses, or both. The current implementation uses F/1.4, 50-mm television lenses and very high power infrared LEDs. Furthermore, we can perform simple filtering such as integration to smooth out the noise. For example, in our system the signal from the photodiode is integrated to smooth out the high frequency noise.

Figure 4.5 shows the calibration result of the photodiode reading. This result was obtained by placing a LED 3 meters away from the camera. The LED was mounted on a translation stage and moved in the direction parallel to the photodiode surface with a 0.5mm movement each time. If the size of the photodiode is 1 $cm^2$, the 0.5mm movement at 3 meters away corresponds roughly to a 1 part in 1200 resolution. The plot shows a good linear characteristic and confirms our ability to get a working range of at least $(3m)^3$.

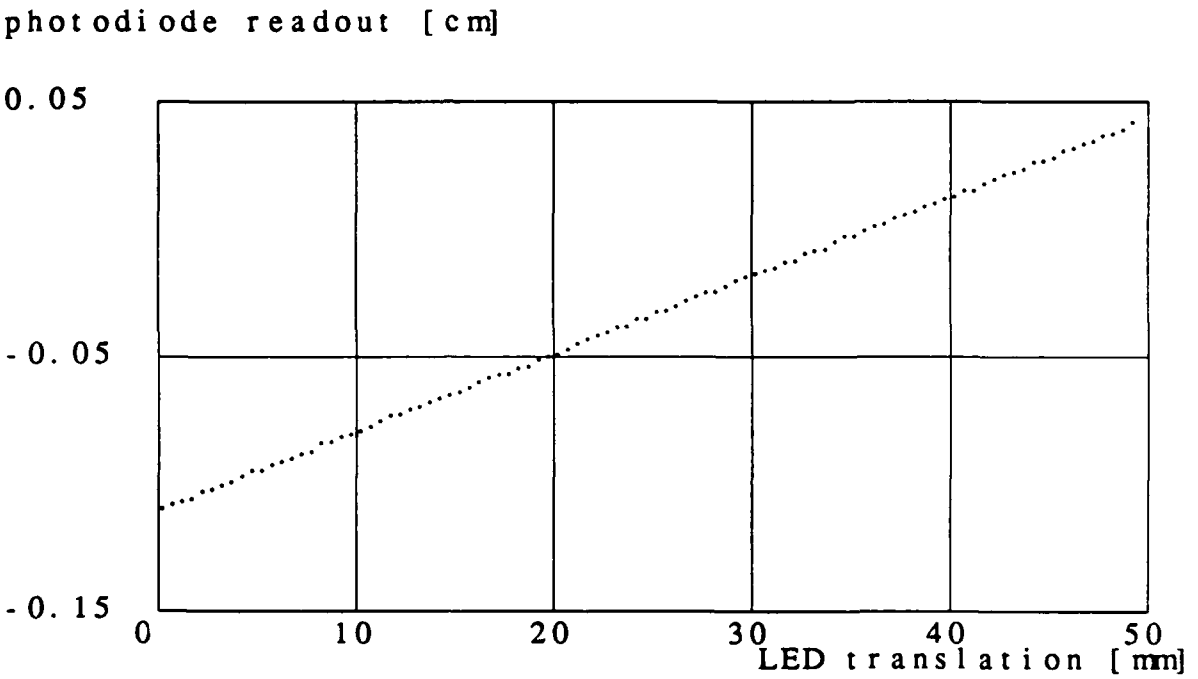



Figure 4.5: Photodiode resolution at 3m away over 20% of the photodiode surface

The surface of the photodiode is not strictly linear. That is, even if the light source moves at a constant velocity along a straight line, the positional readouts from the photodiode will in general not indicate a linear movement. This nonlinear distortion is greatest near the the edges of the photodiode. In order to calibrate the response from the photodiode, an LED was mounted on the pen holder of an x-y plotter, with translational accuracy 0.025 mm. A host computer controlled the movement of the LED on the plotter surface. The movement of the LED traced out a rectangular grid on the plotter surface and the photodiode readings with the LED positioned at the grid junctions were recorded. Figure 4.6 shows this calibration result. The distortion of positional reading at the edges of the photodiode can easily be seen. This effect is partly due to the distortion of the lens and partly due to the nonlinearity of the photodiode. Since the photodiode and the lens will always be used as an assembly, we only calibrate the combined distortion effect in our experiment.

Some post-processing is necessary to compensate the distortion of the reading. A table is created which stores the readings from the photodiode at each grid point. The reading from the photodiode is used as an index into the table to find the closest grid point, and the corrected reading is obtained by linearly interpolating between the nearest grid points.

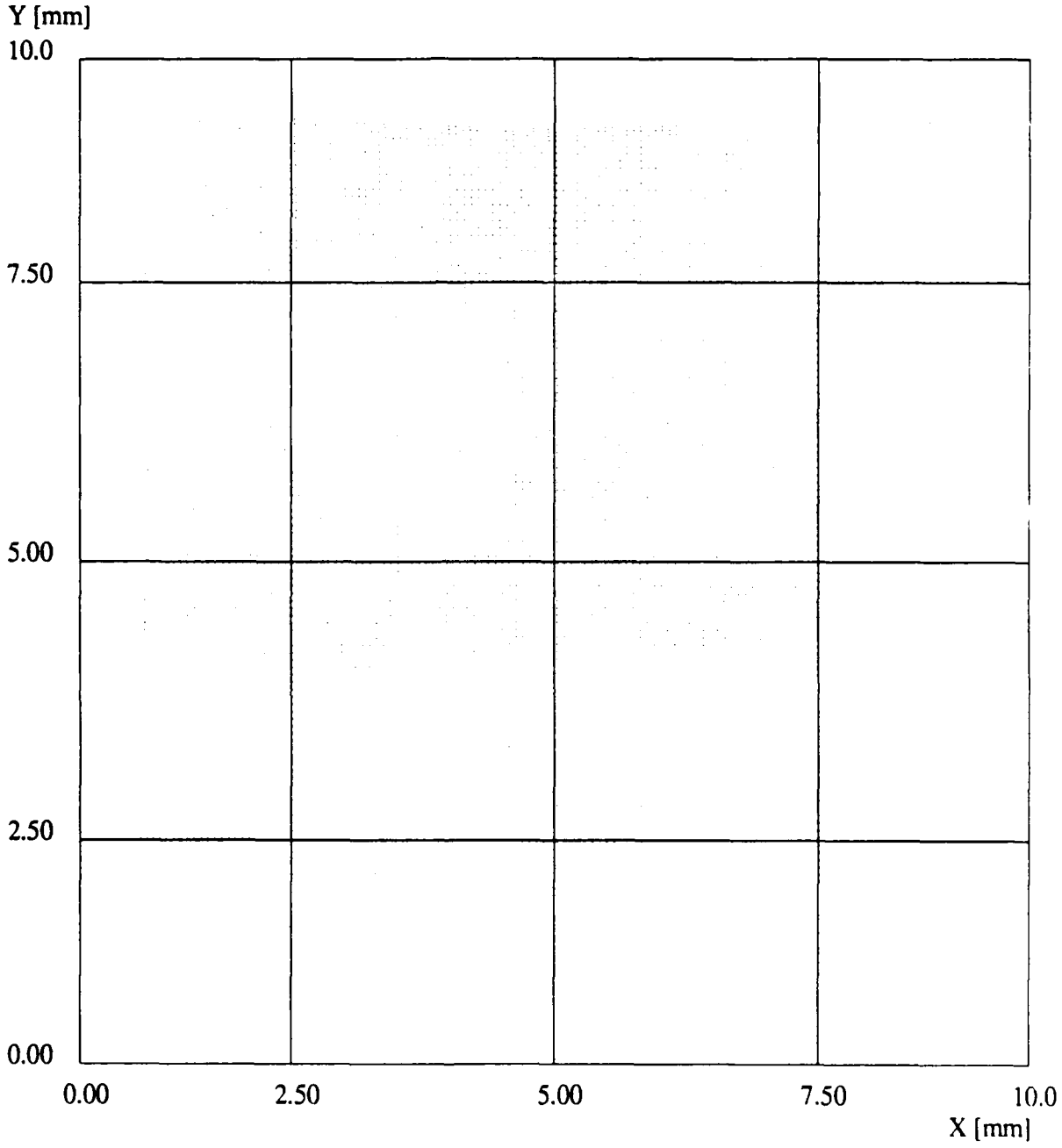



Figure 4.6: Calibration of the photodiode surface and camera lens unit with target LED at 3 meters

## 4.2 LED Circuitry

Infrared LEDs are selected as the beacons of the system. Since the infrared LEDs do not emit visible light, the user will not be distracted by their constant blinking. Infrared LEDs were selected that deliver high output power and a wide emission angle to maximize the working range of the tracker. From a practical point of view, since many such beacons are used, the LEDs should be inexpensive.

The LED used in the system is Seimens' Model SFH487P [Sei85]. It has a flat surface in front, delivers high output power (20mw/sr), has a very wide half power angle of 60° (as shown in Figure 4.7), and is inexpensive (~ $0.71).

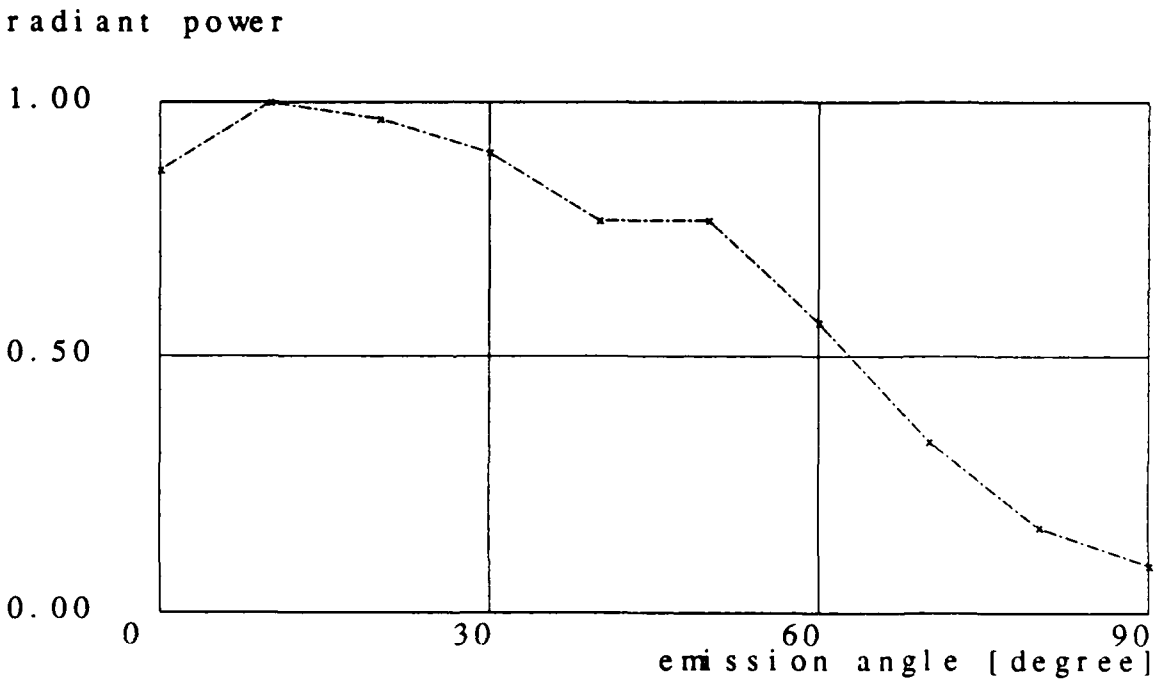



Figure 4.7: Measured LED radiant power spectrum

In the prototype, only three LEDs are used. In a full scale working system, several hundreds or even thousands of LEDs will be used. How to address them efficiently will be an important issue.

## 4.3 Signal Processing Circuitry

Equation 2.1 is the governing equation of the $x$ and $y$ readouts from the photodiode. One straight-forward implementation of the equation would be to use an analog divider and perform the division in the analog domain. However, analog division cannot be implemented with simple linear circuitry using only capacitors, resistors, and amplifiers. Commercial analog dividers suffer poor accuracy and limited working range. Further, several external parameters must be tuned to guarantee satisfactory performance from the divider.

Figure 4.4 indicates that a substantial amount of high frequency noise is present in the photodiode readout. High-frequency noise can be smoothed out by integration. If the transient response of the photodiode is relatively short-lived (the rise time of the photodiode signal is short compared to the integration interval), then during the period of time the LED is turned on, the photodiode readout can be assumed to be a constant except for a negligible portion along the leading and trailing edges of the signal. Hence, we can actually integrate and digitize the four channels, $x^+$, $x^-$, $y^+$, and $y^-$ independently over their individual stable ranges and perform the division digitally. That is:

$$x = \int \frac{x^+ - x^-}{x^+ + x^-} \cong \frac{\int x^+ - \int x^-}{\int x^+ + \int x^-}$$
$$y = \int \frac{y^+ - y^-}{y^+ + y^-} \cong \frac{\int y^+ - \int y^-}{\int y^+ + \int y^-}. \tag{4.3}$$

The integrators are implemented with simple $RC$ elements.

In Equation 4.3, the division operation normalizes the photodiode readout under different signal strengths. Different signal strengths from the photodiode is a result of different light intensities passed through the camera lens. Since we integrate and digitize each signal separately, the normalization is not done in real time any more. To prove that the division operation still compensates for the effect of varying light intensity, we moved the LED along the same path several times but each time with a different luminance from the LED. Figure 4.8 depicts the photodiode readouts with three different LED luminances (different currents flow through the LED). As can be seen, the division still effectively compensates varying luminance as curves of different luminance coincide. This proves the correctness of our new approach. Also note that if the power output from the LED is too low, the S/N ratio decreases and the resolution of the photodiode suffers. This explains the jagged curve in Figure 4.8.

Since noise is an inherent problem when working in the analog domain, we perform all the necessary mathematical calculations in the digital domain. After the analog integration of the signals in the four channels, the signals are sampled and digitized. The subsequent division operation can then be performed by the host in digital domain. The final $x, y$

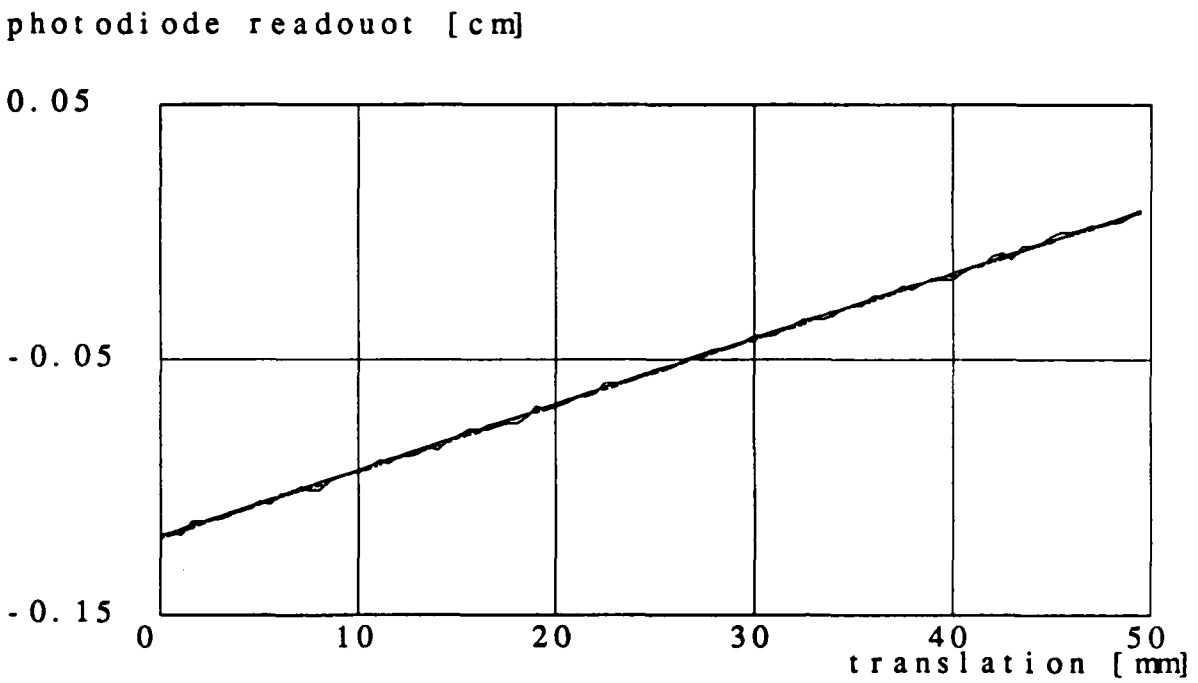



Figure 4.8: Photodiode readout under different luminances (LED current: dashed line: 1A, solid line: 0.75A, jagged line: 0.5A

reading is computed digitally using Equation 4.3.

In order to eliminate the effect of dark current from the photodiode, the background light of the room, and the static offset voltage in the components, the integration is done twice for each channel. First, with the LEDs turned off, we integrate the dark currents of the four channels and digitize them. Then with the LEDs on, we integrate and digitize the signals from the four channels again. We then subtract the first reading from the second to eliminate the constant offset which is induced by the dark current, background light, and the constant offset voltage of the amplifiers, the sample-and-hold (S/H) amplifier and the analog-to-digital (A/D) converter.

Figure 4.9 shows the schematic diagram of the signal processing circuitry. On the left is the integrator. The output from the integrator is fed to the S/H amplifier and then to the A/D converter. Since the integrator is an $RC$ circuit, we need to specify the values of $R$ and $C$. These values are decided as follows: After each integration, the integrator must discharge before next integration can take place. So there must be an inactive time period between two successive integrations. Currently, the integration lasts for 100 $\mu$sec, which corresponds to the longest period of time the LEDs can be left on at high current ($\sim$ 1 A), according to the vendor's specification. The time $t_r$ between two successive integrations is 20

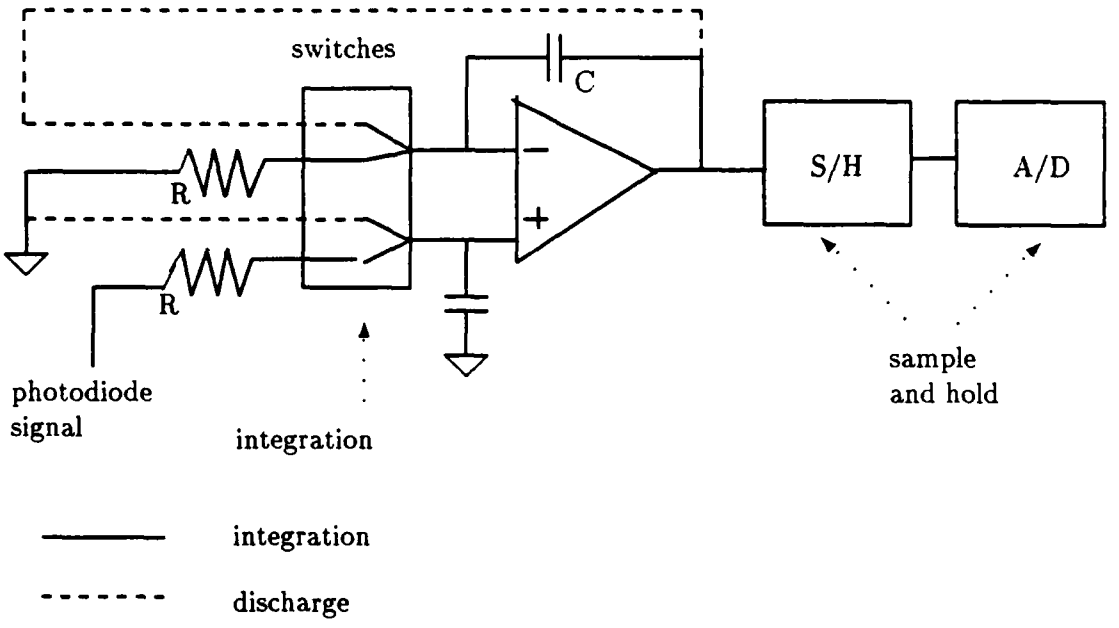



Figure 4.9: Signal Processing Circuitry

$\mu$sec. Usually, between two successive integrations, the host computer has to read back the data from A/D's, which will definitely take longer than 20 $\mu$sec, so the integration circuitry will always have more time to discharge. A rule of thumb states that the time constant $\tau = RC < .1 * t_r$ allows the capacitor to discharge to 0.01%. So $\tau = 0.2 * 10^{-6}$, where $R$, the "on" resistance of the switch, is about 200 $\Omega$. Hence, we have $C = 10^{-8}$F = 10nF. Furthermore, the photodiode output signal saturates at 10 V, this maximum output from the photodiode should correspond to the maximum working range of the A/D converter, which happens to be 10 V, so we have:

$$C = 10^{-8} = \frac{Q}{V} = \frac{\int i\, dt}{V} = \frac{\frac{V}{R}10^{-4}}{V} = \frac{10^{-4}}{R}, \tag{4.4}$$

and we compute $R = 10k\Omega$.

## 4.4 Controller

From Figure 4.9, we observe that the signal processing circuitry is controlled by several control signals: the *integration* signal which controls the the integration and discharge, the *LED* signal which switches the LEDs on and off, and the *hold* signal which toggles the S/H

and A/D between the sample and hold modes. Since the host computer determines which LEDs to flash next, the signal processing circuitry has to wait for the host computer to supply the LED addresses. The host computer, after computing the addresses of the LEDs to be flashed next, transmits a *go* signal and the LED address to the controller. The controller then generates appropriate control signals to the circuitry to start the cycle of integrating dark current, turning on the LEDs, integrating the signals, and digitizing the signals. The timing diagram of the control signals is shown in Figure 4.10.

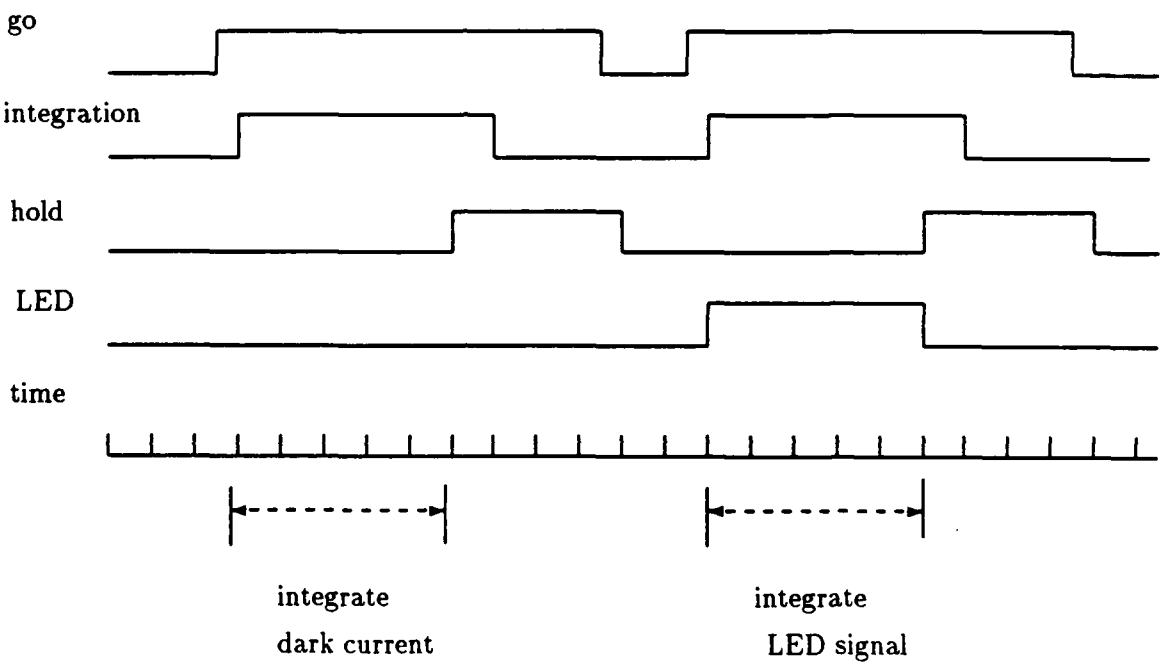



Figure 4.10: Control signals

The controller itself is a finite state machine implemented using an erasable programmable logic array (EPLA). A laboratory generator supplies the clock signal to the controller. Each clock period is 20 $\mu$sec long. This value is selected based on the LED duty cycle and the PLA capacity. Since there are only a fixed number of product terms in the controller chip, we cannot afford too many states. As can be seen from Figure 4.10, integration takes 6 states which is 120 $\mu$sec. The S/H circuit holds the value at the end of the 100 $\mu$sec. The integrator starts to discharge after 120 $\mu$sec while the computer reads the results back from the A/D's. During all these cycles, the computer holds the *go* signal high. When data have been transmitted from the A/D's to the host, the host lowers the *go* signal to prepare for the next iteration. Note that the *led* signal goes high only once in two successive integrations.

The whole circuit was built on a breadboard. Two power supplies were used. The output from the photodiode shows an increased noise at low frequency as in Figure 4.4, so it might be possible at a later stage of this project to optimize the circuitry using filters to filter out the noise. A more sophisticated controller needs to be built to control LEDs, and the circuitry should be constructed on a more robust platform (printed circuit board) which can be plugged directly into the host computer.

## 4.5 Computer Interface

Currently, a DRV-11 parallel interface card is used to connect the circuitry to the host computer, which is a DEC $\mu$VAX-II. The DRV-11 provides sixteen inputs, sixteen outputs, three control lines, and three status lines. Twelve of the sixteen input lines are used to read data from the A/D's. Four output lines are used to address one of the 12 A/D's for output, leaving twelve output lines to address the LEDs. Hence, a total of 4096 LEDs can be directly addressed. One control line is used to send the *go* signal to the circuit. The circuit returns two status signals on the current status of the controller. A block diagram of the interface is shown in Figure 4.11.

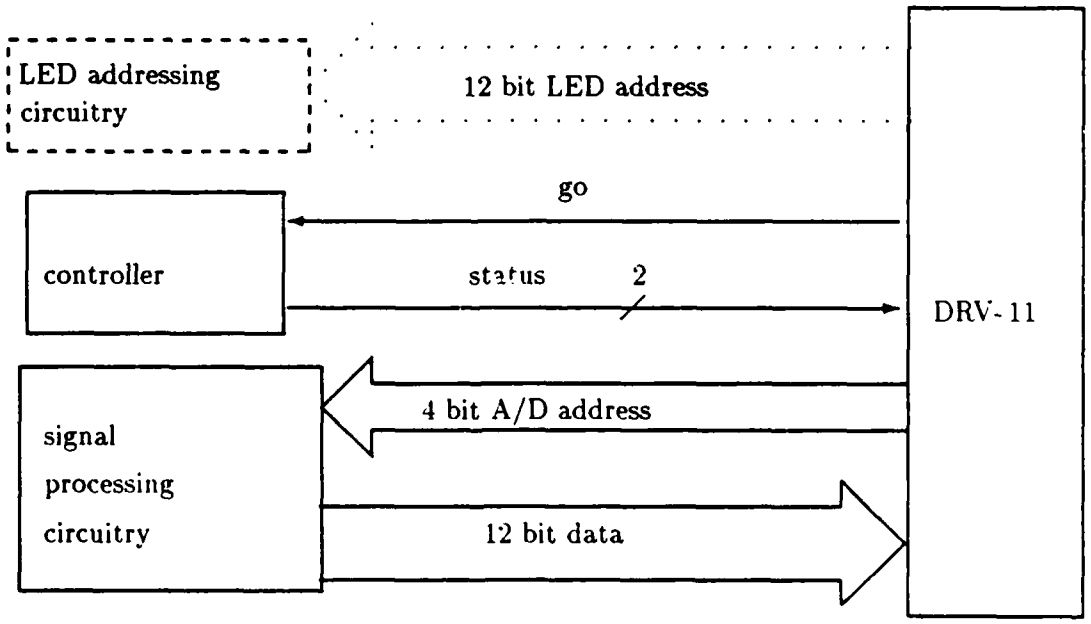



Figure 4.11: Computer Interface

The host computer (DEC $\mu$VAX-II) runs the Ultrix operating system. The problem

with multi-tasking operating systems like Ultrix for real-time control is that the operating system might introduce additional delay due to its time sharing characteristics. Each time when a process running on the computer reads data from the circuitry, the process is temporarily suspended while the operating system steps in to perform the I/O operations. In order to avoid unnecessary delay, a device driver was written in such a way that the communication between the process and the circuitry is handled in the most efficient way. That is, the communication between the process and the circuitry is done in just one handshake. The process prepares the three LED addresses and sends them as one packet to the operating system. The device driver, upon receiving the LED addresses, generates the *go* signal to the controller which starts the integration of dark currents. The dark currents from all three cameras are integrated and then digitized simultaneously by 12 independent S/H's and A/D's, and stored while the device driver sends control signals to flash the LEDs. The images of the LEDs are recorded by the cameras, and the data from each camera are digitized and stored. After the data from all three cameras have been read back into the computer memory, the device driver returns to the application program with the values of both dark currents and LED signals for all three cameras.

# Chapter 5

# System Performance Evaluation

In this chapter, we describe the measurement of the performance of the prototype. To prove the correctness of our design, we have conducted experiments to evaluate the three most important performance parameters, namely speed, range and resolution. In the following sections, we describe the experimental procedures and tabulate the results we obtained.

## 5.1 Speed and Lag

The process of tracking consists of two phases: the *sampling* phase and the *computing* phase. The sampling phase starts when the host sends the *go* signal to initiate the controller cycles and ends when data are acquired from all A/D channels. The computing phase starts immediately afterward to calculate the 6D position and orientation of the camera assembly using the algorithm presented in Section 3.3.4.

The current prototype was implemented on a $\mu$Vax-II workstation. Our data on the $\mu$Vax-II indicate that the time spent on sampling is much smaller than that spent on computing, or the tracking process is computationally intensive. Without doing any position computation, the pure sampling rate of the system can be as high as 1500 Hz. If complete cycles of sampling and computation were performed, a maximum of about 25 updates per second is what can be achieved on a $\mu$Vax-II.

We can see that even though the update rate of this prototype is only comparable to that of most commercial trackers, the performance of the system can be improved significantly using a fast host machine. We estimated the update rate of the tracker—using a number of different computers as hosts—by running Church's algorithm on them. One should note that since the sampling circuitry is hard-wired to the $\mu$Vax-II, the time spent on the sampling phase on different machines cannot be estimated this way.

| machine | speed of computation (updates/sec) | speed (+ sampling) (updates/sec) |
|---|---|---|
| Sun 3/50 | < 1 | < 1 |
| Sun 3/60 | 7 | 7[1] |
| Sun 4 | 86 | 82[1] |
| $\mu$Vax-II | - | 25[2] |
| $\mu$Vax-3200 | 70 | 69[1] |
| DECstation 3100 | 251 | 215[1] |

Table 5.1: Speed comparison on different machines (1: simulated, 2: measured)

We ran the same Church's algorithm for position estimation on several Sun workstations: 3/50, 3/60, and Sun 4, and DEC workstations: $\mu$Vax 3200 and DECstation 3100. Table 5.1 shows the average update rate on different machines. The second column $C_2$ shows the result of running Church's method on different machines, while the third column $C_3$ takes into account the time spending on the sampling phase by adding about 700 $\mu$sec (1/1500) for each update. That is, $C_3 = 1/(1/C_2 + 0.0007)$.

The result seems to indicate that a DECstation 3100 currently best suits our application. Since the tracking process is slowed down by the time-sharing nature of the Unix operating system, single user personal computers such as the IBM-PC or Macintosh might be worth investigating as host computers.

Since the computer controls the both the sampling and the computing phase, and one report is sent out after each complete sampling and computing cycle, the lag of the system is no longer than one update period. Thus, the latency of a 3100 based system is about 4.65 milliseconds (1/215).

In summary, although the prototype provides a slow update rate, it is possible to select a different host machine to achieve a speed for near real-time performance with little lag.

## 5.2 Range and Accuracy

Range and accuracy are tightly related. As the distance from the camera to the beacon increases, the signal-to-noise ratio decreases, hence, the accuracy degrades. As observed in our experiment (Figure 4.5), a photodiode can provide a 1 part in 1200 resolution at about 3 meters in the worst case, which is much larger than the working range of the Polhemus tracker.

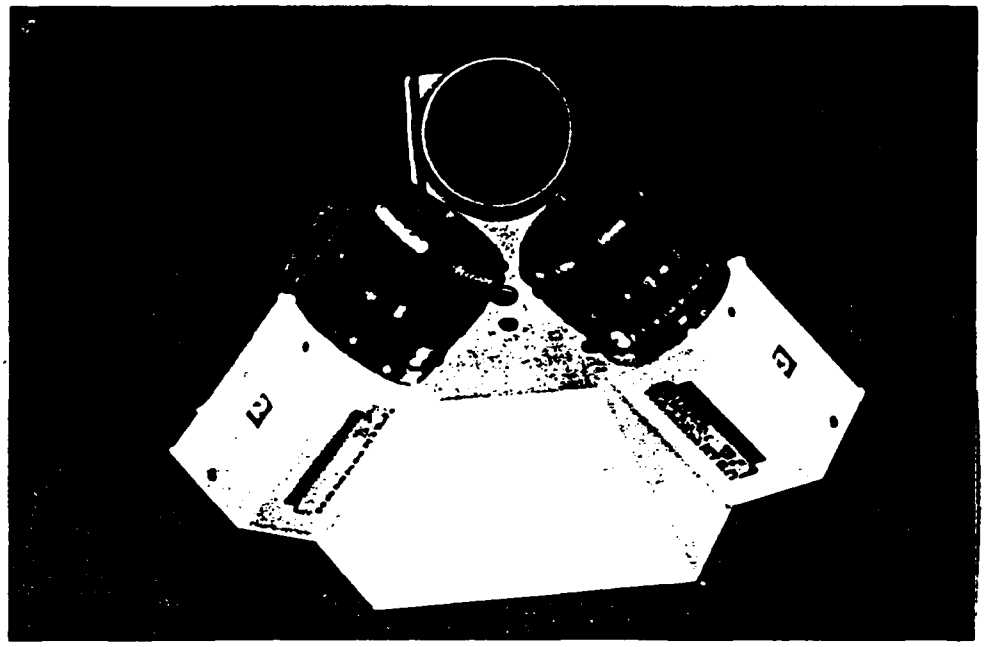


Figure 5.1: Three photodiode cameras mounted on a helmet (size: 8x7x4.5 $in^3$)

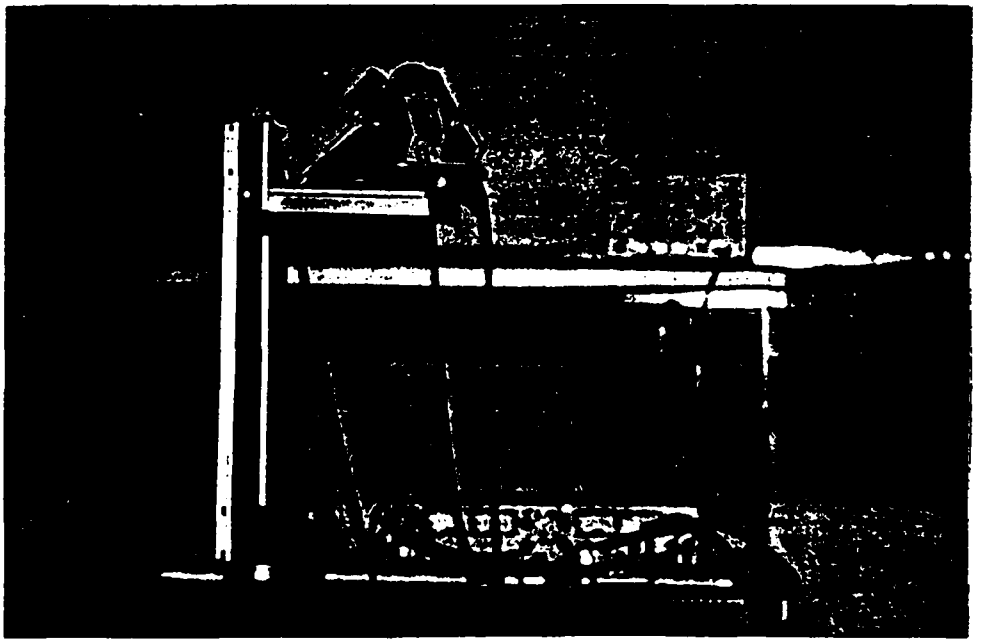


Figure 5.2: The bench-top prototype

To prove that the prototype can achieve such a working range, we conducted the following experiments: A helmet which can mount three cameras was constructed (Figure 5.1). This camera assembly was positioned on a mounting device which provides six degrees of freedom in motion that can be measured precisely. Three beacons were mounted on tripods, with one beacon inside the field of view of each camera. The distances from the cameras to the beacons were about 3 meters. Figure 5.2 depicts the experimental setup. Translational resolution was measured by moving the helmet through 1mm increments along one of the axes, and rotational resolution was measured by rotating the helmet in 0.1° increments about one of the rotational axes. Figures 5.3 and 5.4 graphically depict results from our experiments. From these figures, we conclude that the prototype can provide a 0.1° resolution in rotation and about 2 mm resolution in translation. This extremely high sensitivity in detecting both the rotational and translational motions can be attributed to the use of the inside-out tracking paradigm and the multiple-view concept. Our results clearly demonstrate the superiority of the new design.

Also from Figures 5.3 and 5.4, we see that the tracker can detect rotational movement better than translational movement according to the scales we used (0.1 degree for rotation and 1 mm for translation). The reason of the discrepancy is due to the fact that with inside-out tracking, a small rotational movement induces larger image shift on the photodiode surface than a small translational movement at large object distances. For example, with the 50 mm lens we use, a 0.1° rotation shifts the light source image by 0.087 mm, while a 1 mm translation produces only about 0.025 mm image shift, when the light source is 2 meters distant.

Another factor which affects the performance of the tracker is the jittering effect. Due to the noise inherent in the analog circuitry, the report from the tracker might oscillate slightly even if the user stands perfectly still. Large jittering distracts the user more than poor resolution. Again, jittering depends on the signal-to-noise ratio. To study the jittering effect in our system, we repeatedly sampled the tracker output while the helmet remained stationary at about 3 meters from the beacons. A typical result with five hundred samples shows a maximum of 0.05° jittering in orientation and 1.5 mm jittering in position (observed peak to peak noise). Since the sampling rate of the tracker is very high, jittering could be smoothed out by repeated sampling and averaging.

Finally, a demo program was developed for this bench-top prototype. The user moves the helmet on the mounting stage by hand. The tracker reports the 6D position of the helmet. These reported positions are sent through ethernet to either the Pixel-Planes 4 machine or a X window based workstation for image generation. The video signal from the Pixel-Planes 4 machine is transmitted back and displayed on a television so the user can view the movement directly. While several meters are displayed using X windows to quantitatively

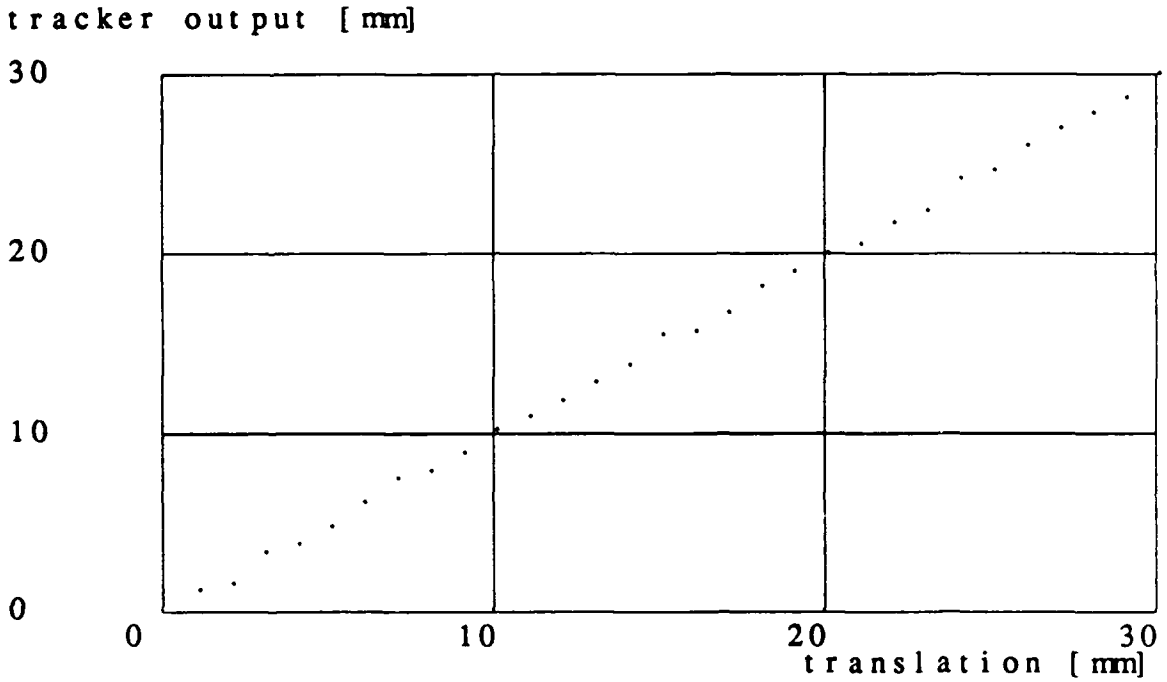



Figure 5.3: Translational sensitivity of the prototype

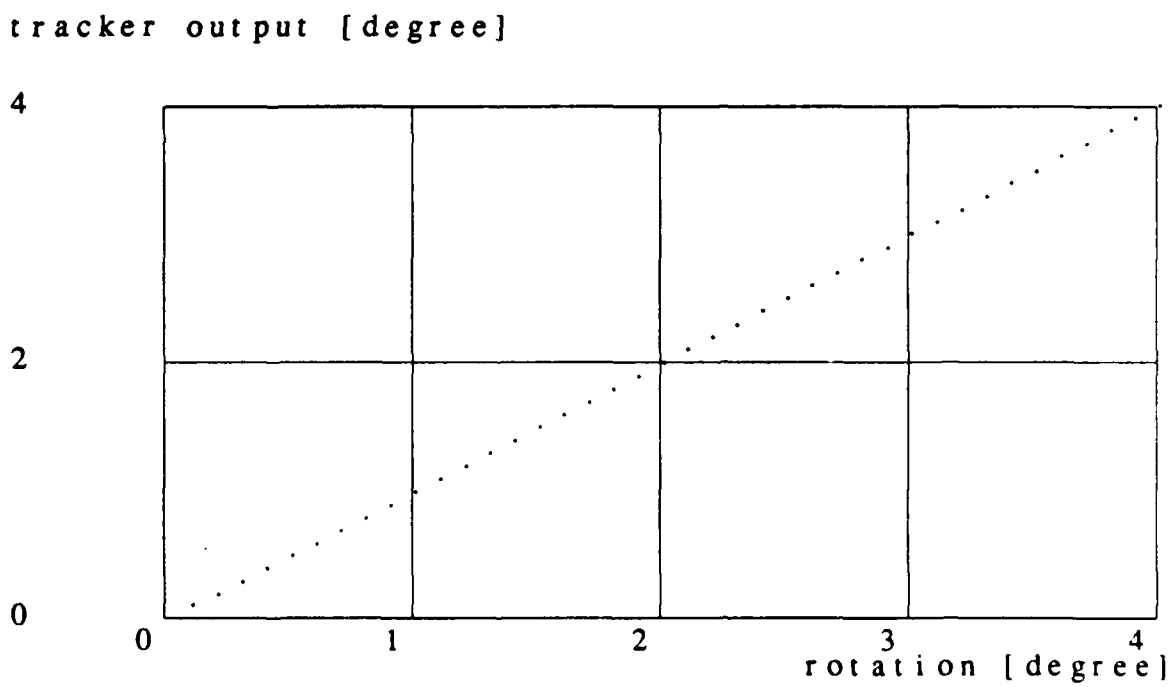



Figure 5.4: Rotational sensitivity of the prototype

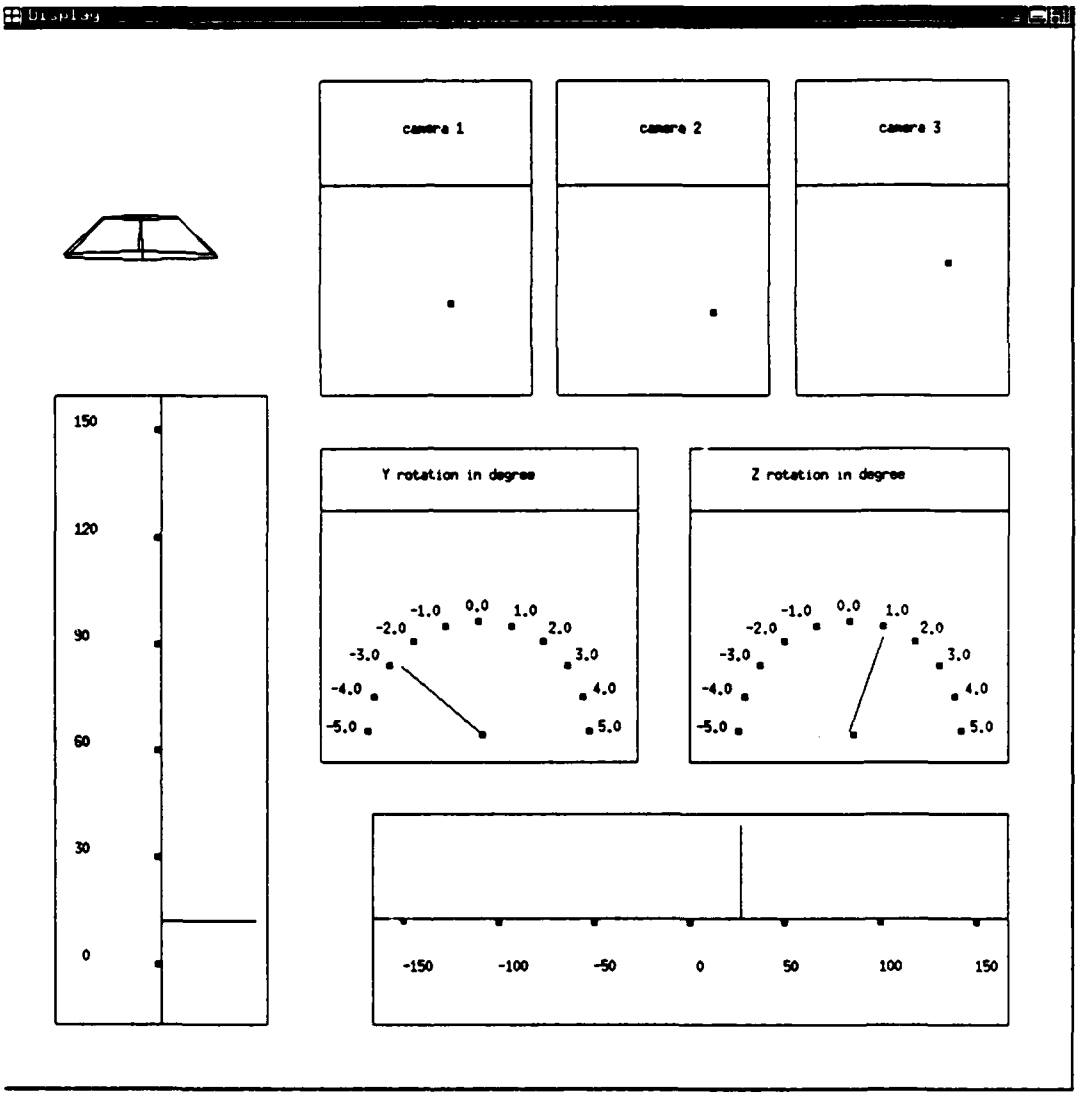



Figure 5.5: The demo program displayed using an X window

show the amount of movement in different directions (Figure 5.5). The three boxes on the top row in Figure 5.5 show the light source images on the photodiode surfaces, so the user can make sure the movement of the prototype will not cause the cameras to lose the LEDs.

## 5.3 Camera Self-calibration

We have also tested the concept of camera self-calibration (Section 3.4.2) using this prototype. The basic idea behind camera self-calibration, as mentioned in Section 3.4, is to figure out the beacon position by back projecting a line through the camera's nodal point and the beacon's image position. The beacon position must be somewhere along the back-

projected line. Since the beacons will be mounted on the ceiling, that position information gives us another constraint to pinpoint the beacon position in 3D.

The prototype consists of only three LEDs, so in order to test this idea, we had to design an experimental procedure to simulate many LEDs on the ceiling. The experiment was carried out by mounting three LEDs on tripods so as to be easily moved. Since LEDs are mounted on tripods, the height of each LED can be measured and remains fixed as the tripod is moved. The height information corresponds to the ceiling height in the real case. We started by setting the helmet at a fixed position, and then let the tracker automatically calibrate the positions of the LEDs by the method described above. The camera unit was moved until one of the LED was about to be out of sight. At this point, we manually moved that tripod to a new location back toward the center of the camera's field of view, but without measuring where exactly we moved it. We then let the system calculate the new position of the LED. Thus we could test the idea of camera self-calibration with only a limited number of LEDs Figure 5.6 shows what we did.

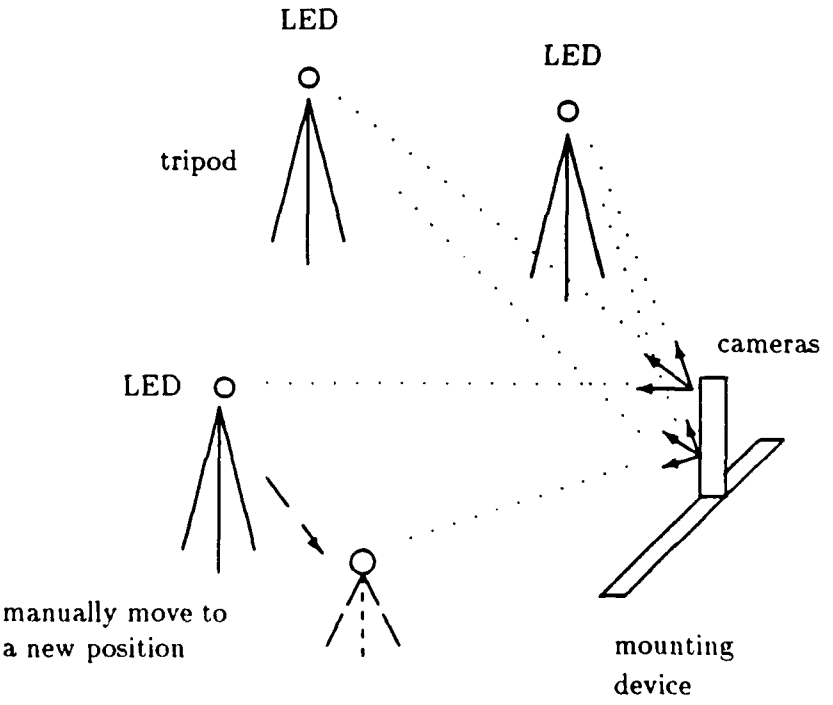



Figure 5.6: Camera self-calibration

In a particular experiment as shown in Figure 5.7, we rotated the tracker horizontally and measured the physical movement by reading the scale from the rotation stage the tracker is mounted on. The output from the tracker was then plotted against the true movement.

The vertical dashed lines show places where new LEDs are calibrated and included as reference. From the experiment, we show that camera self-calibration is a flexible idea. The tracker can effectively determine the new LED positions, then use those as new references to continue the tracking process. Also, since the sampling rate of the system is very fast, the operation introduces little overhead.

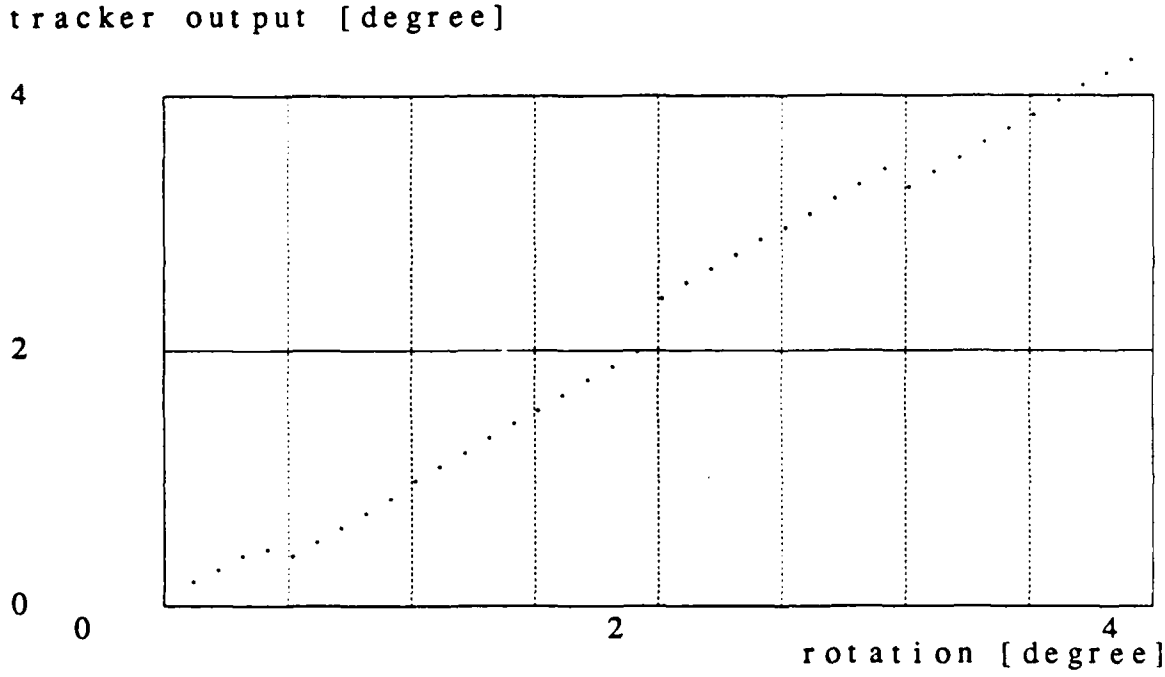



Figure 5.7: Experimental result of camera self-calibration

# Chapter 6

# Discussion

In summary, this thesis presents a new design concept for a 6D position tracking device. This new tracker can be employed in a head-mounted display system to track the movement of the user's head. Our prototype system uses an inside-out tracking method, and was designed and built using off-the-shelf components for easy duplication. In this final chapter, the steps to build a full scale system are outlined. We also point out several interesting related research topics.

## 6.1 A Full Scale System

The following research problems should be addressed before we attempt to build a full scale system.

**Circuit noise analysis** More studies need to be done to minimize the noise effect of the photodiode and the circuitry for signal processing. The noise effect of the photodiode is the major limiting factor on the resolution of the system and produces the jittering effect which distracts the user. A suitable filter may need to be designed and built to clean up the noise should jittering become a problem.

**New circuit design** Currently, the signal processing circuitry is built on a breadboard. Our experience indicates that using a breadboard has many problems and is not reliable for day-to-day operations. Broken and loose wiring is common with a breadboard. This causes the system to cease functioning for no apparent reason, and a lengthy debugging process has had to be repeated many times. A printed circuit board is definitely needed at a later stage of this project.

**Interface and host computer selection** The current host ($\mu$Vax-II) has limited computational power and is not fast enough for real-time tracking applications. Furthermore, the UNIX time sharing system does not allow a process to communicate directly with the tracker. Real-time 6D position tracking calls for a stand-alone facility, such as a microcomputer or a floating-point chip (e.g. Weitek), to be used as a dedicated processor for the tracker. The DRV-11 interface should be removed from the working system, with the circuit board plugged directly into the host.

**LED addressing** In the final working system, thousands of LEDs will be used as beacons. As mentioned before, how to address them efficiently will need to be studied. Ron Azuma, who recently joined the project, is investigating several different alternatives.

**LED placement** In order to make the system more portable, LEDs should be mounted on large circuit boards and the boards then affixed to the ceiling. Boards can be moved relatively easily and configured for different room shapes in this way. Since the system is intended to function in a normally lit room environment, the interference of the ambient light has to be taken into consideration. Either wavelength filters have to be used in conjunction with the lenses to filter out the room light, or the room lights have to be regulated so that they do not emit infrared energy and thus interfere with the photodiode. Currently, the working environment is installed with normal fluorescent lights which emit little energy in the infrared band, and a wavelength filter is used so they do not interfere with the photodiodes.

**Size and Weight** As mentioned, the whole camera assembly weighs a little less than 1 kg. The weight of the helmet and the camera assembly will restrict the movement of the user. Hence, how to reduce the size and weight of the camera assembly is important. The holographic lens described in Section 3.3.3 deserves careful consideration. The holographic lens can shrink the weight of the current prototype down to around 250 grams, and greatly reduce its size. Then the tracker can be positioned somewhat behind the user's head to act as a counter balance the LCD TV sets mounted before the user's eyes. This should lower the center of gravity of the head-mounted system and reduce the moment of inertia of the EyePhone system we use.

## 6.2 Further Research

The final system will use hundreds of LEDs in a room size environment. How to figure out which LEDs should be turned on to guide the tracking process is a nontrivial problem. The LEDs selected depend on the current head position and current field of view of the cameras. How to make the selection efficiently will be important to maintain a high system performance. One solution is to encode the spatial relationship among the LEDs in

a large table. When one LED is out of sight, its nearest neighbor, which is most likely to be the next candidate, can be easily located by a table lookup. A table is desirable since it can be easily changed for different working configurations.

With head-mounted display systems, it is possible to track and report the user's current head position. In such systems, one also allows the user to interact directly with the virtual objects in the environment. For example, the user may use his/her hand to move virtual objects around. Hence, a way to track hand gestures should be studied. In our current head-mounted display system, the user can manipulate virtual objects in several different ways. For examples, we have put a Polhemus inside a pool ball with two buttons added. The user holds the pool ball, and the Polhemus inside the pool ball tracks the 6D position and orientation of user's hand. Hand gestures are encoded using the two buttons. More numerous hand gestures can be distinguished by a Dataglove. However, the Dataglove does not provide any positional information. We can certainly put a Polhemus sensor on a Dataglove to get both position and orientation. But the ideal solution would be to let the system figure out the hand gesture automatically without putting anything on the user's hand. It was suggested by Professor Fuchs that a 2D optical sensor mounted on the helmet can track the user's hand movements in a reasonable working range, and hand gestures can be derived by observing the image shift between frames. This will be an interesting research project.

Finally, even though the system considerably out-performs the Polhemus in speed, range, and accuracy, it will *not* replace the Polhemus! The reason is the complexity involved in setting up the system. The environment will need a lot of changes. The room light has to be regulated and the ceiling has to be totally replaced. One way to alleviate this problem is to incorporate other technologies (e.g. accelerometers) to work with the tracker so the tracker doesn't have to be active all the time. The tracker might just provide reference information occasionally when it can see enough beacons, and the system can correct accumulated drift in its output at these times.

The line-of-sight constraint of the current system restricts the user's movements. In order to overcome this limitation, Gary Bishop's Self-Tracker idea merits further research. It also uses the inside-out tracking paradigm, and does not require any beacons except for recalibration. If we can overcome the size and weight problem of the Self-Tracker, and make sure it can provide adequate resolution in a unregulated environment, it might prove to be a good solution to the tracking problem.

# Bibliography


[BF84] T. G. Bishop and H. Fuchs. The self-tracker: A smart optical sensor on silicon. In *Proceedings Conference on Advanced Research in VLSI.* MIT Press, 1984.

[Bis84] T. G. Bishop. *Self-Tracker: A Smart Optical Sensor on Silicon.* PhD thesis, U. of North Carolina, Chapel Hill, NC, 1984.

[Bis87] T. G. Bishop. Electronic mail from Gary Bishop to the Head-mounted Display Group at UNC, 1987.

[BH80] S. D. Blostein and T. S. Huang. Estimating 3-d motion from range data. In *The 1st Conf. on A.I. Applications*, 1980.

[Bro83] R. G. Brown. *Introduction of Random Signal Analysis and Kalman Filtering.* John Wiley and Sons, New York, NY, 1983.

[Bur73] R. P. Burton. *Real-Time measurement of Multiple Three-Dimensional Position.* PhD thesis, U. of Utah, Salt Lake City, UT, 1973.

[CHB+89] J. C. Chung, M. R. Harris, F. P. Brooks Jr., H. Fuchs, M T. Kelley, J. W. Hughes, M. Ouh-Young, C. Cheung, R. L. Holloway, and M. Pique. Exploring virtual worlds with head-mounted displays. In *Proceedings SPIE Conference, Nonholographic True Three-Dimensional Display Technologies*, Los Angeles, CA, Jan. 1989.

[Chu45] E. Church. Revised geometry of the aerial photograph. In *Bulletin of Aerial Photogrammetry*, volume 15. Syracuse University, 1945.

[Edh30] E. Dehn. *Algebraic Equations.* Columbia University Press, 1930.

[Eal57] J. T. Eallmark. A new semiconductor photocell using lateral photo-effect. *Proceedings IRE*, 45, 1957.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of ACM*, 24(6), 1981.

[FBP+82] H. Fuchs, F. P. Brooks Jr, S. M. Pizer, V. L. Chi, and J. W. Poulton. A 3-d visual environment for human-machine interaction. Proposal to the National Sceience Foundation, 1982.

[FGH+85] H. J. Fuchs, J. Goldfeather, J. P. Hultquist, S. Spach, J. Austin, F. P. Brooks Jr., J. Eyles, and J. Poulton. Fast spheres, textures, transparencies, and image enhancements in pixel-planes. *Computer Graphics*, 19(3), 1985.

[FJ77] H. J. Fuchs and B. Johnson. A system for automatic acquisition of three-dimensional data. In *AFIPS Conference Proceedings*, 1977.

[Gan84] S. Ganapathy. Decomposition of transformation matrices for robot vision. In *Proceedings of the Frist International Conference on Robotics*, 1984.

[Hal89] J. H. Halton. On the geometry of a general three-camera headmounted system. Technical Report 89-019, U. of North Carolina, 1989.

[Ham85] Hamamatsu. *The literature on position sensitive detector.* Hamamatsu Photonics, Hamamatsu City, Japan, 1985.

[Ham83] R. W. Hamming. *Digital Filters.* Prentice-Hall, Englewood cliffs, NJ, 1983.

[HGP81] H. Herbst, H.-P. Grassl, and H.-J. Pfleiderer. Experimental autofocus system for lens shutter cameras. *IEEE Journal of Solid-State Circuit*, SC-17(3), 1981.

[Hol87] R. Holloway. Head-mounted display technical report. Technical Report 87-015. U. of North Carolina, 1987.

[Nor88] Northern Digital. *Trade literature on Optotrak - Northern Digital's Three Dimensional Optical Motion Tracking and Analysis System.* Northern Digital Inc., Waterloo, Ontario, Canada, 1988.

[Rea63] J. A. Reagan Company Inc. *Handbook of Operational Amplifier Applications.* Burr-Brown Research Corporation, 1963.

[Kil76] P. J. Kilpatrick. *The Use of a Kinesthetic Supplement in an Interactive Graphics System.* PhD thesis, U. of North Carolina, Chapel Hill, NC, 1976.

[LO74] L. E. Lindholm and K. E. T. Oeberg. An optoelectronic instrument for remote on-line movement monitoring. *Bioelemetry*, 1, 1974.

[Lyo85] R. F. Lyon. The optical mouse, and an architectural methodology for smart digital sensors. In *VLSI Systems and Computations.* Computer Science Press, Rockville, MD, 1985.

[Nol71] M. A. Noll. *Man-machine Tactile Communication.* PhD thesis, Polytechnic Institute of Brooklyn, Brooklyn, NY, 1971.

[Pol80] Polhemus Navigation Sciences Division. *3Space Isotrak User's Manual.* McDonnell Douglas Electronics Company, 1980.

[PP86] M. A. Penna and R. Patterson. *Projective Geometry and Its Applications to Computer Science.* Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[RBS+79] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems*, AES-15(5), Sep. 1979.

[PW83] S. M. Pizer and V. L. Wallace. *To Compute Numerically.* Little, Brown, and Company, Boston, MA, 1983.

[Rob66] L. G. Roberts. The lincoln wand. In *FJCC*, Washington, D.C., 1966. Spartan Books.

[SD87] D. D. Stearns and R. A. David. *Signal Processing Algorithms.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Sei85] Seimens. *Trade Literature on GaAIAs infrared emitters.* Seimens, Cupertino, CA, 1985.

[Sut65] I. E. Sutherland. The ultimate display. In *Proceedings IFIP Congress*, 1965.

[Sut68] I. E. Sutherland. A head-mounted three dimensional display. In *FJCC Conference Proceedings*, 1968.

[Tei88] M. Teitel. Personal communication with Mike Teitel from VPL, 1988.

[TM84] J. E. Tanner and C. Mead. An correlating optical motion detector. In *1984 MIT Conference on Very Large Scale Integration*, 1984.

[Tsa87] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3, Aug. 1987.

[Uni81] United Detector Technology. *Trade literature on OP-EYE optical position indicator.* United Detector Technology Inc., Santa Monica, CA, 1981.

[Vic74] D. L. Vickers. *Sorcerer's Apprentice: Head-mounted Display and Wand.* PhD thesis, U. of Utah, Salt Lake City, UT, 1974.

[VPL89] Visual Programming Languages Research Inc. *Trade literature on EyePhone.* Visual Programming Languages Research Inc., Redwood City, CA, 1989.

[Wol74] H. J. Woltring. New possibilities for human motion studies by real-time light spot position measurement. *Bioelemetry*, 1, 1974.

# Appendix A

# Error Formulas

In section 3.3, we derived errors caused by limited photodiode resolution and beacon placement uncertainty. Those formulas are derived under a restricted condition. Here we formulate those errors for the general case. If the coordinates of images $A', B'$ are $(x_a, y_a, f)$

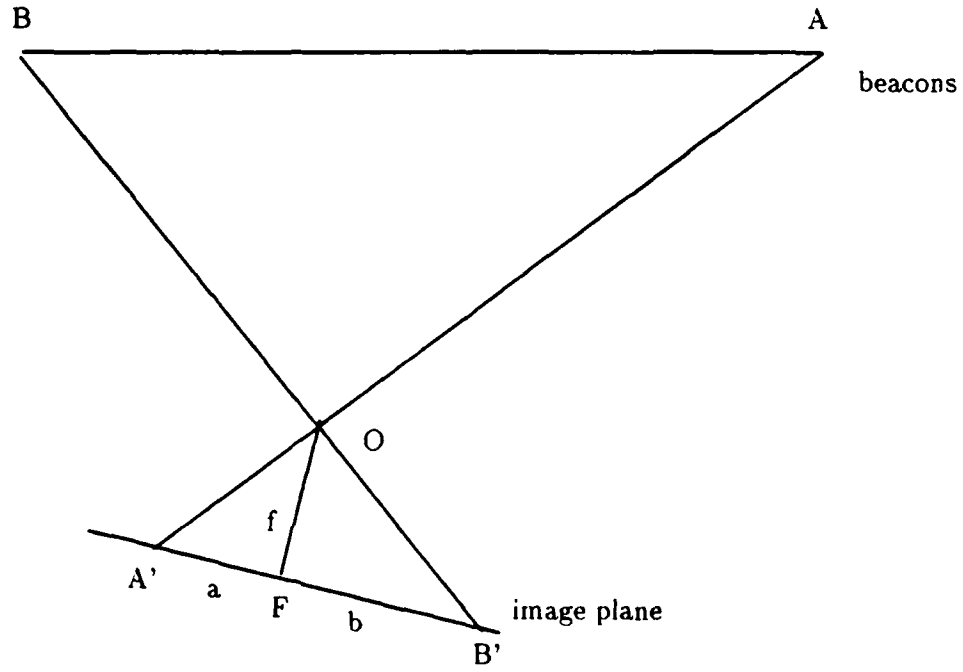



Figure A.1: System Error analysis

and $(x_b, y_b, f)$. Then,

$$\theta = \angle A'OB' = \tan^{-1}\frac{\sqrt{x_a^2+y_a^2}}{f} + \tan^{-1}\frac{\sqrt{x_b^2+y_b^2}}{f}$$

So, the error in the measurement of $\theta$ is:

$$\begin{aligned}\varepsilon_\theta &= \left(\frac{\partial\theta}{\partial x_a}+\frac{\partial\theta}{\partial y_a}+\frac{\partial\theta}{\partial x_b}+\frac{\partial\theta}{\partial y_b}\right)\varepsilon_e + \frac{\partial\theta}{\partial f}\varepsilon_f \\ &= \left(\frac{(x_a+y_a)f}{|Oa|^2\sqrt{x_a^2+y_a^2}}+\frac{(x_b+y_b)f}{|Ob|^2\sqrt{x_b^2+y_b^2}}\right)\varepsilon_e + \left(\frac{\sqrt{x_a^2+y_a^2}}{|Oa|^2}+\frac{\sqrt{x_b^2+y_b^2}}{|Ob|^2}\right)\varepsilon_f\end{aligned}$$

Where $\varepsilon_e = D/2r$ is the absolute error bound on the measurement error of the photodiode.

The error due to the uncertainty of beacon placement is derived as follows: The coordinates of beacons $A$ and $B$ are $(x_A, y_A, z_A)$ and $(x_B, y_B, z_B)$, respectively.

$$\theta = \cos^{-1}\frac{\overrightarrow{OA}\cdot\overrightarrow{OB}}{|OA|\,|OB|} = \frac{x_Ax_B+y_Ay_B+z_Az_B}{\sqrt{x_A^2+y_A^2+z_A^2}\sqrt{x_B^2+y_B^2+z_B^2}}$$

So, the error in the measurement of $\theta$ is:

$$\varepsilon_\theta = \left(\frac{\partial\theta}{\partial x_A}+\frac{\partial\theta}{\partial y_A}+\frac{\partial\theta}{\partial z_A}+\frac{\partial\theta}{\partial x_B}+\frac{\partial\theta}{\partial y_B}+\frac{\partial\theta}{\partial z_B}\right)\varepsilon_e$$

Each term in the above equations takes a similar form. For example,

$$\frac{\partial\theta}{\partial x_A} = \frac{(y_A(x_By_A-x_Ay_B)+z_A(x_Bz_A-x_Az_B)}{|OA|^2\sqrt{(x_Ay_B-x_By_A)^2+(y_Az_B-y_Bz_A)^2+(z_Ax_B-z_Bx_A)^2}}$$

The above formulas, although more complex than those presented in section 3.3, tell us the same results. That is, to minimize errors, large base line separations and long focal lengths are necessary.

# Appendix B

# Implementation of Church's Algorithm

```
/*************************************************************************/
/*                                                                       */
/* church.c: church's method.                                            */
/*                                                                       */
/*************************************************************************/
#include <stdio.h>
#include <math.h>
#include "defs.h"
#indlude "gvars.h"

/*
** variables for church's method
*/
#define MAXTRY 10            /* maximum number of iterations allowed    */
#define EPSILON 1.0E-10      /* iteration stopping criterion            */

static Point origin;         /* camera origin position                  */
double k[MAXVIEW+2];         /* face angles in camera coordinate system */
double L[MAXVIEW],
       M[MAXVIEW],
       N[MAXVIEW];           /* directional cosine in world coordinate  */
double l[MAXVIEW],
       m[MAXVIEW],
       n[MAXVIEW];           /* directional cosine in camera coordiante */


/*************************************************************************/
/*                                                                       */
/* Church's method. It first called calculate_k to compute the face      */
/* angles in camera coordinate system. Then compute the adjustment       */
/* equations to get an adjustment amount. The adjustment amount is       */
/* added to the current camera origin and iterated again. The method     */
/* stops when the adjustment amount is smaller than EPSILON, or number   */
/* of iterations exceeds MAXTRY                                          */
```

```
/*                                                                      */
/************************************************************************/

church()
{
double D[MAXVIEW];              /* lengths from camera origin to beacons */
double K[MAXVIEW];              /* face angles in world coordinate       */
double I[MAXVIEW],
       J[MAXVIEW];              /* auxiliary variables                   */
double A[3MAXVIEW],             /* coefficient matrix of adjustment      */
       dk[MAXVIEW+2];           /* adjustment amount                     */
int count = 0;                  /* number of iterations                  */
int i;


   /*
   ** face angles in camera cocoordinate system, calculate only once
   */
   calculate_k(image);

   /*
   ** calculate directional cosines in world coordinate system
   */
   do {
      for (i = 0; i < cview; i++) {
         D[i] = sqrt(sqr(led[pled[i]].x - origin.x) +
                sqr(led[pled[i]].y - origin.y) +
                sqr(led[pled[i]].z - origin.z));
         D[i] = 1 / D[i];
         L[i] = (led[pled[i]].x - origin.x) * D[i];
         M[i] = (led[pled[i]].y - origin.y) * D[i];
         N[i] = (led[pled[i]].z - origin.z) * D[i];
      }

      /*
      ** calculate face angles in world coordinate system
      */
      for (i = 0; i < cview; i++)
         K[i] = L[i]*L[(i+1)%cview] + M[i]*M[(i+1)%cview] +
                N[i]*N[(i+1)%cview];

      /*
      ** calculate auxiliary variables
      */
      for (i = 0; i < cview; i++) {
         I[i] = k[i] * D[i] - D[(i+1)%cview];
         J[i] = k[i] * D[(i+1)%cview] - D[i];
      }

      /*
      ** adjustment equations, calculate adjustment amount
      */
      for (i = 0; i < cview; i++) {
         A[i*3] = L[i]*I[i] + L[(i+1)%cview]*J[i];
```

```
            A[i*3+1] = M[i]*I[i] + M[(i+1)%cview]*J[i];
            A[i*3+2] = N[i]*I[i] + N[(i+1)%cview]*J[i];
            dk[i] = k[i] - K[i];
         }

         /*
         ** check stop condition
         */
         error = innerproduct(3, dk, dk);
         if (error > EPSILON) {
            inverse(3, A, A);
            multiplication(3, 3, 1, A, dk ,dk);

            origin.x += dk[0];
            origin.y += dk[1];
            origin.z += dk[2];

            count++;
         }
      } while (count < MAXTRY && error > EPSILON);


      /*
      ** print a message if it fails to converge in 10 runs. otherwise
      ** calculate the transformation matrix based on the current position
      */
      if (count >= MAXTRY && error > EPSILON)
         fprintf(stderr, "doesn't converge after %d round, error: %lf\n",
          MAXTRY, fabs(dk[0])+fabs(dk[1])+fabs(dk[2]));
      else
         calculate_matrix(origin);
   }


/*****************************************************************************/
/*                                                                           */
/* Calculate the directional cosines of the camera origin and the light      */
/* source images on the photodiode surface in camera coordinate system.      */
/* Face angles are the inner product of the directional cosines              */
/*                                                                           */
/*****************************************************************************/

calculate_k(image)
Point *image;
{
double length;
int i;

   /*
   ** directional cosine
   */
   for (i = 0; i < cview; i ++) {
      length = sqrt(sqr(image[i].x)+sqr(image[i].y)+sqr(image[i].z));
      length = 1 / length;
```

```
      l[i] = image[i].x * length;
      m[i] = image[i].y * length;
      n[i] = image[i].z * length;
   }

   /*
   ** face angle in camera coordinate
   */
   for (i = 0; i < cview; i++)
      k[i] = l[i]*l[(i+1)%cview] + m[i]*m[(i+1)%cview]
             + n[i]*n[(i+1)%cview];
}


/*****************************************************************************/
/*                                                                           */
/* Calculate the current transformation matrix based on the head             */
/* position from Church's method                                             */
/*                                                                           */
/*****************************************************************************/

calculate_matrix(origin)
Point origin;
{

   /*
   ** calculate the 3x3 rotational matrix
   */
   copyvector(3, &X[0][0], l);
   copyvector(3, &X[0][3], m);
   copyvector(3, &X[1][2], n);
   transpose(3, 3, X, X);
   inverse(3, X, X);

   multiplication(3, 3, 1, X, L, L);
   multiplication(3, 3, 1, X, M, M);
   multiplication(3, 3, 1, X, N, N);
   identity(4, X);
   copyvector(3, X[0], L);
   copyvector(3, X[1], M);
   copyvector(3, X[2], N);
   transpose(4, 4, X, X);

   /*
   ** copy in the translational vector
   */
   copyvector(3, X[3], &origin);

   /*
   ** use for debugging and testing
   */
   if (debug) {
      fprintf(stdout, "correct transformation matrix: \n");
      printmat(4, 4, ctow);
```

```
        fprintf(stdout, "calculated transformation matrix: \n");
        printmat(4, 4, X);
    }
}
```

```
/*************************************************************************/
/*                                                                       */
/* defs.h: common definitions used in all subroutines                    */
/*                                                                       */
/*************************************************************************/

#define ERROR -1
#define TRUE   1
#define FALSE  0
#define XYI    0                          /* polar coordinate system */
#define POLAR  1

#define radian(x) (double)(M_PI/180.0*x)
#define degree(x) (double)(180.0/M_PI*x)
#define sqr(x)((x)*(x))

typedef struct {                  /* homogeneous coordinate system    */
   double x, y, z;
   double d;
} Point;

typedef struct {                  /* two dimensional image coordinate */
   double x, y;
} Point2d;

typedef struct {                  /* camera view direction            */
  double rx, ry, rz;
  double tx, ty, tz;
} View;

typedef struct {                  /* a complex number presentation    */
  double r, i;
  int f;
} Complex;
```

```
/*************************************************************************/
/*                                                                       */
/* gvars.h: global variables                                             */
/*                                                                       */
/*************************************************************************/

#define CHURCH 0
#define DIRECT 1
#define NEW    2

#define CLOSE           0
#define OPEN            1
#define ENTER_HEADER    2
#define ENTER_DATA      3
#define UPDATE_JOY      4


#define MAXCAMERA 2             /* camera number (outsidein)        */

/*
** variables common to both tracking methods
*/
int insideout;                  /* insideout or ousidein tracking   */
int method;                     /* which method used to get solution */
double fl;                      /* local length of the camera       */
double efl;                     /* local length of the camera       */
double diode;                   /* photocell size                   */
int resolution;                 /* photocell resolution             */
double rmx, rmy, rmz;           /* room size                        */
View camera[MAXCAMERA],         /* camera position                  */
     head;                      /* head position (outsidein)        */
double spacingx, spacingy,
       spacingz;                /* led spacing                      */
double errx, erry, errz;        /* lead position error              */
double vrerr, vterr;            /* error in position each camera    */
int jitter;                     /* jittering                        */
int filter;                     /* filter jitter motion             */
int k1k2uv;                     /* which value set in new method    */
int debug,                      /* report church's error and the    */
                                /* calculated and correct matrix    */
    perror, qerror, verror,     /* position and quantization error  */
    select,                     /* select led, seldom used now      */
    mylog,                      /* log file                         */
    ledchange,                  /* report led change                */
    smooth,                     /* calculate led position           */
    setorigin,                  /* set head position                */
    global,                     /* report when move too fast        */
    converge,                   /* report iteration result          */
    nonconverge,                /* report when non converge         */
    display,                    /* final matrix got inverse (pxpl4) */
    time,                       /* timing                           */
    selfcal,                    /* camera self calibration          */
    deform,                     /* set when 1 view with 2 leds      */
    joystick;                   /* really enter display mode        */
```

```
/*
** variables used in church's method and Ganapathy's method
*/
#define MAXLED 1200
#define MAXVIEW 4

double ctow[4][4],              /* camera to world transformation     */
       wtoc[4][4];              /* world to camera transformation     */
double ctov[MAXVIEW][4][4],     /* transformation from camera to view */
       vtoc[MAXVIEW][4][4];     /* transformation from view to camera */
View   view[MAXVIEW];           /* keep view error                    */
double ectov[MAXVIEW][4][4],    /* transformation from camera to view */
       evtoc[MAXVIEW][4][4];    /* transformation from view to camera */
double X[4][4];                 /* calculated transformation matrix   */

int nled;                       /* number of led                      */
int nview,                      /* number of view                     */
    cview;                      /* view index.                        */
int diodex, diodey, diodez;     /* diode number                       */
Point led[MAXLED];              /* led array                          */
Point eled[MAXLED];             /* true led array, with eror          */
int pled[MAXVIEW];              /* the three led the camera use       */
Point image[MAXVIEW];           /* iamge points on image plane        */

FILE *fopen();
FILE *tty;

double innerproduct();
double determinant();
double vectorlength();
double pdist();
```