# Vision-based Tracking with Dynamic Structured Light for Video See-through Augmented Reality

by

**Mark Alan Livingston**

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

1998

Approved by:

Henry Fuchs, Advisor

Gregory Welch, Reader

David Eberly, Reader

James Coggins, Reader

**MARK ALAN LIVINGSTON. Vision-based Tracking with Dynamic Structured Light for Video See-through Augmented Reality.**
**(Under the direction of Henry Fuchs.)**

## ABSTRACT

Tracking has proven a difficult problem to solve accurately without limiting the user or the application. Vision-based systems have shown promise, but are limited by occlusion of the landmarks. We introduce a new approach to vision-based tracking using structured light to generate landmarks. The novel aspect of this approach is the system need not know the 3D locations of landmarks. This implies that motion within the field of view of the camera does not disturb tracking as long as landmarks are reflected off any surface into the camera.

This dissertation specifies an algorithm which tracks a camera using structured light. A simulator demonstrates excellent performance on user motion data from an application currently limited by inaccurate tracking. Further analysis reveals directions for implementation of the system, theoretical limitations, and potential extensions to the algorithm.

The term *augmented reality* (AR) has been given to applications that merge computer graphics with images of the user's surroundings. AR could give a doctor "X-ray vision" with which to examine the patient before or during surgery. At this point in time, AR systems have not been used in place of the traditional methods of performing medical or other tasks.

One important problem that limits acceptance of AR systems is lack of precise *registration*—alignment—between real and synthetic objects. There are many components of an AR system that contribute to registration. One of the most important is the tracking system. The tracking data must be accurate, so that the real and synthetic objects are aligned properly.

Our work in augmented reality focuses on medical applications. These require precise alignment of medical imagery with the physician's view of the patient. Although many technologies have been applied, including mechanical, magnetic, optical, et al, we have yet to find a system sufficiently accurate and robust to provide correct and reliable registration.

We believe the system specified here contributes to tracking in AR applications in two key ways: it takes advantage of equipment already used for AR, and it has the potential to provide sufficient registration for demanding AR applications without imposing the limitations of current vision-based tracking systems.

# Acknowledgments

I thank my advisor, Prof. Henry Fuchs; my readers, Prof. Greg Welch and Prof. David Eberly; and the other members of my committee, Prof. Gary Bishop and Prof. James Coggins. Much of this document is written in the first-person plural, reflecting the fact that many of the ideas presented here (as noted in Section 1.3.1) are in fact ideas from the committee members. I also thank Prof. Mike Schlessinger of the University of North Carolina Department of Mathematics, who helped me understand projective geometry and its application to this work.

Andrew Ade and Kathy Tesh were very helpful in making sure that I was able to meet with Prof. Fuchs and that administrative tasks were handled. David Harrison and Todd Gaul provided significant technical assistance with video equipment and production.

I thank Prof. David Breen of the California Institute of Technology and Dr. Gil Ettinger of the Surgical Planning Laboratory at Brigham and Women's Hospital for allowing me to use pictures of their work in augmented reality.

The work I have done in medical applications of augmented reality has been supported under grants from the Defense Advanced Research Projects Agency (ISTO DABT 63–93–C–0048, Henry Fuchs and Frederick P. Brooks, Principal Investigators) and the National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization (ASC–8920219, Henry Fuchs, Principal Investigator, UNC site). My dissertation research was also supported in part by a Link Fellowship from the Link Foundation of the Institute for Simulation and Training and by the Naval Command, Control, and Ocean Surveillance Center (N66001–97–1–8919, Henry Fuchs and Nick England, Principal Investigators).

I wish to thank the members of the augmented reality research group for their comments and insights on my work and in the field: Andrei State, Mary Whitton, Michael Bajura, Gentaro Hirota, Bill Garrett, Paul Rademacher, Ramesh Raskar, Jessica Crawford, Jeremy Ackerman, Michael Rosenthal, and Michael North. I also wish to thank several other current and former students, faculty, and staff who have been good friends and colleagues during my graduate studies: Ron Azuma, Rich Holloway, Leonard McMillan, Steve Goddard, Jacob Furst, Stefan Gottschalk, Hans Weber, Matt Cutts, Weihei Chen, Bill Mark, Kevin Arthur, Carl Mueller,

Jon Cohen, Dave Luebke, David Ellsworth, Mark Mine, Marc Olano, Andrew Thall, Nick England, Vern Chi, Kurtis Keller, Herman Towles, Turner Whitted, Brent Seales, Mike Brown, Arthur Gregory, Terry Yoo, and many others who took the time to listen to me talk about my research and gave helpful comments and suggestions.

Last, but certainly not least, I thank my parents, Peter and Bette Livingston.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **1D** | one-dimensional |
| **2D** | two-dimensional |
| **3D** | three-dimensional |
| **AR** | augmented reality |
| **ARG** | angular rate gyroscope |
| **avg** | average |
| **CCD** | charged-coupled device |
| **CRT** | cathode-ray tube |
| **CT** | computed tomography |
| **DMD** | digital micromirror device |
| **DOF** | degrees of freedom |
| **EKF** | extended Kalman filter |
| **HMD** | head-mounted display |
| **Hz** | Hertz |
| **KF** | Kalman filter |
| **LCD** | liquid crystal display |
| **LED** | light-emitting diode |
| **LEPD** | lateral effect photo-diode |
| **m** | meters |
| **mm** | millimeters |
| **mr** | milliradians |
| **MRI** | magnetic resonance imaging |
| **OST** | optical see-through |
| **QCPD** | quad-cell photo-diode |
| **rad** | radians |

| | |
|---|---|
| **RMS** | root-mean-square |
| **SCAAT** | single constraint at a time |
| **s** | seconds |
| **UNC** | the University of North Carolina |
| **VE** | virtual environment |
| **VR** | virtual reality |
| **VST** | video see-through |

# Chapter 1

# Introduction

> In this chapter, I will introduce virtual and augmented reality, including the components of these systems. The component of particular interest is the tracking subsystem; the tracking problem will be discussed in further detail. Finally, I will discuss the contribution of this project to the problem of tracking.

The term *virtual reality* (VR) has been used in the popular press to describe systems that completely replace the user's view with computer graphics, a view of a synthetic world. Such systems allow building designers to visualize the "completed" building and potential problems in the design before construction. A VR system might allow scientists to view environments as large as galaxies or as small as atoms as if they were human-sized. The primary benefit in these cases would be to interact in more natural ways with objects that, due to their size, are usually beyond our reach.

Three key hardware components comprise such systems: a display subsystem through which the user looks into the synthetic world, an image generation subsystem that paints the proper image onto this display, and a tracking subsystem that determines the user's viewpoint and view direction in order to paint the correct picture. The goals are to present a new world that is "real" to the user's senses, and to give the user a sense of presence in that environment. "Real" in this context implies in that the environment exhibits a consistent and believable appearance and behavior.

A variant on this idea is not to completely replace the user's view of the surrounding environment, but rather to add to it. Thus the user's sense of "reality" is "augmented" with synthetic imagery. Since the user sees real objects in the environment, the sense of immersion changes from presence in a different environment to added awareness or insight to the user's actual surroundings. However, the merging of synthetic objects with real imagery requires a consistent, believable depiction of the appearance and behavior of the synthetic objects.

Synthetic objects must behave as their real counterparts would behave. They must be situated in–and stay in–the proper place as the user moves through the environment. In other words, they must be properly aligned with the real objects. They must disappear when another object (real or synthetic) obstructs the user's view. These are two of the most difficult tasks in merging real and synthetic imagery. For most implementations of this type of system, achieving alignment reduces to acquiring accurate tracking data for the user's viewpoint and properly depicting occlusion between real and synthetic objects reduces to knowing the relative distance of the real and synthetic objects from the user.

Achieving alignment and depicting occlusion have been difficult problems to solve. Most proposed solutions have either limited success or a limited domain of application. The major motivation behind this dissertation was to create a system that could achieve alignment without restricting the user in the ways that previous approaches had. We did this in a framework that provides a good basis from which to solve the problem of correctly depicting occlusion relationships as well.

To introduce the current work to the reader, I will discuss, in the remainder of this chapter, the history of these merged environments and the types of tasks for which researchers have built systems. We've already noted above one of the most difficult problems in these systems—achieving alignment–and the need for accurate tracking of the user's viewpoint; thus the discussion moves to tracking subsystems. To close this chapter, I will propose an alternate strategy to current tracking subsystems.

## 1.1 Augmented Reality

The term *augmented reality* (AR) has been given to applications that merge computer graphics with images of the real world [Caudell92]. We have noted the ability of a VR system to present a different environment to the user. AR, on the other hand, has the ability to give the user further information about the surrounding environment. For example, AR could give a doctor "X-ray vision" with which to examine the patient before or during surgery. AR could also guide a manufacturing worker through a process with a series of visual demonstrations of the next step in the manufacturing process. Both of these applications have been demonstrated with AR [Bajura92, Caudell92].

Figure 1.1 gives an example of a simple environment which has been augmented with virtual objects; in this image, the chairs and the lamp are synthetic. Note that the lamp appears to be resting on the table and the chairs appear to be resting on the floor. This is an example of the type of alignment such systems seek to depict. Also note that the lamp occludes the portion of the table that is behind it and that the table partially occludes both of the chairs. Figure 1.2 gives another example in which a synthetic teapot is overlaid on a desktop scene of cube shapes. Note the reflection of the surrounding environment in the teapot, achieved by

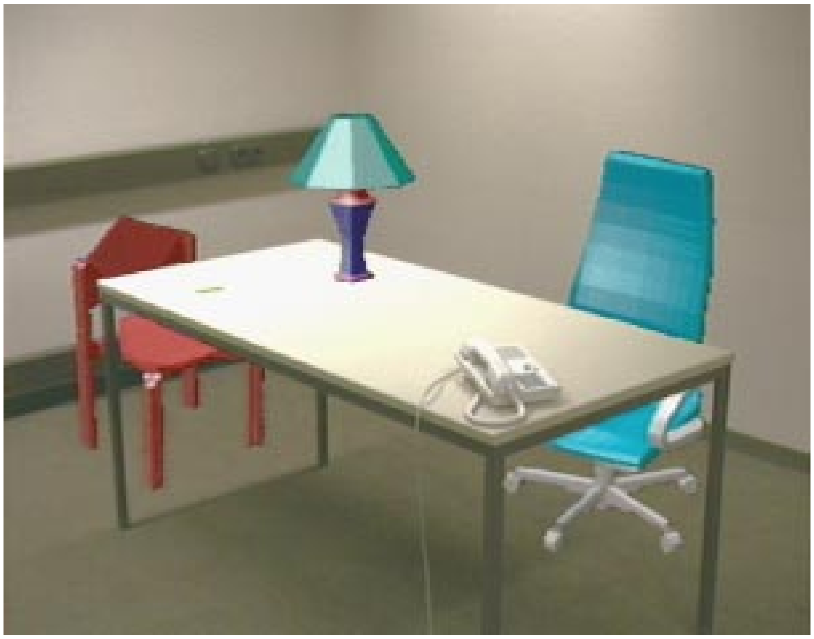


Figure 1.1: A simple AR environment. The walls, floor, table, and telephone are all real objects. The two chairs and the lamp on the table are all synthetic. Note how the lamp is aligned with the table top; it neither floats above the table nor sinks into the table. Also note that the synthetic lamp occludes part of the real table, while the real table occludes the two synthetic chairs. These two tasks are frequently difficult to perform in AR systems. Image ©1995 David Breen, Eric Rose, European Community Research Center.

placing a reflective metal sphere at that location on the table and capturing video of the reflection.

Thus far, AR systems have not been used in place of the traditional methods of performing these tasks. The lack of acceptance of AR systems is due to a series of technical challenges that have yet to be overcome. VR systems are more often accepted, and the components of an AR system are similar to those in a VR system. The difficulties that hinder current AR systems are primarily in two categories. The display device for AR is necessarily different than for VR, and the design of such devices is a formidable task. At UNC we have chosen to mount cameras near the user's eyes and display the images from those cameras to the user on an opaque display, a method known as a *video see-through* (VST) display. Display devices for AR will be discussed further in Section 2.1.1.1. The demands on the tracking subsystems are somewhat different for AR than for VR, and most tracking subsystems are not designed to meet these demands.

The basic notion of AR is to add sensory cues to the user's experience in the real world, so that the user perceives synthetic objects to be present in the real world. In theory, any of the senses could receive synthetic input, but the vast majority of work has been directed at the visual sense. The haptic (touch) and audio senses have also received some attention; however, I will discuss only the visual presentation of AR in specific terms.

The following discussion assumes that the reader is familiar with the basic technology of VR, also known

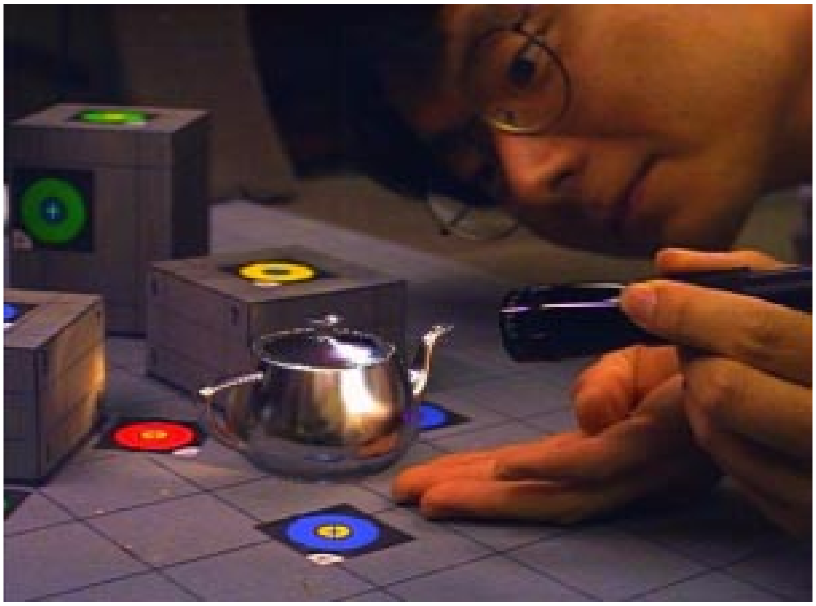


Figure 1.2: A tabletop AR environment in which a synthetic teapot reflects the real world that surrounds it. This effect is achieved by placing a reflective metal sphere at the appropriate location on the table, capturing the reflection map of that sphere by accurately knowing the projection of the sphere in the image, then rendering the reflection map onto the synthetic teapot. This image was created to demonstrate a vision-based tracking system developed at UNC [State96a] and described in Section 2.2.3.

as virtual environment (VE). Recent texts [Rheingold91, Sherman98] provide a complete discussion of VE. Further details about AR can be found in survey articles [Azuma97, Klinker97], but the primary sources are still the various papers describing specific implementations cited below.

### 1.1.1 Historical Context

The concept of a display system indistinguishable from reality was introduced by Ivan Sutherland in *The Ultimate Display* [Sutherland65]. A preliminary realization of this goal for the visual sense was described in *A Head-Mounted Three-Dimensional Display* [Sutherland68]. The system included not only a head-mounted display (HMD), but also an image generation subsystem and a tracking subsystem for the user's head and one of the user's hands. It is interesting to note that this first HMD was an AR display, not a completely immersive display suitable for VE. (Design of displays for AR is discussed in Section 2.1.1.)

The early work quickly revealed the technical hurdles that need to be cleared in order to realize useful applications of the technology. I motivate discussion of these problems with a brief description of a series of increasingly more demanding applications. These applications have in some cases been implemented; in others, they have merely been suggested or are similar to suggested applications. With this basic understanding of the goals and requirements of the AR system, I describe in Section 2.1 the hardware and software used to build AR systems. I list in Section 2.1.2 the types of applications that have been suggested.

### 1.1.2 A Sample Application Set

Let us consider a series of increasingly more challenging AR applications that will introduce the difficult problems in constructing an AR system. The challenge in these applications will come from the number and complexity of tasks that must be solved in order to realize a useful system. The tasks will thus be introduced here, although detailed discussion of them will wait until Section 2.1.1. I will use a medical theme for these examples, since my experience lies primarily with medical AR systems.

Suppose that a physician wants to monitor a patient's vital statistics while performing an examination. One easy solution is to place the monitors that display the statistics in the physician's field of view during the course of the examination. This approach, however, would require making assumptions about the position of the physician and of the patient during the examination, which would limit the applicable procedures. A more complex solution is to automatically have the monitors appear in view at all times. This would require having a display fixed on the physician's head, with sufficient resolution to enable the reading of text. Observe that although the information added into the physician's display is connected to the object (the patient) within the field of view, it is not important where in the field of view the added information appears, except that it not obstruct the physician's view of the patient. Also observe that there is no depth associated with the synthetic imagery—that is, there is no question whether the text should disappear behind the patient; it is always in front.

Now suppose that the physician is examining a pregnant patient. The physician uses an ultrasound scanner to provide images of the internal anatomy. In order to understand the relationship between the ultrasound data and the patient's exterior anatomy, the physician must mentally align the shape visible in the data with the external shape. It is not in general an easy task to establish these relationships mentally. However, the physician's display could simply overlay the medical image data on top of the view of the patient's exterior anatomy with the proper alignment. The accuracy required for this alignment would be enough to place the medical imagery near the proper location, but a high degree of precision would not be necessary simply to evaluate the development of the fetus. However, since there are now two sets of three-dimensional (3D) imagery (the real world view of the patient and the ultrasound data) in the visual input, the merged image should correctly represent the relative depth of the real and the synthetic imagery. The HMD needs sufficient quality to show medical imagery in enough detail. This depends on the physician and the task. Since the physician's goal is only to examine the patient, not perform an interventional procedure, the requirements of visual quality and accuracy of alignment and relative depth are moderate in degree, in the sense that they are within reach of current AR systems.

Now suppose that the physician needs to insert a needle into the womb in order to extract some fluid for evaluation, a procedure known as amniocentesis. The task of aligning the medical imagery and the exterior anatomy remains, but now the accuracy required has increased dramatically. The medical imagery must be aligned to within a few millimeters, if not closer, in order to avoid inserting the needle into the fetus. Similarly, the quality of the display must increase. The need to establish the depth relationship between the real and synthetic imagery also remains. Current tracking and display subsystems are unable to meet these increased requirements for AR systems.

From this discussion, it should be evident that the application greatly affects the requirements of the various pieces of the AR system. Section 2.1.1 will continue this discussion with a description of how AR systems and technology can be implemented. For now, let us turn our attention to the tracking subsystem, which we have identified as one on which AR places significant demands.

## 1.2 Tracking

The tracking problem can be stated as follows.

> Determine the position and orientation angles of a given object with respect to a reference pose in the environment.

The position and orientation together describe the six degrees of freedom (DOF) offset between two discrete 3D objects. This six DOF quantity is referred to as *pose* in the following discussion.

The tracking system in an AR or VE application determines the viewpoint and view direction for rendering the synthetic imagery. For monitor-based computer graphics applications, the tracking system is often a joystick or mouse controller, or perhaps a set of keys on the standard keyboard. In this case, the viewpoint is fixed until the user takes action. This action, however, is not an intuitive viewpoint control method.

Part of the novelty of AR and VE as a user interface is to provide a natural and intuitive method of interaction. The most intuitive method for controlling the viewpoint is for the user to physically move into the desired viewing location relative to the environment. In order to allow the user this natural interface, a more sophisticated tracking system must measure the user's pose in the environment in real-time. This enables the application to use a spatially immersive display, such as projection displays or HMDs.

I discuss tracking of the user's head only; however, many applications track other objects, such as the user's hands or objects that the user may want to move arbitrarily but are otherwise fixed. For example, in the ultrasound-guided needle biopsy application described in Section 2.1.2.4, the physician needs the ultrasound probe tracked during scanning, but might leave the probe positioned in a fixed pose while performing the biopsy. However, I concentrate on tracking the eyepoint since that is the topic of this dissertation.

Tracking for AR has proven to be a very difficult problem to solve. In a naïve implementation of AR, the synthetic imagery is rendered with the measured pose and simply overlaid on the real imagery with the assumption that the tracking system will provide enough accuracy in its measurement of the pose to accurately align real and synthetic imagery. However, alignment has rarely been achieved to sufficient accuracy in the demanding medical applications. This is partially due to the stringent requirements which various applications carry, and partially due to inadequate technologies that have been applied to the problem. Hardware and algorithms for tracking are discussed further in Chapter 2.

## 1.3 Thesis

### 1.3.1 History of Proposed System

The idea for this work comes from several concurrent projects at UNC. In my work as a research assistant, I helped implement the AR system for needle biopsy and laparoscopic visualization described in Section 2.1.2.4. As noted, these applications require very precise alignment of medical data with the physician's view of the patient. Without this alignment, the surgeon can not properly perceive the 3D structure of the medical data within the patient. We have not yet found an existing system that provides the required accuracy.

As part of this effort, we are developing a hybrid tracking system [State96a] consisting of a magnetic tracking system and tracking of a set of painted markers which are visible in an image of the environment acquired and displayed to the user. This system, described more fully in Section 2.2, achieves very good registration, but at the cost of limiting the surgeon's movements to maintain that precise registration. The limits are in the view position and direction, which must be such that the landmarks are in the view frustum of the physician, and in the placement of the physician's hands and surgical instruments, which must be such that the landmarks are not occluded from the surgeon's viewpoint. These restrictions defeat the system for practical use. We believe the answer to these problems is to have the landmarks not be physical objects, but projected patterns. This scheme introduces similar geometric constraints to those used in many algorithms in the computer vision community. In the computer vision literature, the technique of projected light patterns to acquire geometric information is known as structured light.

Prof. Henry Fuchs suggested to me the possibility that a limited version of structured light that consisted only of dots might be suitable to replace the physical dots we had been using for the vision-based portion of the hybrid tracking system. After some consideration of the constraints that such a system would place on the location (position and orientation) of the camera, I wrote a simple linear system that theoretically would solve the system to within a scale factor. At the same time, Prof. Gary Bishop was advising Leonard McMillan on his dissertation on image-based rendering. This work requires knowing how an image taken from one camera will look from another viewpoint. Prof. Bishop noticed the similarity of the known information in that work and in my proposed tracking algorithm. He suggested a single constraint at a time (SCAAT) [Welch96] approach to the system that forms the basis of the algorithm presented in Chapter 3. The SCAAT framework is behind the algorithm for the current version of the UNC optical ceiling tracker, which achieves very good performance.

Prof. Fuchs had been discussing the generation of structured light patterns for depth extraction with Prof. Bishop. They believed that the new digital micromirror device (DMD) [Hornbeck95] available from Texas Instruments would be suitable for projecting precise light patterns across a surface. Such patterns have long been used in computer vision to acquire scene structure. Their hope was that a real-time depth-extraction system could be developed that would allow the AR surgical system to know where the patient's skin was at all times. This in turn would allow proper depiction of occlusion with synthetic models of interior patient anatomy, exterior patient anatomy, and tracked medical instruments. Eventually, the physicians hands would be added to this computation, though tracking of deformable objects such as human hands would be a more difficult task.

Profs. Fuchs and Bishop had also noted that the structured light patterns when generated by fast-switching DMD could then be inverted and projected again, in order to keep the amount of light in the scene constant across the image captured by the cameras. Fast switching would make the patterns of light imperceptible to the human eye. This is because the human visual system integrates the amount of light seen over time in order to acquire an image. Many of us experience this effect every day in our offices. A fluorescent light flickers on and off, even though that flicker is not noticeable to observers in the room. On the other hand, strobe lights such as are popular in dance halls flash at much slower speeds and can have a dizzying effect on people. For the proposed tracking system, imperceptible structured light would imply that the landmarks would not interfere physically with the user, and not interfere perceptually with the user or others in the environment.

In initial experiments, we have demonstrated that imperceptible patterns consisting of a single point, of a single line, or of a set of text are barely noticeable to human users [Raskar98]. More complex patterns are somewhat noticeable, although we believe we can reduce this by constructing a direct interface to the DMD chip, rather than communicating via the analog video interface we currently use. The crucial component in determining whether a pattern is imperceptible is the amount of change in the image on the viewer's retina. Any change that occurs completely within the interval over which the eye is integrating light will not be perceived. Thus a high-speed flicker will be unnoticed. We have noticed that saccadic movement of the eye or rapid occlusion-unocclusion events (such as rapidly waving your hand in front of your eyes, then out of the field of view) will create enough change in the retinal image to enable a human to perceive a pattern that was not otherwise perceptible.

### 1.3.2 Thesis and Contribution

The motivation for this work was to design a tracking system suitable for VST AR systems that would overcome the key limitations of previous vision-based tracking systems. Specifically, we wanted a system that

- reduced restrictions introduced by physical landmarks.
- reduced problems resulting from landmark occlusion.

As discussed in the previous section, we postulated at the beginning of this research that we could overcome these two limitations by using projected light patterns as landmarks instead of brightly painted markers. Thus the landmarks would no longer be physical entities, and we neither need nor want to know their 3D locations. Instead, the landmarks are identifiable points in projected patterns of light. With the projectors at known locations, the camera detects those landmarks, allowing the algorithm to constrain its estimate of the pose of the camera. This observation and system design lead us to the following thesis statement.

> A real-time estimate of the camera pose can be computed by projecting and detecting light patterns onto an unknown, dynamically changing surface geometry. This real-time estimate is sufficiently accurate to enable precise registration in video see-through augmented reality applications in which the user performs a task demanding hand-eye coordination at arm's length.

The most significant aspect of the new algorithm is that, unlike all current optical tracking systems, knowledge of the 3D locations of the landmarks is not required for tracking. While some systems have demonstrated the ability to calibrate the locations of landmarks during tracking [Welch96, Neumann98], this system is fundamentally different than those approaches in that it avoids knowing or computing the location at all. This can be a significant advantage for vision-based tracking systems when applied to dynamic environments for augmented reality. In our work, we have had difficulty maintaining accurate registration in dynamic environments (i.e. when objects are moving within the user's field of view). Since our new algorithm does not require knowledge of the 3D locations of the landmarks, it does not restrict the static or dynamic geometry of the environment. Since our algorithm is vision-based, it does not require placing additional sensors on the user's head beyond the video cameras that would already be there to establish the VST view.

This type of constraint is not new; it is common in computer vision algorithms. However, in those algorithms the constraint is generated by tracking features in the environment in an image stream. This configuration is susceptible to the problem of landmark occlusion in the same way as the tracking of colored dots described above. Our configuration does not assume that the landmarks are fixed in the environment (or even are the same from frame-to-frame). Thus we trade temporal coherence of landmarks for the potential to detect landmarks more consistently in dynamic environments. We believe that this will overcome the limitations of our current vision-based tracking system listed above. This will, we believe, benefit AR systems in both accuracy and robustness. Performance tests in Chapter 4 support the claims of accuracy with a user study of motion paths acquired in our primary intended AR application.

One unforeseen, but important, advantage is the potential to add capabilities to the tracking system that are valuable to a VST AR system. Namely, it can track the intrinsic parameters of the cameras used for the VST view, and it can acquire surface information which can be used to help resolve occlusion. These are not implemented even in simulation; however, they demonstrate that the new algorithm is an elegant framework for solving other important problems in VST AR, occlusion and camera calibration. Further details about both of these features will appear in Section 5. Another advantage is that the proposed system eliminates one difficult calibration step required in previous AR systems by estimating the camera pose through the camera that acquires the AR view.

This algorithm derives from both current vision-based tracking systems applied to AR and research in computer vision. Compared to previous vision-based tracking systems in AR, our algorithm places few restrictions on the user and the environment and is more robust to landmark occlusion. Compared to previous work in computer vision, our algorithm identifies corresponding points more directly via the use of dynamic structured light, provides a simple recursive algorithm for estimating the camera pose, and demonstrates a real-time system capable of six DOF measurements.

# Chapter 2

# Related Work

This chapter discusses the issues involved in implementing augmented reality: display devices, registration, occlusion, and latency. It surveys the range of applications for which AR has been suggested or demonstrated. The discussion then moves into a detailed examination of tracking devices. We look at the requirements of tracking subsystems, the hardware that has been used to create such devices, and the tracking algorithms relevant to the algorithm proposed in Chapter 3. Also, this dissertation uses concepts that are well-known in the field of computer vision and summarized here.

## 2.1 Augmented Reality

### 2.1.1 Implementation of Augmented Reality

With the example applications of Section 1.1.2 in mind, let us now look in greater detail at the requirements for some of the hardware and software components of an AR system. I concentrate on four requirements which are different for AR systems as compared to VE systems: the display device, registration, occlusion, and latency management.

#### 2.1.1.1 Providing a See-through Display Device

While many hardware requirements are shared with VE systems, AR presents unique challenges for the device that displays merged real and synthetic imagery. I discuss AR displays only; complete discussions of displays for VE are available [Melzer97]. Designing a display device for augmented reality is not a simple task. The issues range from technological to perceptual in nature, and have long been a subject of study in our lab [Rolland94]. This subsection briefly discusses some of these issues.

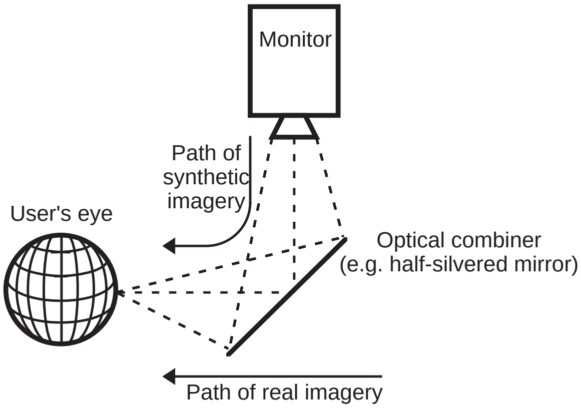

Monitor
Path of synthetic imagery
User's eye
Optical combiner (e.g. half-silvered mirror)
Path of real imagery

Figure 2.1: Conceptual design for an OST HMD. The merging of real and synthetic imagery is done by an optical combiner such as a half-silvered mirror. This means that the real world is dimmed when viewed through the combiner, regardless of whether any synthetic imagery is presented to the user. The synthetic imagery is displayed by a monitor which is aimed at the combiner. The partially reflective coating of the combiner reflects the synthetic imagery to the user's eye. Like the real imagery, less intensity than the source intensity of the synthetic imagery reaches the user's eye. This design also does not permit full occlusion of the real imagery by synthetic imagery, since the combiner is partially transmissive everywhere.

**Optical See-Through** Augmented views of the world have traditionally been generated by one of two technologies. The first method is known as *optical see-through* (OST) . Here the merging is performed by having the user see through (in a literal sense) half-silvered mirrors, as described for Sutherland's original implementation [Sutherland68]. The synthetic imagery is projected onto the mirrors' surfaces and reflected into the user's eyes. An example design is shown in Figure 2.1. Another implementation of an OST HMD is the Augeye display [Manhart93], which uses a slightly different arrangement of lenses to direct images into the eye.

OST blends real and synthetic imagery without reducing the user's visual acuity of the real world. This comes, however, at the cost of not being able to fully occlude the real world with the synthetic imagery, and at the cost of reduced brightness of the portion of the real world that is seen through the mirrors. The lack of complete occlusion can severely hamper the user's ability to believe in the merged environment. This might not be a problem in an application that merely provides a textual overlay of information, but would be a big problem in establishing depth relationships between, for example, medical image data and the patient surface.

Another, more recently proposed approach to OST is to scan images directly on the user's retina using a low-power laser. This device, known as the Virtual Retinal Display [Viirre98], was developed primarily to overcome macular degeneration, keratoconus, and other common vision problems. However, the authors note its potential to be used as a display for AR, since it can produce bright images even in the presence of bright ambient light. It still lacks the ability to completely occlude a bright portion of the real view with a dark

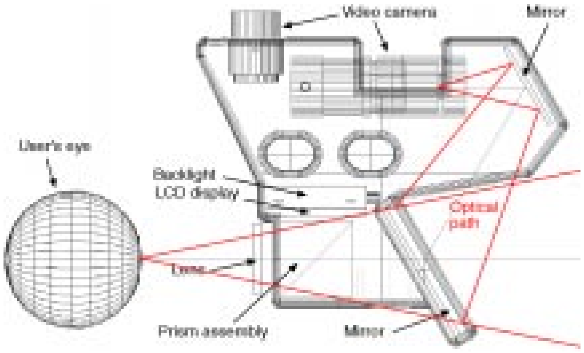


Figure 2.2: Design schematic for one eye pod of a VST HMD designed by our research group at UNC in collaboration with the University of Utah. The camera on the top (in two pieces) acquires a view of the real world by looking down an optical path that is twice folded by mirrors. This folding allows the camera to acquire the same image that the eye would acquire if the display were not occluding it. The display screen is nested inside a small cube just in front of the user's eye. With custom optics, this display appears further away than it really is, matching the distance of the optical path of the camera to the real world. The display is opaque, but by loading the image from the camera into the display, the user sees the surrounding environment.

synthetic image. The authors point out that the power levels are well within the American National Standard for safety, but the long-term effects of such displays merit further study.

**Video See-Through** The second method for generating an AR view is known as VST, in which the merging is performed by having the user "see through" (in a metaphorical sense) video cameras that acquire an image of the real world [Bajura92]. The display itself is a standard opaque HMD, but by positioning the video cameras at or near the eyes and placing the images they acquire in the display, the VST AR system allows the user to view the surrounding environment (Figure 2.2). This VST view comes at a cost of reducing the user's visual acuity of the real world to that allowed by the resolution of the cameras and the resolution of the display screen in the HMD. Currently the highest resolution available in a commercially available HMD is $1280 \times 1024$ pixels with full color [Kaiser98], and this quality comes at a significant cost. Less expensive displays have a resolution of up to $640 \times 480$ pixels with full color in recently available models.

VST is currently implemented with raster-scan cameras and displays. This implementation introduces latency between user motion and perception of that motion in the display of the real world. Latency is currently a significant problem in AR, and is discussed in Section 2.1.1.4. Other technological problems in VST have been partially solved or seem with reach. For example, stereo VST assumes the availability of video acquisition equipment for two real-time video streams, which requires greater bandwidth than exists in many current architectures. However, multiplexing of video signals can make efficient use of the available bandwidth. The VST design assumes that the center of projection of the camera is precisely aligned with the user's eye. Any offset in this alignment leads to an offset between visual cues and other sensory cues and affect depth perception. The design in Figure 2.2 overcomes this offset with an arrangement of mirrors.

**Field of View** Another major issue is the field of view. For a VST AR HMD, the field of view acquired from the cameras defines the field of view used for rendering the synthetic imagery. There are currently no designs capable of acquiring a wide field of view for VST. This is a fundamental problem for VST systems.

Most HMDs achieve a field of view that covers only a fraction of the normal human field of view. Typical values for a human's total field of view are [Barfield95] $180^\circ$ to $200^\circ$ horizontally ($150^\circ$ in each eye with around $120^\circ$ overlap) and $125^\circ$ vertically. HMDs for VE systems have fields of view ranging from $20^\circ$ to $140^\circ$ horizontally and $20^\circ$ to $40^\circ$ vertically. This shrinking of the field of view makes the user feel as if he is wearing blinders. This can be very disturbing if the user is to move in an AR environment.

**Display Screen** There are other issues related to HMDs that affect AR and VE systems and prevent AR from reaching end users. The most important of these is the display screen itself. Most HMDs use liquid crystal display (LCD) for the display. Until very recently, these devices could not be made with sufficiently high resolution and sufficiently small physical size to provide a suitable HMD. Other HMDs use cathode-ray tube (CRT) for the display device, which amounts to a standard television screen. While such devices can have very high resolution in a small unit, they tend to be too heavy to hang on a user's head. Such systems have been constructed, however, and put to limited use.

The standard design of HMDs for both VE and AR has become to place optical elements between the display screens and the user's eyes in order to make the display appear as if it occupies a larger field of view. This has the side effect of reducing the apparent resolution of the display by making the individual pixels appear larger.

Clearly, the resolution required varies from application to application. If the physician in our earlier example were merely reading text, then the resolution may affect the **size** of the synthetic imagery, but would not severely limit the ability to read text of an appropriate size **in pixels**. This is because there is no depth associated with such a two-dimensional (2D) overlay. Thus the number of pixels we devote to the text does not change with the user's viewpoint, and we can choose to use a sufficient number of pixels. This may come at a cost of occluding an important portion of the real world, depending on the user's viewpoint. If, in another application, the physician were trying to view ultrasound data that is aligned to the patient in depth, then the resolution would affect the acuity with which the physician sees the data. Features in medical images can be quite small, thus already requiring a high resolution display.

**Peripheral Vision and Alignment to User's Head** Since the user is not immersed in the synthetic world, the HMD might want to allow the user to have peripheral vision around the display. If so, then the border between the peripheral vision and the augmentable vision should ideally be seamless, and objects should appear continuous through this border. Latency and poor calibration of the HMD both accentuate the boundary. Most OST systems have a seamless border, but it has been difficult to achieve this in a VST system.

Another issue for VST display design is the offset between the user's eyes and the cameras' centers of projection. This offset creates a seam in the real world objects, which can be disturbing. It also hinders depth perception, and conflicts with the user's measurement of depth achieved by touching objects. This is obviously not a problem in OST.

Once a VST system achieves alignment between the cameras and user's eyes, the alignment is presumed to remain correct. However, if the HMD shifts on the user's head, then a small error would be introduced in the transformation from the user's eye (or the camera) to the tracker sensor rigidly affixed to the HMD. Most AR systems track the user's head and infer the position of the user's eyes with respect to the displays. This approach can work, although it can be tedious to measure this offset, as discussed in Section 2.1.1.2.

**Discussion** Most commercially available HMDs are not ready off the shelf for AR. Those devices that are available are, to our knowledge, currently all OST devices. We believed that the occlusion resolution offered by VST was important enough to select it over OST for our application, after having first attempted OST [Bajura92]. Thus we were left to custom-design our own VST HMD for our research in medical AR systems for performing interventional medical tasks. (Our research is similar to the last application mentioned in Section 1.1.2 and is described in Section 2.1.2.4.) We used a $640 \times 480$ grayscale display, the highest resolution available at the time. Even this resolution was too low for much of the synthetic imagery, especially medical data displayed at arm's length. We also found that adjusting the device to each individual user in order to maintain good alignment to the user and a good display focus was a never-ending task.

All these reasons make AR systems difficult to design, build, and deliver to clients. Design issues and calibration issues [Edwards93, Janin93] for HMDs have both received attention in the literature.

#### 2.1.1.2 Registration

Another major technical problem that has limited acceptance of AR systems by clients is the lack of *registration*, or the alignment between real and synthetic objects on the image plane. There are many pieces of an AR system that contribute to registration of the final image. Those relating to HMDs were discussed in Section 2.1.1.1. One of the most important, however, is the tracking subsystem. The output of this subsystem is passed to the image generation subsystem in order to generate a view of the synthetic objects that matches the user's view of the real world. This data must be accurate, so that the real and synthetic objects are aligned (spatially registered), and this data must be timely (temporally registered) [Jacobs97], so that the synthetic objects do not appear to "swim" back and forth in relation to the real world objects.

Generally, registration in AR is accomplished by having a precise model of the virtual object, knowing the location of the virtual object(s) in the real world, and knowing the pose of the real viewpoint–be it the user's eyes in OST or the camera viewpoint in VST–precisely. The rendering engine can then simply rasterize the object at its location, using the real viewpoint, on top of the video background provided by the video camera. This requires a lengthy and accurate calibration procedure, which remains error-prone. Thus, recent work has attempted to apply computer vision techniques or projective geometry to the problem of aligning synthetic imagery with its real counterpart.

Calibration is also important for registration [Tuceryan95, Whitaker95]. This encompasses a variety of non-continuous measurements of system parameters, such offsets between coordinate systems or camera projection parameters in VST. These measurements are frequently tedious, computationally expensive, or inherently difficult to perform accurately. Here are some of the coordinate systems that may be included in the system parameters.

- the coordinate systems of the tracking subsystem reference and sensor(s)
- the coordinate system(s) in which the synthetic objects are defined
- the coordinate system of the eyes or cameras

Note that hidden in the last coordinate system is an offset between the user's two eyes in an OST system or between the two cameras in a VST system. Also, the parameters that control the projection of the world onto the user's retina (OST) or the camera image plane (VST) must be measured and used for generating the synthetic imagery. For VST, this measurement is a classic problem in computer vision, known as camera calibration [Faugeras93].

With the tedious nature by which most of these measurements are acquired, calibration is an area that requires further research. Vision-based tracking systems have been applied to automatic calibration because of the non-intrusive nature of their acquisition, but frequently non-intrusive implies tracking natural features, a difficult and computationally intensive task. The SCAAT algorithm [Welch96] discussed in Section 2.2.3.3 provides a good framework for moving system parameters into the set of parameters estimated in real-time rather than in an initial calibration phase.

Alternate image-based techniques for achieving proper registration don't explicitly compute the viewpoint and view direction with a tracking system. A principle of projective geometry states that for a set of four or more non-coplanar points in 3D, the projection of all of the points in the set can be computed as a linear combination of the projections of just four of the points. This implies that by simply tracking four fiducial points on an object in the video image, the entire object can be registered to the real image [Kutulakos96]. This avoids having to know the position and orientation of the object relative to some fixed coordinate basis, in which the camera is also tracked. The camera-to-object transformation is effectively computed directly, but not explicitly. It is only determined how it projects into the final (augmented) image.

It is possible to track a set of features on an object without explicitly establishing correspondence. This requires tracking a set of features on an object and minimizing an error function. Such a function might be based on the moments of the distance between the noisy measurements for the 2D image locations of the points ordered arbitrarily and the points ordered in some standard order [Iu96]. This presumes an accurate model for the object.

#### 2.1.1.3 Occlusion

If precise alignment is achieved, then (and only then), can proper *occlusion* relationships be established between real and synthetic objects. This is the second major technical problem that most AR systems must overcome. The problem can be stated as follows.

> Portions of real objects that are behind portions of synthetic objects in the merged world must be obscured by those synthetic objects in the final image. Synthetic objects that are behind real objects in the merged world must similarly be obscured by those real objects.

Not all AR applications require occlusion resolution to accomplish their goals. Annotation applications clearly want to block out the real imagery with the annotation, so that it is clearly visible and legible. Repair and design tasks may or may not need proper occlusion in order to give clear instructions. For example, an unoccluded view of an object that is to be extracted from behind other objects might help the user better visualize the path necessary to free it. Then while the object is being moved, correct occlusion would probably improve the user's understanding.

In order to correctly represent the occlusion relationships between real and synthetic objects, an AR application needs to consider the depth of the real imagery in the rendering of the synthetic imagery. Synthetic image generation for AR is the same as for VE [Foley90]. Of particular interest in this discussion is the depth buffer, or $z$-buffer. This is a standard way that occlusion relationships are established for objects in VE. The way that AR should differ is that the AR system needs to measure or infer depth for the real imagery and compare this data with the computed depth for the synthetic imagery. The AR system can then display exactly the synthetic imagery that is closer than the real imagery, and not paint over real imagery that is closer.

Performance of either version of this task requires that the system know the depth of the real objects from the camera. Effectively, the rendering engine must assign a value in the depth buffer to each pixel corresponding to the real objects in the scene. Note that no information is written in to the color buffer; the color from the see-through display (video or optical) appears in the final image. One option is to acquire the depth data before the system runs and assume that the scene is static. In some applications, this assumption is not valid, so the object's depth must be recomputed or reacquired in real time in order to maintain the illusion of a merged world.

Knowing the depth at the pixels containing synthetic imagery is not enough. Our visual system uses other cues to determine depth relationships. The context around those objects must be correct as well. For example, in the ultrasound application, the synthetic data consists of a set of small objects. If these objects are merely overlaid onto the real scene, they appear to float in front of patient's skin, since we are accustomed to seeing a human form as a continuous smooth surface. The proper depth relationship at the pixels of the synthetic imagery must be complemented with a surrounding synthetic object that gives 3D context to the depth relationship, such as a synthetic depiction of a hole [Bajura92].

The most common method of resolving occlusion in AR is to simply render models of real world objects into the depth buffer. These objects must either be static in the world or tracked in real time. Assuming the model is accurate, the rendering engine can rasterize depth and use these values in the $z$-buffer computations that are standard in computer graphics architectures. Several systems have used this approach [Whitaker95, State96b, Breen96].

If a real-time depth map from the camera's pose is available, then this depth data can be loaded directly into the $z$-buffer for its usual computations during rendering of the synthetic imagery. Acquisition of a real-time depth map is a classic computer vision problem which has proven to be very difficult to solve in an accurate and robust fashion, let alone in real time. Some systems have used this approach [Wloka95, Breen96].

Another approach to resolving occlusion is to compute a mask based on the occluding contours of the real objects [Berger97]. This requires computing the outline of the synthetic object, finding any contours in the real image within this outline, labeling the contour points as behind or in front of the virtual object, then building the mask from the contour points that are in front.

The new algorithm we propose for tracking also has exactly the information needed to compute the depth of the nearest real object at the pixels at which the light patterns are detected. Thus our approach has the ability to compute a partial depth map in real time. This idea is discussed further in Section 5.2.

#### 2.1.1.4 Latency

In any AR system, there are multiple input sources from the environment; at a minimum, there are the eyes or cameras acquiring a view of the real world and the tracking system acquiring the pose of the user's head (OST) or of the camera (VST). In order to properly align real and synthetic imagery, these systems must agree on not only a spatial reference so that registration can be achieved (Section 2.1.1.2), but also a temporal reference. Temporal displacement of input sources is called *latency*.

The image generation system must know where the user **will be** when the image is displayed. If it does not and cannot estimate it, then there will be latency between the real and synthetic imagery, and the illusion of a the synthetic objects existing in the real environment would be lost. For OST, the system must know where the user's eye will be when the synthetic imagery is displayed. For VST, the system must know where the camera was when it acquired its image in order to render the synthetic imagery from the same viewpoint. If the total time required to present a merged image is too long, then there will be delay introduced between the user's motion and the user's opportunity to see the effect of that motion on the real world.

For raster scan hardware, used for most current AR implementations, the time limit for producing a new synthetic image depends on the rate at which the display device can display new imagery. In a VST system, the time also depends on the video acquisition rate. Ideally, the AR system would generate new images (including the digitization of the video data in VST) fast enough so as not to miss data from the video stream or to have the display device present the same image twice. If too many frames of video data are missed, then the user will notice delay between the sensation of head motion and the apparent motion of the world (i.e. changed imagery of the world in the display). If the synthetic image generation consumes significant time, this may cause loss of input video frames. This can limit the complexity of the synthetic imagery. As noted, current display and acquisition systems for AR are rater-scan devices. Current technology for both display and acquisition is raising the speed. Currently available equipment can acquire and digitize video at 1000 Hz [AOA98], although I have not heard of this equipment being used for VST. Using it would require resolving issues of brightness and image quality.

Delay can also be introduced by the tracking subsystem. If the data from the tracking subsystem is not timely, then there will be relative latency between the apparent motion of the real world and the apparent motion of the synthetic imagery. This type of latency can be managed [Jacobs97] by calibration and techniques such as interleaving data acquisition and computation, predictive tracking [Azuma95], and in cases where the need for spatial alignment justifies allowing delay, by interpolating past data [Jacobs97].

Delay in presentation of images and in tracking subsystems can be very disturbing to the user, to the point of inducing physical sickness. Users can tolerate some latency, however, depending on the user and the application. In viewing a dynamic stream of ultrasound data in order to perform amniocentesis, for example, latency of the data directly affects the confidence with which the physician can insert the needle. While not optimal, when visualizing a static data set, the physician could wait for the image to “settle in” to place, then move slowly to view the data from different angles. (Slow head motion will limit the displacement on the image plane.)

#### 2.1.1.5 Other Issues in Merging Imagery

There are numerous other cues that our visual system uses in order to perceive shape and depth, including stereo, lighting, shadows, and focus (accommodation). Any mismatch between these cues for the real and virtual objects will diminish the illusion that the synthetic objects exist in the same space as the real objects. Despite the knowledge of how to compute and render computationally expensive features such as depth of field and lighting, these features are not nearly as critical as other issues, and so have yet to be incorporated in many AR systems. However, there are systems that do compute these effects. Real and synthetic objects can collide in the merged environment. This physical event should be computed and a realistic response should be displayed [Aliaga94]. One non-real-time system computes many types of interactions such as inter-reflections between real and synthetic objects, shadows, and collisions [Jancène95]. By combining image-based rendering techniques with traditional geometric and material modeling, one can realistically render synthetic objects into real scenes [Debevec98]. With computer vision techniques and ray tracing [Fournier95] or 3D scene calibration and radiosity preprocessing [Loscos98], accurate shadows can be pre-computed, then displayed at interactive rates.

### 2.1.2 Applications of Augmented Reality

The range of applications that could potentially benefit from this kind of technology is wide and continues to expand as the hardware used to build such systems improves. Numerous proof-of-concept systems have been demonstrated, though as noted, few systems are currently used in place of the traditional methods to accomplish the respective tasks. This section will describe several types of applications and specific examples of these types. The types of applications are ordered by increasing requirements on registration accuracy.

#### 2.1.2.1 Annotation

Textual annotation is perhaps the simplest method of augmenting the real world. Such annotation might take the form of a head-up display for pilots [Furness86], automobile drivers [Weintraub92], or manufacturing workers [Caudell92]. In these applications, however, the placement of the synthetic imagery within the display or world is often arbitrary, significantly reducing the complexity of the problem. More sophisticated systems might guide the user through a library [Fitzmaurice93], label the parts of an engine model [Rose95] or computing equipment [Kakez97], or display personal notes [Feiner93]. Such applications place only loose restrictions on the placement of the synthetic imagery within the world, requiring only that it be aligned closely enough to convey the necessary information. In fact, some systems that simply provide textual information prefer to place the synthetic imagery out of the user's central field of view, which presumably contains the real objects that the text discusses. Registration error on the order of several centimeters in world space or several dozen pixels in image space is acceptable. However, the merging might have to occur over a large range of viewpoints and view directions.

#### 2.1.2.2 Repair, Manufacturing, and Design

From applications that merely place labels in the world, it is but a small step to use such labels to perform tasks. Indeed, if labels constitute user manuals, then the user should be able to perform simple manipulation tasks with the assistance of synthetic imagery. The system can go beyond textual labels to providing synthetic images that represent a sequence of locations through which a real object should be manipulated or the final arrangement of a set of real objects. This idea has been applied to laser-printer maintenance [Feiner93] and computing equipment [Kakez97], and suggested for automobile engines [Tuceryan95].

A similar suggested application [Drascic93] is to have the user manipulate the synthetic object in order to plan a task that will subsequently be performed with the real object. This basically shifts control of the synthetic object from the application to the user, but still allows the application to constrain the object motion to physically realizable situations.

It is another small step from instructions for repair to instructions for building a product. A system at Boeing [Sims94] guides manufacturing workers through assembly of wiring harnesses. This system also deserves credit for being placed in the hands of assembly workers. However, none of the wiring bundles constructed with this system are in use in planes. Still, this is among the most advanced AR systems, in terms of being closest to being used for "real" work. According to one of the designers, the system merely "needs some engineering details worked out before it's deployed on a large scale in the Boeing factory." [Mizell98]

A variant on the same idea is to show the location of hidden objects such as the infrastructure of a building [Feiner95, Webster96]. The ARGOS [Drascic93] system tackles such tasks under difficult viewing conditions, including viewing of a space shuttle bay interior. Designers could visualize a proposed room arrangement [Klinker97] or even a proposed renovation to a building.

#### 2.1.2.3 Entertainment

Another application of AR is entertainment. While VE have often been suggested for entertainment applications in the form of video games, AR offers the advantage of the ability to use real objects as part of the game. One example of this is the AR air hockey game [Ohshima98] in which the players play on a physical table with physical mallets, but a synthetic puck. This would benefit greatly from force feedback devices. This application also emphasizes the collaborative potential of AR.

Another entertainment application is that of virtual studios. This production technique combines real video of actors or hosts with synthetic imagery of a setting or studio. This can be a less expensive form of building a unique environment for a live television production. This application is usually made easier by always having the real imagery appear in the foreground–i.e. in front of–the synthetic imagery, eliminating the occlusion problem. Also, the real imagery is generally of actors standing in the midst of a large synthetic studio. Thus an alignment error of even several feet is often not significant.

#### 2.1.2.4 Medical Care

Perhaps the AR applications with the most powerful potential are in the medical field. Surgical simulations might help teach anatomy [Rolland97] or identify organs and specify objects to avoid [Durlach95 (Chap. 5)] by giving a physician or medical student “X-ray vision.”

Perhaps the most challenging set of applications, however, tries to assist in a surgical procedure. AR can be used to display features visible in standard medical imagery such as computed tomography (CT) or magnetic resonance imaging (MRI) that are not visible to the naked eye or require lengthy training to identify. It also offers the potential to perform the difficult task of mentally reconstructing the (often) inherently 2D standard imagery into three dimensions automatically. Currently, the physician must learn to perform this task mentally. The AR system can further display the resulting 3D imagery in the proper place with respect to the exterior anatomy of the patient. This should provide a more direct and more natural interface to the procedure, similar to the look of open surgery, without requiring the risks of open surgery. It could also allow more efficient use of non-real-time medical imagery such as CT and traditional MRI. These features have yet to be proven, but these systems described below have demonstrated the potential.

**Application to MRI and CT Visualization** Three systems have combined synthetic MRI or CT imagery with views acquired from an operating microscope. Two independently developed systems aided the physicians in planning and performing the delicate neurosurgery interventions [Kelly86, Lorensen93]. Work continues on the latter project [Grimson95, Mellor95]. Figure 2.3 shows the work of the most recent systems

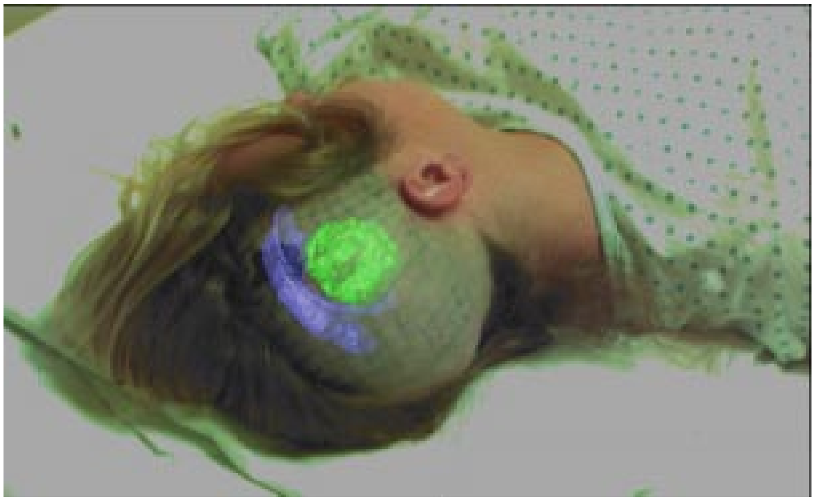
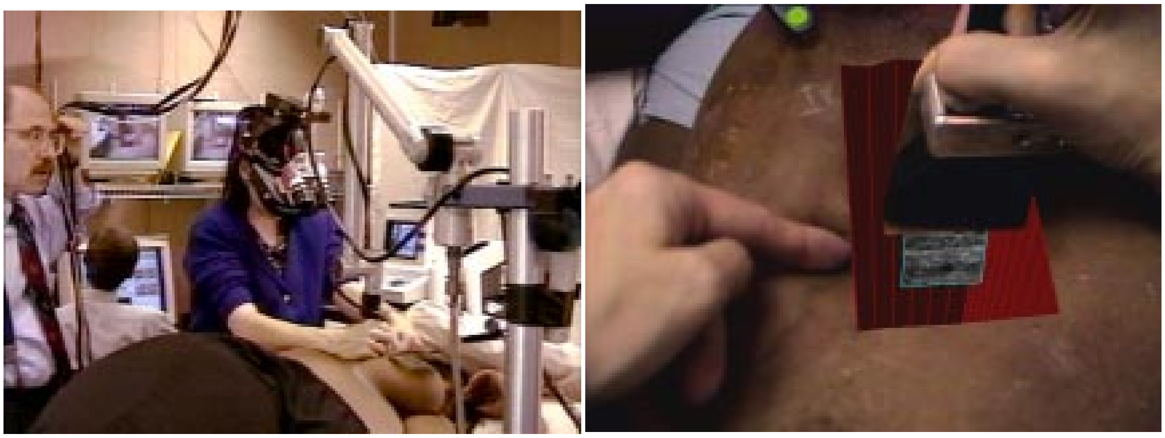


Figure 2.3: Image from the AR system for performing neurosurgery, developed at the Massachusetts Institute of Technology. In this system, pre-acquired and segmented CT data is overlaid on the view of the patient in order to guide the surgeon. Image courtesy of the Massachusetts Institute of Technology Artificial Intelligence Lab and Brigham and Women's Hospital Surgical Planning Lab.

developed at the Massachusetts Institute of Technology. A similar system was applied to both neurosurgery and otolaryngology [Hawkes97].

These three systems are notable because they are in fact being used by physicians to provide treatment for patients. However, in order to achieve this goal, these systems have used less demanding configurations of the AR technologies than the description given in Section 2.1.1. The displays used in these systems are not HMDs, but rather a conventional monitor or an operating microscope. Neither of these displays move significantly or often during the procedure, simplifying the registration task immensely. Also, these applications can make assumptions about the environment. For example, the patient in neurosurgery is literally fastened to a stereotactic frame, which is in turn fastened to the table, to prevent movement during surgery.

Registration of real and synthetic imagery in neurosurgery is performed by manually aligning the stereotactic frame to the table. Laser measuring systems are used to have a repeatable location for the frame relative the the microscope. In AR otolaryngology, the registration is assisted by a conventional optical tracking subsystem. This decreases the accuracy of registration compared to the laser system, but not by a significant amount. Also, the microscope does not move much in this application, simplifying the registration problem. Occlusion relationships are not a significant concern since the primary goal of the task is to assist with the drilling into the patient's skull or in peeling back the next layer of the skull tissue; thus the synthetic imagery of the next layer of the medical data can always be placed in front of the real imagery.



(a) (b)

Figure 2.4: (a) Observer's view of the UNC AR ultrasound visualization system in use. The physician is examining the patient prior to successfully inserting the needle into the target lesion. (b) View presented to the physician through the HMD at the time the observer's view was taken. Note the alignment between the physician's finger, positioned with tactile feedback, and the lesion, which is visible as the dark circle in the lower-center region of the ultrasound data beneath the ultrasound probe.

**Application to Ultrasound Visualization** The group in which I have been working at UNC, under the direction of Prof. Henry Fuchs, has been working on AR ultrasound visualization since 1990. The principles are the same as those used in the overlay of CT or MRI data. Applications thus far have consisted of fetal examinations and ultrasound-guided needle biopsies. The ultrasound data is received as a video stream, digitized, and registered in real time to the patient. In order for the physician to be able to view the dynamic data from any direction, the physician's head must be tracked. In order to register the discrete ultrasound data slices to each other, the ultrasound probe must be calibrated to determine the location of the data with respect to the external shape of the probe. In order to register the ultrasound data set to the patient, the probe must be tracked. The use of separate tracking subsystems in the current implementation creates a difficult calibration task.

The visualization of ultrasound data is more difficult since the ultrasound imagery is acquired in real time and the target can move, decreasing confidence in previously collected data. Successive implementations have used real-time, non-reconstructed data [Bajura92], off-line reconstruction [State94], and most recently real-time volume visualization [State96b]. Images of this most recent system in action are visible in Figure 2.4. The patient in this application is allowed to move; she will breathe and can shift her weight for comfort. This creates a demand for real-time depth data to provide proper occlusion. This has not been an significant problem, however, since this motion is small. In order to provide better relative depth cues, we use the metaphor of a virtual pit within the patient's body.

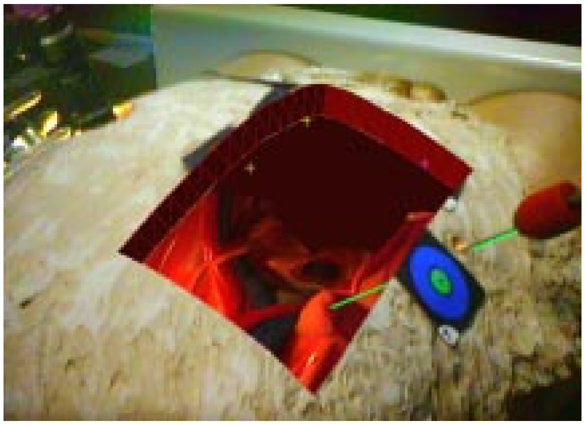


Figure 2.5: View presented to the physician through the HMD in the AR laparoscopic visualization system. This proof-of-concept image was taken early in the work. The AR paradigm is similar, but this application also requires acquisition of a 3D model of the patient anatomy, a very difficult task.

**Application to Laparoscopic Visualization** The UNC ultrasound visualization system is currently being expanded to integrate real-time laparoscopy data [Fuchs98]. A custom-designed adaptation to a conventional laparoscope enables the acquisition of range images of the internal patient anatomy, which are reconstructed into a synthetic model and registered to the patient (Figure 2.5). This application has similar requirements for tracking of the physician's head and the probe, and also has the challenge of a dynamic data set. This is made more difficult by the dynamic nature (due to breathing, blood flow, and patient shifting) of the internal anatomy imaged by the laparoscope.

## 2.2 Tracking

To this point, I have mentioned tracking devices as a particular subsystem of an AR system. However, tracking devices have numerous applications in computer graphics providing interactive control of the viewpoint and of objects. Not surprisingly, there has been a significant amount of research to solve the problem of tracking. Note that I will slightly change terminology in this discussion and use "system" to refer to a tracking device, since I am for the moment not discussing complete "AR systems" as in the previous section.

This section surveys the hardware and software technology of tracking systems. I list in Section 2.2.1 the features that are sometimes desired in tracking systems. Technologies that have been applied to build tracking systems are discussed in Section 2.2.2. Details of algorithms relevant to this dissertation and more recent research on tracking systems appears in Section 2.2.3. Several good surveys of tracking systems are recommended [Durlach95 (Chap. 5), Mulder94, Meyer92, Ferrin91].

### 2.2.1 Properties of Tracking Systems

This section lists properties of tracking systems and discusses the relationships between the various properties.

- **Accuracy**: the degree to which the reported data matches the true values. This assumes the existence of a “gold standard” to which the data can be compared. Accuracy can be very difficult to measure, except by another tracking system which is assumed to be of higher accuracy. Even then, the tracking system that is being tested can only be determined to be “less accurate than” or “at least as accurate as” the second system, but not “more accurate.” Obviously, in general we want as much accuracy as possible. In a practical sense, however, the application determines the accuracy required of the tracking system.

  Accuracy can be limited by several factors. The **measurement resolution** of the system is the smallest possible change in the reported data. Clearly, a system can only be accurate to within its ability to perceive a change in the data. A tracking system might be susceptible to **interference** from the environment. Interference takes many forms, depending on the technology of the tracker. Examples will be given in Section 2.2.2. Fortunately, loss of accuracy due to environmental interference can often be reduced by mapping the interference and providing a corrective inverse mapping [Livingston97].

  Accuracy can be lost due to **noise**, which is unexplained, random, zero-mean error in the data. Of course, more complete characterizations of the physical or electrical behavior can offer explanations of noise. The degree to which a behavior can be characterized determines the degree to which the error it causes can be corrected. For example, electrical interference is frequently considered a source of noise, but once it is modeled, it should be considered a systematic loss of accuracy due to interference. Noise can in effect reduce the resolution of the device by consistently corrupting the low-order bits of the reported data.

- **Robustness**: the system’s ability to continue to function in adverse conditions or with missing or incorrect measurements. Some systems make assumptions about the surrounding environment during operation. Also, a system may be unable to take a measurement at a particular time. Specific examples of these problems depend on the technology and will be discussed in Section 2.2.2.

  Related to the robustness is **repeatability** in the reported data. If the reported values are consistent over time and over operating conditions and environment, then measuring the accuracy (or the lack thereof) is possible, and corrective algorithms can be applied.

- **Range**: the space within which the system can measure enough data in order to compute a result. Sometimes the usable range can be reduced by environmental interference, loss of accuracy, or noise.

One important issue connected to the range is whether it is **scalable**. That is, whether the range can be extended easily with additional hardware and how well the system can continue to satisfy the other desired features with this extended range. Also related to the range is the **number of degrees of freedom** that a system can measure. Some systems can measure only position (three DOF), some measure position and direction (five DOF), while most measure position and orientation (six DOF). Also, systems vary in the **number of targets** that can be tracked simultaneously.

- **Speed**: the frequency at which measurements can be obtained from the tracking system. There are two significant numbers to consider. The first is the **update rate**, which is the raw speed at which the tracker generates new data reports. This may or may not be equal to the rate at which the device makes its raw measurements from the sensors, be they magnetic, optical, mechanical, or any other technology. This "raw measurement rate" is not necessarily a concern to the application; it is the algorithm that generates tracking data from the raw measurements that determines the rate at which data is sent to the application.

  The second number to consider the **latency**, which is the delay between the time the tracked object was (first) at a given pose and the time at which the application on the host computer (first) receives the corresponding data. Both depend on the technology and on the number of new measurements the device must take in order to compute a new pose. As with the accuracy, the application determines the required update rate and maximum tolerable latency. In general, the tracker's update rate must be at least as high as the display rate of the graphics. AR is known to tolerate very little latency before it causes registration error [Holloway95]. Recent algorithms that can reduce the difficulty of using slow measurement technology are discussed in Section 2.2.3.3.

- **Hardware**: the physical realization of the components of the tracking system. The important issues are the number of components, and the size and weight of those components, especially those that the user is required to carry (or wear). Some systems may have a significant amount of hardware that must be set up in the environment, although it may need no further attention from the user once in position.

  Ideally, the application would like to give the user complete **freedom of movement**. Some devices tether the user to a fixed object. Some systems may have a heavy or unwieldy device which the user must manipulate in order to move. Some devices have a tendency to pull the user back to a "resting position" for the device.

Depending on the application, some of these features may be vital, while others may be of minimal concern. In AR applications for surgery assistance, for example, accuracy is of utmost concern while large range is not usually crucial. For an spatial planning application, less accuracy may be acceptable while large range (e.g. the size of a room) is very important. In a flight simulator, a high update rate is paramount because pilots can move their heads very fast, while tethering the pilot's helmet may not be of great concern.

## 2.2.2 Tracking Technologies

From the preceding discussion of the desired features of tracking systems, the reader should begin to realize that it is difficult for any single technology to satisfy all these (frequently conflicting) requirements at once. Most tracking systems are considered to have at best a limited application domain. Numerous methods have been tried to solve the problem. The following discussion of technologies is ordered by decreasing commercial success for AR and VE applications, although optical systems, which are most closely related to this dissertation, are reserved for last.

### 2.2.2.1 Magnetic

Magnetic tracking systems measure the strength of a set of magnetic fields generated by a fixed transmitter. The transmitter and the sensors consist of three coils mounted in mutually orthogonal directions. The sensors range in size, but tend to be around a few cubic centimeters. The transmitters range in size with the power of the field they are expected to generate, and range from several cubic inches to a cubic foot. There are four magnetic fields that must be measured: the environmental field (including the Earth's magnetic field), and three orthogonal fields in the transmitter's coordinate directions. Each of these fields is measured in the sensor's three coordinate dimensions for a total of twelve measurements for each sensor. From this information, the position and orientation of the sensor with respect to the transmitter can be computed [Kuipers80, Raab79].

Magnetic systems are robust, fast, and inexpensive compared to most other technologies, making them one of the most popular choices for both VE and AR [Ascension98, Polhemus98]. However, magnetic trackers are inaccurate in practical environments, due to distortion of the magnetic field caused by ferrous metals in the environment. If the metal is fixed, this is a correctable type of interference [Livingston97]. Most magnetic trackers can track at least several sensors, though for some systems, the communication required for reporting data of several sensors slows the report rate for each sensor.

### 2.2.2.2 Mechanical

Jointed mechanical arms were among the earliest tracking systems [Sutherland68] and continue to be used today. They use a variety of rotary transducers or shaft encoders to track the orientation or position of each joint of an articulated arm. Because each of these measurements tends to be very accurate, the total accuracy of the system is usually excellent when compared to magnetic systems. Early mechanical systems used reels of string together with a rotary measurement to determine radial distance from a fixed point [Sutherland68]. Several measurements can form a three DOF mechanical tracker.

Mechanical systems suffer from limited range and from being able to track only one object. Many are clumsy for tracking HMDs due to the tethering of the user's head to a device that may restrict movement or pull on the user. A notable exception to this is the BOOM, which is an integrated mechanical tracker and display system [Fakespace98]. Two displays are housed in a box that looks like a simple HMD, however, the weight of the display system is supported by the mechanical arm, which is counterbalanced in order to support the weight and protect the display system from falling to the ground or burdening the user.

It is very difficult to use more than one mechanical tracking system, since the arms or strings get tied up with each other. Mechanical systems have found a good commercial niche as measurement devices and hand tracking systems [Faro98, Immersion98].

#### 2.2.2.3 Acoustic

Acoustic systems typically measure the time of flight for ultrasonic sounds and relate the time to the speed for a given room temperature. An alternate strategy is to measure the phase difference between multiple waves. Generally the user carries the transmitter, and a series of sensors around the room determine the linear distance to the transmitter. Some systems have the user carry a receiver and listen to a series of transmitters positioned around the volume. An algorithm akin to sphere intersection can then determine the position of the user's transmitter (or receiver). Other variations are possible, including multiple transmitters alternating the sending of a signal. Acoustic trackers frequently provide only position measurements (three DOF tracking), however, some systems do infer orientation data by tracking multiple points. Acoustic systems are of limited accuracy and speed. They suffer from environmental interference (e.g. temperature variations), inter-reflections, and obstruction between emitter and receiver. Despite this, there are systems that have used acoustic systems for real-time tracking or for measurement [Sowizral93, Logitech98, Foxlin98].

#### 2.2.2.4 Radio Waves

The measurement principles of time of flight and phase difference also apply to radio waves. The Global Positioning System [Parkinson96] is an example of this type of system which allows tracking over the entire surface of the Earth, so long as a sufficient number of the orbiting satellites that transmit the radio waves are within range. Until recently, commercially available hardware was limited to an accuracy of approximately 100 meters. However, researchers found ways to work around the military's encoding of time stamps to achieve accuracy reportedly on the order of a few millimeters [Herring96]. Indoor systems based on time-of-flight measurements of radio waves have also been suggested. However, the accuracy and speed of measurements are likely to be insufficient, and the inter-reflections could pose a significant problem.

#### 2.2.2.5 Inertial

Rate gyroscopes and accelerometers directly measure velocity and acceleration. If the initial pose is known, then the equations for rigid-body kinematics will compute the current pose. This formulation with inertial sensors can measure only the change in position and orientation. To provide a complete 6-DOF tracking system, we require an initial estimate, which we must obtain with an alternate technology. When used in combination with another technology, inertial sensors provide a fast and accurate system. Such multi-sensor tracking systems are discussed in Section 2.2.3.4. Because they directly measure velocity and/or acceleration, inertial sensors are well-suited to predictive tracking schemes [Azuma95].

The major drawback of inertial sensors is that they can also make relative measurements. Thus they accumulate error over time, and appear to "drift" in space. The acceleration due to gravity must be subtracted from the readings of an accelerometer. This is non-trivial; it requires calibration of the relative orientation of three accelerometers, and it requires that each accelerometer have a large dynamic range in order to not be overwhelmed by acceleration due to gravity and still be able to discern subtle changes in acceleration. Commercial systems are just beginning to include inertial trackers [Intersense98, Foxlin98].

#### 2.2.2.6 Optical

Perhaps the oldest form of optical tracking is "navigation by the stars." The basic notion is that if a sailor sees known constellations whose positions and orientations in the sky are known, then he can compute his position and orientation on the globe. A variety of modern technologies have been applied, but in all systems, the mathematical principle that places constraints on the pose of the user is triangulation.

I shall categorize optical systems in two ways, and then place the current work in this context. The first delineation between optical systems is the sensor technology. It can be one-dimensional (1D) or 2D, and may or may not form an image. If the imaging technology is 2D and image-forming, then the system is frequently referred to as vision-based, especially if the same images that are used for tracking are also used to provide the user an image of the real world, as in VST AR. The second difference is the type of landmarks that are employed. Landmarks can be broken into categories of active and passive. The former category generally implies light-emitting diode (LED), while the latter includes a variety of features, such as colored shapes, reflective markers, or even natural features like edges and corners. Various combinations are represented in Table 2.1.

| | Landmark type | |
|---|---|---|
| Sensor type | Active | Passive |
| 2D image-forming | [NDI98]<br>[Selspot98] | [State96a]<br>[Neumann98]<br>[Koller97]<br>[AOA98] |
| 2D non-image-forming | [Ward92]<br>[Welch96]<br>[Kim97] | |
| 1D image-forming | [Fuchs77]<br>[IGT98] | |
| 1D non-image-forming | | |

Table 2.1: Fitting together sensor and landmark technologies to build optical tracking systems.

This work uses 2D imaging technology and projective geometry to create a 1D non-image forming sensor. The landmarks in this work are qualitatively different from the systems cited here. The systems discussed here use physical landmarks placed in the scene, whereas my thesis is that using reflections of light for landmarks with unknown positions can be used to track a moving sensor. The physical manifestations of landmarks in the new system are passive in the sense that they merely reflect light. But the landmarks are projections of light and could also be considered active in the sense that they move and flash under control of the tracking system's host.

Optical systems are among the most accurate, along with mechanical systems, if not more accurate. As noted in Section 2.2.1, it can be hard to measure the accuracy of such systems. Depending on the sensor technology, optical systems can also be among the fastest systems. For example, the UNC opto-electronic ceiling tracker uses lateral effect photo-diode (LEPD) and runs at speeds of up to 2000 Hz [Welch96], while systems that use charged-coupled device (CCD) are limited to the speed of standard video signals, which is usually 30 Hz in the United States, although faster systems are becoming available [AOA98].

One problem with conventional optical approaches is that the landmarks impose constraints on the user's interaction with the world. The user must keep the landmarks in view in order to benefit from the vision-based tracking, but must avoid them while performing the task at hand. As noted above, it is not always practical to assume the environment to be static. Also, the user cannot be expected to keep the landmarks in view as his viewpoint changes. This can lead to occlusion of the landmarks from the user's view. The problem can be reduced by using numerous landmarks, but this is not always a practical solution. A primary goal of this research is to overcome the limitations imposed by the use of physical landmarks and develop a real-time tracking algorithm using projected light.

There are other issues surrounding optical tracking systems that are yet to be answered for our new algorithm. Optical systems frequently place restrictions on the lighting of the scene. There have frequently been questions raised about the stability and repeatability of measurements, especially in 2D image-forming devices. Systems that distribute fixed landmarks around the environment and track sensors that move within the volume suffer from poor accuracy of position measurements. Systems that distribute fixed sensors around the environment and track moving landmarks suffer from poor accuracy of orientation measurements of the assembly carrying the landmarks. The proposed system more closely resembles the former category, since the landmarks are fixed instantaneously to the environment.

Optical tracking methods have been used to track the user's head pose or the structure of a scene, but in a relative sense only. That is, either the landmarks or the cameras are assumed to be static, and the other can therefore be tracked relative to the static object. The "tracking" of the scene, or more precisely, determination of the structure of the scene, is a classic problem in computer vision and outside the capabilities of conventional tracking systems. One recent development counter to this is the UNC optical ceiling tracker with the algorithm described in Section 2.2.3. It has demonstrated the ability to refine gross errors in the estimated locations of its LEDs, and the algorithm could be extended to compute LED positions without use of a prior estimate.

The dual nature of the problems of recovery of camera pose and recovery of scene geometry should be clear, however. Computer vision algorithms will be discussed in Section 2.3. Computer vision has also offered algorithms to compute both tracking of cameras and scene structure simultaneously. This is considered a very difficult task, but a possible extension to this work that moves toward a solution for this task is discussed in Section 5.2.

### 2.2.3 Related Tracking Algorithms

In the previous section, I gave a description of a variety of tracking technologies. Here I concentrate on algorithms that pertain to the current work. Most optical systems apply the geometric concept known as triangulation. One example of this idea is known as collinearity and was implemented for the initial version of the UNC optical ceiling tracker. The current algorithm for the new UNC optical ceiling is known as SCAAT, which uses the concept in a somewhat different manner. It has been postulated that hybrid tracking systems will further the capabilities of tracking systems in the future, and preliminary work backs up this claim.

#### 2.2.3.1 Optical Triangulation

The basic principle of optical triangulation is that if you have two views of a scene and can establish correspondence between points that are visible in both images, then you can determine a variety of parameters of

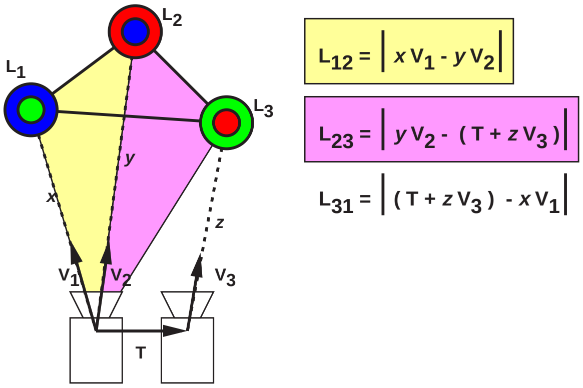

$L_2$
$L_1$
$L_3$
$y$
$x$
$z$
$V_1$
$V_2$
$V_3$
$T$
$L_{12} = \left| x V_1 - y V_2 \right|$
$L_{23} = \left| y V_2 - (T + z V_3) \right|$
$L_{31} = \left| (T + z V_3) - x V_1 \right|$

Figure 2.6: The triangulation algorithm used in the UNC vision-based tracking system [State96a]. The landmark locations $L_1$, $L_2$, and $L_3$ are known, and thus so are the distances between them. We can also express the distance as a distance between two vectors at some unknown depths $x$, $y$, and $z$ along those vectors. The view vectors $V_1$, $V_2$, and $V_3$ are known in the camera coordinate system (calibrated cameras) and the transformation $T$ between is known. The only remaining unknowns are the six degrees of freedom between the world (tracking device) coordinate system and the camera coordinate system. These parameters determine the view vectors in the tracker's coordinate system. We can solve the quadratic equations on the right for these six parameters and the three depths along the view vectors.

the two images, including the offset in position and orientation between the two cameras, the parameters of the projection operations performed by the cameras, and the precise locations of the points in space. There are two ways constraints can be placed on the pose of the imaging device, which are dual to each other.

1. The correspondence of two images of a single static feature defines a triangle with the feature and the two centers of projection of the imaging device as the corners. In this case, the length of the side of the triangle between the two centers of projection is known.

2. Images of multiple landmarks from one pose forms a triangle with an apex at the center of projection and a known length of the opposite side (the relative location of the landmarks).

Either of these basic facts can be used in numerous ways to create a system of constraints involving the parameters of the pose of the camera or system of cameras. Numerous vision-based tracking systems have recently appeared in the literature, using as landmarks LED [Bajura95, Madritsch96], colored circles [State96a] or colored circles and triangles [Neumann96], or black squares [Koller97]. An example of a triangulation algorithm is given in Figure 2.6.

Another variation is to determine the direction cosines of a landmark with respect to a photodetector. This approach is used in the UNM tracker [Kim97]. Using the direction cosines, a series of tetrahedra can be formed. This is similar to setting up a series of triangles which locates the aperture point of the photodiode. A minimization procedure helps to choose from among the various solutions that are possible.

#### 2.2.3.2 Collinearity

The optical ceiling tracker [Ward92] demonstrated at UNC relies on knowing the 3D locations of a set of LEDs mounted on the ceiling and imaging those LED with a set of LEPD fixed on the user's head. The collinearity algorithm [Azuma91] used in the initial version of the UNC optical ceiling tracker, is, according to its authors, "nothing more than similar triangles." The algorithm expresses the observation that the vector in the camera coordinate system from the front principal point of the LEPD to the LED must be collinear with the vector from the rear principal point to the image location of the LED. For a set of LED sightings, this places a system of constraints on the pose of the camera, which can then be solved for the parameters of camera pose[1].

#### 2.2.3.3 Single-Constraint-At-A-Time Tracking

The formulation of the constraints in collinearity assumes that the optical sensor does not move during the time it takes to image sufficient landmarks to fully constrain the pose. However, this condition cannot be guaranteed in practice. To overcome this requires a filtering mechanism that can integrate a set of measurements made over time. This need and a desire to provide a high update rate for an accurate tracking system inspired the idea of SCAAT. This algorithm has been applied to a moving LEPD assembly attached to the user's head and pointed towards a ceiling with LED at known locations. One of the novel features of SCAAT is to be able to integrate a single new constraint to update a previous estimate of the pose. SCAAT uses a predictor-corrector framework to accomplish this.

The same concept that relates the parameters of the camera pose through multiple images of a single feature can also be applied to predict the image location of that feature, given the last known (or estimated) camera position and orientation, velocity, and acceleration, by the equations of rigid-body kinematics. If the feature does not appear in the predicted location, then the estimates for the parameters must be incorrect (or out of date).

The Kalman filter (KF) [Kalman60, Brown92, Welch95] provides a minimum mean squared error estimate to a set of parameters which affect a series of measurements. The filter maintains a current estimate of the values of interest ("the state"), their error covariances, and a model of the behavior and noise of the system over time. "Behavior" in this sense includes a function that determines the measurement from the state and a method of mapping the measurement onto the state.

The KF operates in a predictor–corrector fashion. The prediction step consists of pushing the state and error covariance matrix forward in time according to the system behavior model. From this predicted state, the

[1] In reality, the implementation uses not a single camera, but multiple LEPD. The mathematical constraints are the same, however, since the relative poses of the LEPD are known.

filter predicts a single measurement, then takes a measurement and computes the error. This error is mapped back onto the state to determine a corrective update to the state and the error covariance matrix. This is similar to adding a new constraint to a system of constraints and computing a new solution, except that the computation is not done in batch, and thus the previous constraints need not be retained in memory, only the previous estimate. The basic KF works only for discrete, linear processes. However, an extended Kalman filter (EKF) applies to a continuous, non-linear process with discrete measurements. In this case, the mapping of the error onto the state is determined by the Jacobian matrix of the measurement function.

A SCAAT filter based on an EKF is simple in implementation, and has several benefits. It allows fast refinement of a current estimate, which provides a high update rate. A high update rate in turn reduces the time between measurements, thus increasing the likelihood that the current estimate will reflect the true state of the system–i.e. be more accurate. It also allows noise in the measurements to be filtered, though this filtering induces a delay.

#### 2.2.3.4 Multi-sensor approaches

Section 2.2.2 described the limitations of various technologies that have been applied to tracking. However, some systems have complementary sets of limitations. Such systems can be combined into a single system that overcomes the limitations of *each* of the component systems. For example, the UNC ultrasound team implemented a tracking system that used a conventional magnetic tracking system and a custom vision-based tracking system to yield a robust system and that achieved accuracy in the crucial portion of the tracking volume [State96a]. The SCAAT formulation also provides a good framework for integrating multiple data sensors into a single tracking system. Other systems [Foxlin96, Emura94] have used an EKF in order to integrate different types of measurements into a single estimate of a set of parameters and to reduce latency through the inherent prediction mechanism of the EKF.

## 2.3 Computer Vision

The tracking approach introduced in this dissertation can be thought of in two ways. The first is as an optical tracking system with projected landmarks. This is the view given in Section 1.3.1. It largely ignores the view of this algorithm as a solution to a classic problem in computer vision. The second way to think of our approach to tracking is as a solution to the motion-from-structure problem. This problem can be stated as follows.

> Given two (or more) images of a scene and a list of corresponding points or lines, determine the relative camera pose used to acquire the second (and succeeding) images.

The two basic steps of such algorithms are (1) to establish correspondence and (2) to determine offsets with a linear or non-linear system. Several algorithms suggested for this problem have shown to be quite sensitive to noise, and correspondence can be difficult or expensive to establish unambiguously [Faugeras93]. Our algorithm is similar to algorithms presented in the computer vision literature to solve this problem.

In Chapter 3, we present a SCAAT tracking algorithm that uses point-to-point correspondences between two image planes, a constraint that is familiar in computer vision algorithm. For the new tracking algorithm, this means that we need to know what the pattern from the projector looks like in the image plane of the camera. This in turn requires the same projective geometry operations that are used in computer vision to describe the six DOF offset of two cameras looking at the same scene. Thus in the course of discussing the related work in computer vision, numerous concepts and terms from computational projective geometry are introduced.

The problem of tracking the camera with respect to the projector thus reduces to the problem in computer vision of determining the pose (rotation and translation) of the camera with respect to another camera. The constraints are given in the form of images of the same scene from the two cameras–or more generally, the two image planes–in which pairs of features, one from each image, are identified as corresponding to a single object in the environment. “Features” in this case can be points, lines, or arrangements of points and lines. In our work to this time, we have used only points. Indeed it is unclear whether line correspondences could be generated in a useful manner. Thus I discuss only point-correspondence algorithms in detail. In computer vision, the two image planes considered frequently belong to a single moving camera. In our work, one image plane belongs to the tracked camera, and one image plane belongs to fixed projector. Only one projector at a time fires a ray.

Two concepts from computer vision deserve early mention in this discussion. First is a classification of the parameters that affect vision. The position and orientation of a camera relative to some fixed reference are known as the *extrinsic* parameters. In 3D space, this requires six parameters to describe. The parameters of the perspective projection performed by the camera are known as the *intrinsic* parameters of the camera. These can include the field of view, aspect ratio, focal distance, vertical and horizontal offsets between the image plane origin and the central optical ray, and distortion coefficients. Many algorithms assume that the intrinsic parameters are known and previously accounted for in the coordinates—thus designated *normalized* coordinates. Methods to measure the intrinsic parameters have been presented in the literature [Faugeras93, Davies97].

The second concept is that no computer vision algorithm can solve for the six DOF relative pose without some external measurement to account for the scale factor that cannot be accounted for in an image sequence. This measurement can be the absolute distance of the translation (including the distance between a stereo pair of cameras), a distance of two identified points in the world, or the depth to an identified point measured. This ambiguity of scale can be visualized by considering the change in the camera's image if the entire structure of the environment, including the camera's position in that environment, were scaled equally in all dimensions. There would be no change. The miniature camera would capture the same image of the miniature scene as the large camera would capture of the large scene.

### 2.3.1 The Eight Point Algorithm and Essential Matrix

While many iterative, non-linear techniques were developed for this in early work, Longuet-Higgins introduced the Eight-Point Algorithm in a classic 1981 paper [Longuet-Higgins81]. He showed that eight point correspondences were necessary and sufficient to determine the relative pose up to a scale factor with a linear algorithm[2], and gave a formulation of expressing that difference in terms of the *essential matrix*. In his formulation, the depths of the eight points are simultaneously computed. These are also only to within a common scale factor.

The essential matrix can be shown [Faugeras93] to be composed of the rotation matrix and the skew-symmetric matrix derived from the translation vector between the two image planes. This skew-symmetric matrix is the matrix $T$ derived from the vector $t$ so that $Tx = t \wedge x$ for any vector $x$. Note that $T$ is a rank two matrix, and like $t$, can only be determined up to a scale factor. These properties are thus also true of the essential matrix.

Certain geometric configurations of the real world points can produce a degenerate system of constraints on the elements of the essential matrix [Longuet-Higgins84]. Thus the Eight Point Algorithm will fail in these situations, such as when the eight 3D points lie on a quadric surface that passes through the two centers of projection. Degenerate forms of this condition may occur if four points are collinear, six points lie on a conic, seven points are coplanar, or all eight points constitute a cube. Once the essential matrix between the projector and the camera is determined, it can be factored into a skew-symmetric matrix that determines the translation vector and a $3 \times 3$ rotation matrix [Longuet-Higgins81].

Many researchers criticized the Eight Point Algorithm as sensitive to noise. However, by introducing normalization of the image plane coordinates [Hartley95], the conditioning of the constraint matrix is improved. Also, it is necessary to force the essential matrix computed by the Eight Point Algorithm to be of rank two. By forcing the smallest singular value of the computed matrix to zero, a "close" singular (rank two) matrix can be found [Hartley95].

[2] Five correspondences are necessary for non-linear, iterative methods.

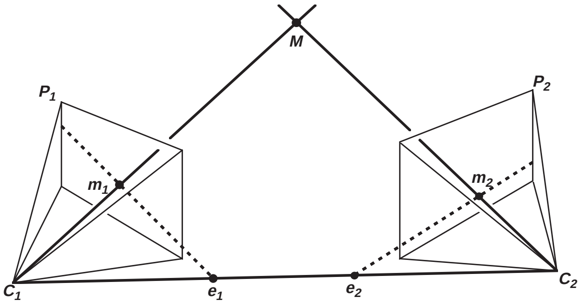


Figure 2.7: The offset between two cameras is given by the epipolar geometry. Using uppercase to denote 3D points and lowercase to denote 2D points, we can express the epipolar constraint. The 3D line $C_1M$ intersects the image plane $P_1$ at $m_1$. The projection of line $C_1M$ onto image plane $P_2$ is the line $e_2m_2$. Similarly, the 3D line $C_2M$ projects to the line $e_1m_1$ in plane $P_1$. Note that $e_1$ and $e_2$ need not be in the visible portion of the image plane.

In order to determine the scale factor associated with the translation, one would need to either be given the scale factor directly or be given a measurement that relates an image metric to a real world metric. For example, the real-world distance of two points known in the scaled coordinates would allow computation of the scale factor. This would in turn require that the image points be identified in correspondence between two images so that the position in scaled coordinates could be computed via triangulation with the estimated camera poses for those two images. Computation of the scale factor requires a calibration on starting the system.

### 2.3.2 The Fundamental Matrix

Faugeras, Luong, and Maybank generalized the work of Longuet-Higgins to include cameras for which the intrinsic parameters are unknown. This removed the restriction that image-plane coordinates be expressed in a normalized coordinate system [Faugeras92]. By composing the essential matrix with a projection matrix that reflects the intrinsic parameters of the projection operation performed by the camera, they create a simple mapping between the pixels in one image plane and *epipolar lines* in the other image plane. They refer to the generalized matrix as the *fundamental matrix* [Luong95] and the collection of epipolar lines and constraints as the *epipolar geometry*.

The *epipole* is defined as the image-plane location of the center of projection associated with the other image plane–that is, where the other center of projection *projects to* in our image plane. An epipolar line is the projection of the ray emanating through a given pixel in one image plane onto a second image plane (Figure 2.7).

Let $M$ be a 3D point imaged at pixel $m_1$ on the first image plane. The *epipolar constraint* says that every 3D point $M_i$ imaged at pixel $m_1$ on the first image plane will lie on the 2D line $e_2m_2$ in the second image plane, where $e_2$ is the epipole and $m_2$ is the imaged location of $M$ in the second image plane. Formulas for the fundamental matrix will be given in the description of the tracking algorithm in Chapter 3.

The basic observation is that if two image-plane points are determined to correspond to the same point in 3D, then each image point will lie on the epipolar line determined by the point in the other image plane. Note that this description is symmetric in the two image planes. The fundamental matrix that maps from the second image plane to the first is the transpose of the matrix that maps from the first image plane to the second. The fundamental matrix, like the essential matrix, is defined only to within a scale factor.

### 2.3.3 Recursive Motion Estimation Algorithms

Several algorithms have attempted to solve the motion estimation problem in a recursive fashion using an EKF or a similar filter.

Since the original Eight Point Algorithm leads to noise-sensitive implementations, you might consider a least-squares method to compute the essential matrix [Faugeras87]. From displacements of points and lines on the image plane, you can compute the camera motion, then "back-solve" for the static scene structure. For the case of point correspondences, Faugeras et al minimized the vector norm of the constraint matrix Longuet-Higgins had used. In this way, they had a framework in which any number of measurements could be used to determine the essential matrix. They also used the analogous system for line correspondences and applied an EKF to minimize the mean-squared error of the predicted and actual sequences of rotations of the camera. Their "measurement" in the EKF formulation was a distance metric between the measured lines and the predicted lines. Using the values computed for the rotations, they solved for the translation using a simple linear algorithm.

Viéville and Faugeras used line matches amongst three frames to estimate camera displacement and scene structure in a static environment. They used a measurement equation that states that the three views of a line must be collinear when expressed (i.e. rotated) in a single coordinate system, say that of the second of the three views [Viéville90]. They minimized the Mahalanobis distance of the measurement function that weights each measurement with its covariance matrix. Instead of an EKF, they used a non-iterative numerical method. The previous "second-to-third view" matrix and the new "first-to-second view" matrix were combined into a single estimate for the rotation parameters. (They again were directly measuring only the rotation.) The translation was computed after the rotation was known. Finally, they computed the structural parameters of the lines (again assuming that the real world lines are static).

Jezouin and Ayache applied an EKF to use detected and predicted locations of both points and lines to estimate camera displacement and scene structure over an unknown, static environment. Their measurement equations explicitly account for the projection matrix, but they do not attempt to estimate the parameters of this projection [Jezouin90]. They also fused points and lines into line segments when possible and use line segments as features.

The *essential filter* enforces the epipolar geometry on the estimated camera pose [Soatto94a]. This filter is similar in structure to an EKF. However, Soatto et al did not use the EKF to directly estimate the motion parameters. Rather, they used it as a smoothing operator on the parameters of motion. At each time step, they computed the difference pose, determined by the essential constraint and expressed by an essential matrix, to update the estimated pose. They extended this idea [Soatto94b] to account for the intrinsic parameters, implying use of the fundamental matrix. The difference fundamental matrix is fundamental by construction, thus the structure required need not be explicitly enforced. They had previously applied the same approach of measuring a difference pose using optical flow to update an estimated pose [Soatto93]. All of these approaches need an external measurement at one time instant, which can then be propagated to other time steps, to compute the missing scale factor.

Azarbayejani and Pentland used an EKF to estimate motion, pointwise structure, and focal length. Their measurements were the image locations of the points. They also directly measured the incremental rotation, since this is better approximated by a linear function (as required in the EKF formulation) than is the complete rotation [Azarbayejani95]. This prevents the equivalence of $360°$ to $0°$ from invalidating the linearity assumptions. The complete orientation is maintained externally. To account for the unknown scale factor, they fixed the depth of a single point.

### 2.3.4 Structured Light

The current work proposes to determine correspondences by a simple variant of structured light–that is, a single dot. Structured light, however, has long been used in computer vision to acquire depth information [Besl89, Daley95, DePiero96], dating back to at least 1932 [Schmaltz32]. Note that this is before the advent of computers. In fact, structured light is rumored to have a 200-year history, although this is not verified [Daley95].

A variety of patterns have been tried [Besl89], including points, lines, multiple points, multiple lines, grids [Bhatnagar91], circles, cross-hairs, thick stripes, coded binary patterns, color-coded stripes [Chen90], and random textures. Pseudo-random binary arrays [Lavoie96] are grids with recognizable points based on a pattern of “large” and “small” intersection points. Patterns can be time-encoded as well [Daley95]. Hybrid vision techniques [Nandhakumar92] can use the strengths of different forms of structured light [Bhatnagar91] or of structured light and other techniques such as stereo vision [Chen90] to produce systems that improve robustness, accuracy, or both over each method separately, much like hybrid techniques for tracking. In any of these methods, the basic notion is that the light structure establishes a correspondence, which is then used in a triangulation method with a known offset between the projection device and the detector (camera or photodiode).

Structured light has become a useful technique for mobile robots and other situations in which humans will not view the environment. The regular, flashing patterns in structured light tend to disorient a human user. One of the novel aspects of structured light research at UNC is to present *imperceptible* structured light in the visible spectrum. The imperceptibility comes from flashing the pattern and its inverse in rapid sequence. This balances the amount of light across the image and, if the pattern display rate is above the flicker fusion frequency of the observer, then the light will be integrated over time by the human visual system to completely hide the pattern from the user. A camera with its shutter synchronized to the display system, however, could detect the pattern. Since the pattern could be displayed as a binary image, the image processing may be more robust than trying to process imagery of the fully-illuminated environment.

### 2.3.5 Our Approach's Relationship to Computer Vision

Normally, in computer vision, the problem of motion-from-structure is to find the offset between two cameras or between a single camera that moves between acquisition of two images. Thus the correspondence is established between two different views of a single set of 3D landmarks with unknown coordinates. These landmarks must be located in both camera images, and then correspondence established between the two sets of 2D landmarks, or equivalently, between a set of epipolar lines in the second image (determined by the coordinates of landmarks in the first image) and the set of detected landmarks in the second image. Instead of two cameras, however, we use one camera (attached to the user's head in the VST system) and a projector mounted statically (pointing down from the ceiling or at some other known, fixed pose) in the environment.

The epipolar geometry described in Section 2.3.2 is a common approach to placing constraints on relative geometry in computer vision algorithms. Such constraints are used to solve not only motion-from-structure, but also structure-from-motion (the dual problem, that of determining the scene structure from images taken along a known camera motion path), or motion-and-structure (determining both quantities simultaneously).

Our approach to solving the tracking problem can thus be viewed as using epipolar geometry to establish relationships between the camera and projector (Figure 2.8). Epipolar geometry describes the relative pose, particularly how images in the projector image plane will appear on the camera image plane. The epipolar geometry remains the same as in the case of two cameras, however. This is similar in spirit to triangulation algorithms, including the collinearity algorithm that forms the basis of the constraints of the old UNC opto-electronic ceiling tracker [Azuma91] and including the UNC vision-based tracking system for AR using col-

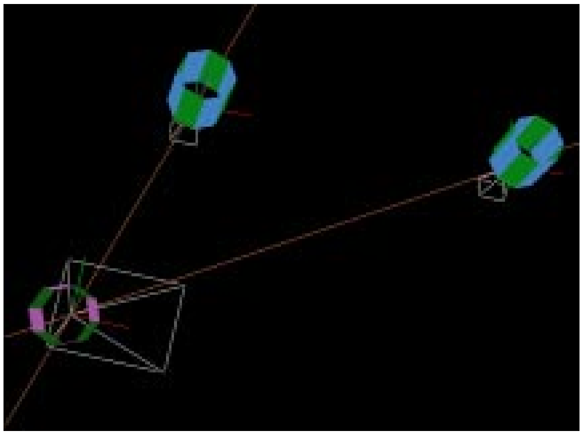


Figure 2.8: The basic design of our algorithm is to constrain the estimated camera pose with epipolar geometry. The lines between the camera model (lower left) and the two projector models represent two pencils of epipolar planes on which the camera lies. More details about the geometric constraints can be found in Chapter 3.

ored landmarks [State96a]. In Section 3.3, I will present the epipolar geometry as it is used in the proposed tracking system.

# Chapter 3

# Structured Light-Based Tracking Algorithm

This chapter summarizes the new tracking algorithm. We can identify characteristics of which we can take advantage in the augmented reality (AR) applications with which we intend to use this new tracking system. These assumptions will refine the nature of the tracking problem and guide our design decisions. Then the algorithm is presented in two stages. First is an intuitive description of the geometric technique for estimating the pose of the camera. Second, the complete algorithm is presented in a sufficiently detailed description, including equations, that the reader should be able to implement the algorithm. Analysis of the performance of the algorithm will be left for Chapter 4.

## 3.1 Characteristics of Our Applications of Interest

Recall the UNC ultrasound visualization system discussed in Section 2.1.2.4. This system is one example of AR in which the user is indoors and working in a small portion of the volume of the room. The same could be said of systems for office equipment repair or certain manufacturing tasks. An example configuration is shown in Figure 3.1. The three characteristics we want to emphasize, in order of most important first, are given below. Along with each characteristic is the implication that we use in the design of the tracking system.

1. **The environment is indoors and under a controlled lighting system.**

   Control in this sense means only that the purpose of the lights is to illuminate the work environment and surrounding areas.

   *Implication* We can integrate the room lighting to generate structured light and illumination for the work environment, as mentioned in Section 1.3.1 and discussed further in this section. Thus we want to convert projectors into "smart" work lights and substitute them for conventional lights. For example, we can fix the projectors in the ceiling much as conventional lights typically are, and have the projectors illuminate the work volume. In order to both light the environment and provide tracking information, the projectors must be synchronized with the camera. This will allow the camera to know when and how long to open its shutter in order to acquire an image for exactly the time interval during which a pattern is displayed, and when and how long to acquire an image during which lighting suitable for performing the task is projected.

2. **The user views the real world through video cameras, defined as a VST AR system.**

   Thus we are already processing a real-time video stream of the environment as viewed from each of the user's eyes. Figure 2.2 presented one implementation that allows the VST AR system to capture such a view. Cameras on an older prototype VST display are visible in Figure 2.4.

   *Implication* We have real-time video imagery from which we acquire not only a view of the real world, but also data that we can use in the tracking system. For example, at UNC and elsewhere, systems have been implemented to track a camera using views of landmarks of known color and shape. These systems were discussed in Section 2.2.2.

3. **The user is interested in adding synthetic objects to only a small portion of the surrounding scene.**

   For example, in the ultrasound application, the patient is lying on an examination table. For repair of office equipment, the machine is frequently placed on a table or comes integrated into a small cart, such as a photocopier.

   *Implication* While the user may move to or look at any region of the environment, the interest in precise registration of synthetic imagery (and thus precise tracking) is limited to a small subset of the possible viewpoints and view directions. Ideally, this limitation would not enforced by the tracking system, but would rather be a natural characteristic of the task to be performed. In practice, however, there are frequently poses the tracking system may be unable to reach physically (e.g. a "stop" in a mechanical arm, or the end of a tether in many magnetic and optical trackers) or cannot provide tracking data (e.g. an optical system missing measurements when not looking at its beacons). We can use features in the application to minimize the deleterious effects of such limitations. For example, if the task is biopsy of a tumor, then there is little interest in augmenting the view of the walls of the room. An optical system would thus not need to have beacons on the walls. The synthetic imagery will consist of medical image data to be aligned with the patient, or perhaps visual cues related to the physician's tray of instruments.

The controlled lighting and small volume of interest imply that we can use conventional projectors to generate structured light patterns over this volume to generate tracking information. The VST paradigm implies that we can acquire these patterns and extract the tracking information, but we will need some method of gen-

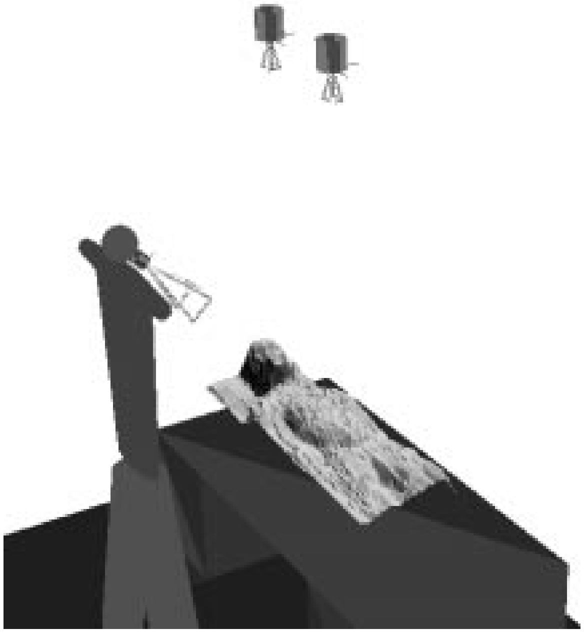


Figure 3.1: A sample environment for which the proposed tracking system is well-suited. In this application, a physician examines a patient lying on the table using AR visualization. The region of interest is small, the lighting under control, and the viewpoints and view directions for which computer-augmented images will be needed are determined by the physician's reach and by the surgical field.

erating correspondence between features in the projected pattern and in the detected pattern. Of course, we must also ensure that humans (whether users of the system or observers) will not be adversely affected by this system. Let's now examine these issues.

### 3.1.1 Applying the Characteristics to Design for our System

Since the region of interest is a small volume, we can aim the projectors at this volume and expect that the user will usually be looking at this volume. Accurate tracking is only necessary when the user is looking at this volume. Thus we want to replace the conventional lights in the environment with projectors, mounted statically in the ceiling, much as conventional lights would be, and projecting structured light onto and near the objects with which registration must be achieved. When the user is looking at this volume, the VST paradigm implies that we will acquire a view of the pattern from the user's viewpoint.

The system will of course allow the user to look anywhere, but it need not provide precise estimates of the camera pose at such times. This attention to a small volume leads us to believe that the projectors can illuminate the environment and provide tracking information. They must light the volume in which the synthetic imagery will be merged, but they don't need to be able to light the entire environment. These projectors will provide at least most of the light for working in the augmented volume, but the environment may contain other, conventional sources of light that are not integrated with the tracking system. This would not prohibit the use of our proposed system, although the tracking system would need to measure the amount of light generated by other sources so that it can be subtracted from the images of patterns that are acquired.

The projectors will give us the ability to control to some degree where the light falls in the scene. When we want to acquire tracking information, we project a pattern of light—for example, a checkerboard or a set of non-adjacent pixels. The appearance of that pattern from the user's viewpoint will tell us something about the user's pose. We can then use this information to establish geometric relationships between the camera's pose and the projectors' poses. The next question to address is what the patterns should be.

### 3.1.2 Choosing Patterns for Structured Light

As noted in Section 2.3.4, using patterns of light has a long history in the field of computer vision, where the technique is known as *structured light*. These methods work by establishing correspondence between landmarks in the projected pattern and in the detected pattern. This is not unlike the problem in the vision-based tracking systems discussed in Section 2.2.2. The systems discussed there and structured light techniques use a variety of methods to identify landmarks.

- color
- local shapes, such as pixels, triangles, or circles
- global patterns, such as checkerboards, grids, or grey codes
- temporal coherence
- dynamic patterns
- time-sequential measurements

In order to be able to recognize the pattern of light falling on the scene, we use *dynamic structured light* patterns. The "structure" we use in the current implementation is pixels. From this point on in the dissertation, we will need to differentiate between three manifestations of the landmarks. In previous systems, the landmarks were always physical objects, and thus were in the 3D environment at which the user looks. Now we have landmarks on the projector image plane, the camera image plane, *and* in the 3D environment. In discussing the algorithm, then, I will refer to "projected landmarks" on the projector image plane, "detected landmarks" on the camera image plane, and "3D landmarks" on the surfaces in the environment.

Single-pixel landmarks are good for projected landmarks since they are easy to identify. Since they are small and symmetric, partial occlusion will not severely change the detected location. We will need multiple detected landmarks in order to compute the camera pose. The question then becomes how to identify the correspondence between the set of projected landmarks and the set of detected landmarks.

Our experience [State96a] and computer vision research [Davies97] has shown that reliably detecting the colors in the environment is difficult. For example, the detection algorithm needs to be robust to specular highlights, which can change the apparent color of the surface that reflected the light. We could also use local shape, including more complex patterns than used in current vision-based systems, although without knowing the surface geometry, it would be difficult to guarantee that the pattern will be recognizable when reprojected (and possibly partially occluded) from another viewpoint. The same argument applies to global patterns, especially when the surfaces are potentially self-occluding.

Since we want to be able to move the landmarks around the 3D environment to decrease the likelihood of consistent occlusion, temporal coherence—assuming that a detected landmark moves by only a small amount on the camera image plane from one image to the next—seems difficult to establish without also knowing the surface motion. Time-sequential measurements—e.g. projecting a single landmark at a time—are more reliable for establishing correspondence. To stay within our design criteria and make it as easy as possible to maintain a real-time system, we have opted in the current implementation to use a single landmark and time-sequential measurements. Thus at a given time step, a single projector displays a pattern that consists of a single pixel. By projecting a single pixel at a time, we avoid having to establish correspondence among several detected landmarks to the set of projected landmarks[1]. The disadvantage of this method is that it yields only a single new constraint at each frame. We observed already that the minimum number of constraints to solve the motion-from-structure problem using point-to-point correspondences is five. Thus in Section 3.2.5, we will present a method to integrate time-sequential measurements with a previous estimate of the camera pose in order to compute an updated estimate of the camera pose.

Projecting a single landmark at a time is merely a choice we have made at the current time; there is nothing fundamental that prevents the system from projecting and detecting multiple landmarks simultaneously. It can be a difficult problem to determine the correct correspondence amongst multiple landmarks. We could use any of the strategies listed above to determine this correspondence. We have chosen what we considered to be the easiest method–projecting a single landmark at a time—in order to simplify the initial system design. In the future, we envision a system that uses temporal coherence, color, and other properties to identify multiple landmarks in a single frame. Then the information at each time step becomes a vector of measurements, each of which is essentially equivalent to the (single) measurement we get now. More complicated patterns such

[1] It is possible that a single projected point landmark can be split over scene geometry into multiple fragments, but the correspondence is still established between all such detected fragments and the single projected landmark.

as lines or polygons could also be incorporated into the framework.

### 3.1.3 Dynamic Structured Light for Humans—Imperceptible Structure

One other concern with this approach is that display of structured light in the visible spectrum can be disorienting to humans in the environment. Structured light in the visible spectrum is used for robot vision, and infrared structured light is frequently used in computer vision and conventional tracking systems, as discussed in Section 2.2.2.

As mentioned in Section 1.3.1, the structure in projected patterns can be made imperceptible with the following strategy. If these patterns are projected for a short time, followed by projection of the inverse pattern for a short time, then the variation in lighting over the scene becomes imperceptible to a human user. Thus the limitation implied here is a *minimum* speed of projection. Fluorescent lights alternate between unstructured light and dark at 120 Hz without (the vast majority of) humans noticing the effect. In preliminary work in our lab [Raskar98], we have displayed simple structured light patterns consisting of pixels, lines, or text at 180 Hz without humans noticing the effect. This has been demonstrated with display surfaces as varied as white walls, human skin, and internal anatomy of recently deceased animals. For all surfaces, we have has success extracting features in the structured light patterns.

We are continuing to improve the system for projecting and detecting imperceptible structured light. Our current focus is to eliminate the need for an encoded identifier in the projected pattern. This is necessary because we cannot yet achieve a consistent elapsed time between the moment we instruct the projector to display a pattern and the moment we receive the image of that pattern from the camera. We are developing a more tightly coupled interface between the projector and camera that can synchronize the two video streams in this manner.

## 3.2 Overview of Algorithm

The data that the structured light patterns generate when viewed from a camera is fundamentally different than the data used in current real-time tracking systems for virtual environments. Recall from Section 2.2.2 that one set of data crucial to most optical and vision-based tracking systems is the 3D locations of the set of landmarks or the set of sensors. Some systems need this data to a high degree of accuracy [State96a]; some need only a good initial approximation of this data [Welch96]. The 3D locations of the landmarks were not on the list of features that we want to use in designing a tracking system for VST AR. In a dynamic environment, it would be difficult to maintain the 3D model of the environment necessary to know the 3D locations[2]. If we are using projected light, viewed from a second (unknown) pose, as the basis of the tracking system, then **we are placing a qualitatively different type of constraint on the user's pose than in previous optical or vision-based tracking systems**.

Whereas previous systems view a set of 3D landmarks in space, the proposed system will see a set of landmarks that lie on a set of rays in space (Figure 3.2). The big difference is that in the proposed system, the 3D locations of the landmarks along the respective rays is unknown; it is only known that the landmarks lie on the rays. This is a fundamentally different geometry than that experienced by systems that see a set of known 3D landmarks. We need to develop some intuition about this geometry, from which we can go on to express geometric relationships regarding the projectors and the camera.

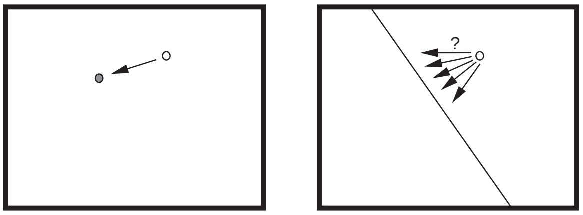


Figure 3.2: The geometry experienced in the new tracking algorithm is fundamentally different than that experienced in current vision-based optical tracking systems. Current systems see landmarks that are at known 3D locations and thus can know the correct 2D location of landmark in the image plane of the sensor (left). The new algorithm does not use the 3D location of the landmark, only the 2D location of the projected landmark on the image plane of the projector. This results in the system only knowing the correct location of the detected landmark in the camera image plane to within a line, but not knowing where on that line the landmark belongs (right).

### 3.2.1 Expressing the Geometry of Rays in Space

So the question we must ask is, "What do the landmark correspondences tell us about the camera pose?" To answer this, let us begin by listing what we know about the landmarks. In this sense of "know," I mean that these are quantities for which we could write geometric coordinates.

- the 2D location of the projected landmark on the image plane of the projector
- the ray in world space defined by the projected landmark and the center of projection of the projector[3]
- the 2D location of the detected landmark on the image plane of the camera

[2] That would be a fundamental change to the system, but it is theoretically possible. This variation on the system is discussed in Section 5.2.

[3] Since the landmark on the image plane has a finite area, it defines a pyramid in space, which can be represented by the central axis of that pyramid.

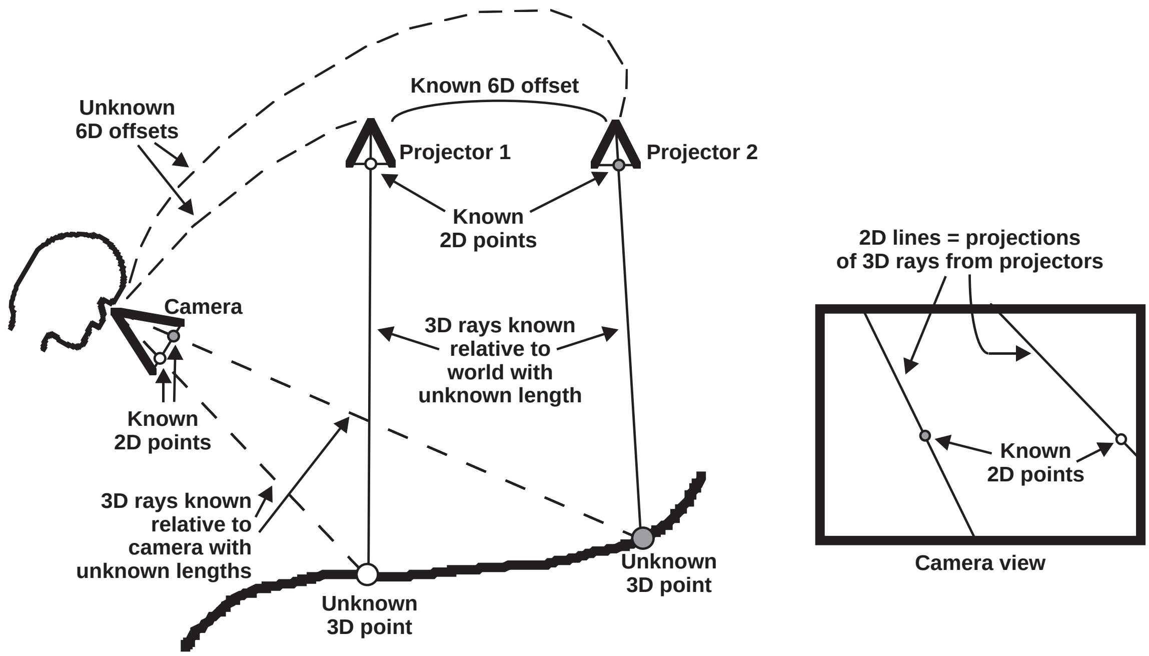

Known 6D offset
Unknown 6D offsets
Projector 1
Projector 2
Known 2D points
Camera
3D rays known relative to world with unknown length
Known 2D points
3D rays known relative to camera with unknown lengths
Unknown 3D point
Unknown 3D point
2D lines = projections of 3D rays from projectors
Known 2D points
Camera view

Figure 3.3: The geometry of detecting landmarks projected onto the scene and viewed by the camera. This geometry generates information about the pose of the camera through the correspondence between landmarks projected on the environment by the projectors and landmarks detected by the camera.

- the ray in camera space defined by the detected landmark and the center of projection of the camera

You might visualize the ray emanating from the projector by thinking about the illuminated cone of light from a car's headlight on a foggy night, although the rays here will have a narrower beam and a sharper fall-off at the edges of the beam. The camera sees the landmark in a 3D location in the environment, although the 3D location is unknown. We do have the 2D location on the camera's image plane. This allows us to define the ray emanating from the camera (Figure 3.3). Note that this ray is only known relative to the camera, as opposed to the ray emanating from the projector, which is known relative to the world since the projector pose in the world is known.

We can now answer the question of what we know by expressing the relationship of the landmarks. **This relationship is the fundamental observation that makes the algorithm work**.

We know that the two optical rays defined by the two 2D locations must meet at the 3D landmark (the surface patch) onto which the light falls. This landmark will project to a small shape on the image plane of the camera, the central point of which will be the point where the ray from the camera intersects its image plane. The ray emanating from the projector will project to a line in the camera image plane. Since the 3D landmark lies on both of these rays, the 3D landmark's projection must lie on the ray's projection in the camera image plane. This expresses again the epipolar constraint discussed in Section 2.3.2.

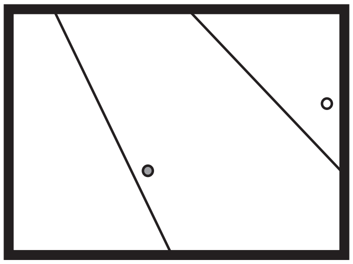


Figure 3.4: We predict lines in the camera image plane on which the landmarks will be detected. If we know the camera pose perfectly, then the landmarks will lie precisely on the respective lines. However, camera motion will introduce error into our estimate, and the landmarks will be separated from the lines by a (hopefully small) distance.

Another observation should be made immediately. We can compute the line on which the detected landmark will lie—if our estimate of the camera pose is perfect. Since this will rarely be the case, the detected landmark will be some (hopefully small) distance from the line. This implies that we have a good pattern with which to search the camera image. We know to start looking in a band around the line, then expand the search outward from there. This can dramatically reduce the image processing requirements of the system, which can be significant without special-purpose hardware.

There are three remaining questions regarding the rays and detected landmarks. The first is "How many corresponding pairs of landmarks do we need to have for this to completely constrain the camera pose?" This question will be examined in Sections 3.2.2 and 3.2.3. The second is "How can we decrease the sensitivity of the algorithm to errors in our estimate of the known geometry parameters?" This will be discussed in Section 3.2.4. The third question is "What constraints must the 3D configuration of the landmarks satisfy to allow the algorithm to compute a solution?" This question requires insights into the mathematical constraints, and will be addressed in Section 4.2.

### 3.2.2 From Rays in Space to Determination of Camera Pose

In Section 3.2.1, we specified a method by which we can generate a single correspondence between the camera and projector. The immediate question is, of course, how many of these correspondences between projected and detected landmarks are necessary in order to completely determine camera pose. To answer this, we should consider what information we get from each such correspondence. We are expressing the epipolar constraint: if we know the camera pose perfectly, then we can identify a line in the camera image plane on which the detected landmark will lie. This was the case for Figure 3.3. Since there will be error introduced in our estimate when the camera moves, the detected landmark will be slightly off the line we compute (Figure 3.4). The information we get, then, is the distance of the center point of the detected landmark from the line.

Several algorithms in computer vision literature, discussed in Section 2.3, use the epipolar constraint to determine camera pose. Since we use the same constraint, the minimum number of landmarks necessary for our algorithm to determine the pose of the camera is five, the same as the number necessary in the general cases of those algorithms[4]. (Certain algorithms use other assumptions, such as that all the 3D landmarks are contained in a single plane, to reduce the number of landmarks necessary.)

In 3D there are six DOF for pose, and we have five constraints. These constraints are scalar; we do not get the distance along both coordinate axes within the image plane, only the distance along the normal to the line. This is because we do not know where along the line the detected landmark should lie. With five scalar constraints, we are clearly not able to fully compute the six DOF camera pose. There is still something missing.

The remaining problem is the concept of *scale*, well-known in the field of computer vision. How is one to determine whether a picture of a house is of a doll house or a human house?

### 3.2.3 Resolving the Ambiguity of Scale

There are a variety of methods that we could use to determine the scale, or the absolute size, of objects in the image. We could place an object of known size, for example a $12''$ ruler or an $8.5'' \times 11''$ sheet of paper, in the scene and detect its boundaries in the camera image. Alternatively, with the tracking equipment we have in our lab at UNC, we could measure the distance between any two 3D landmarks in the scene that are detected in the camera image. Since this would be a one-time measurement upon starting the system, we need not be quite as concerned about the difficulty of keeping an object in view as we are during real-time operation of the system. (Recall that the difficulty of keeping specific objects in view during real-time operation prompted this dissertation on tracking with structured light patterns.)

In a system with two cameras mounted rigidly on the user's head, such as our VST AR system, we could use the distance between the two cameras to measure the scale factor. This would require seeing a single landmark in both cameras' images. This technique is known as *stereo vision* and is very common in computer vision. One of the most difficult problems in stereo vision is automatically determining which landmarks correspond. This is not unlike the situation discussed in Section 3.1.2, in which we explained our choice to solve the correspondence problem by projecting and detecting only a single landmark at a time. Thus this would appear to be a good option for computing the scale. However, we believe that using two projectors is a better choice, for reasons we now explain.

We place at least two projectors at known poses in the environment. In particular, we will know the dis-

[4] As noted earlier, in computer vision, the image planes would belong to two cameras, not to a camera and a projector. The geometry, however, is the same.

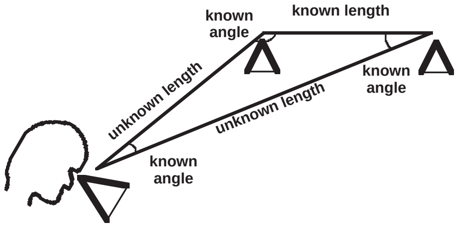


Figure 3.5: Triangulation of the distances from the camera to two projectors from known angles and a known length between the projectors. The known length between the two projectors is used to determine the unknown lengths from the camera to each of the two projectors. The angles are known since the directions in the world are known.

tance between the projectors. This will give us the information necessary to compute the scale factor, which reduces to determining the size of the sides of a triangle with known angles (Figure 3.5). The disadvantage of this approach is that multiple projectors complicate the system hardware. The projectors must be synchronized, and we must measure the relative pose of each additional projector with respect to the first projector. We have chosen multiple projectors, however, largely because this solution has the potential to improve the *stability* of the algorithm. This is discussed further in Section 3.2.4.

To conclude our discussion of resolving scale, we need to consider how requiring two projectors affects the number of corresponding pairs of landmarks we need. We know we need at least five from one projector to compute the pose with respect to that projector. We could use another five from the other projector to compute the pose, and complete the triangle. Another option is to directly generate correspondences between the two projectors and thereby compute 3D locations of landmarks also visible to the camera. This would require two 3D locations for landmarks, which could also be used for corresponding projected and detected landmark pairs; however, it would be very expensive to generate such three-way correspondences. Thus it appears preferable to use multiple landmarks over time, with at least five from each projector, to measure the six DOF pose of the camera.

### 3.2.4 Advantages of Using Two Projectors

As noted in the previous section, using two projectors allows us to determine the missing scale factor. However, there are stronger reasons for choosing a system design involving multiple projectors.

The strongest argument for using two projectors involves the lines that we compute in the camera image plane. If there is only one source (projector) of the rays in space, then for a given pose of the camera, the lines in the camera image plane that correspond to these rays will all look very similar (Figure 3.6). If the camera and the surface both move, for example, in the direction towards the projector within the image plane (left image of Figure 3.6), then the image of the landmarks will not change much. This creates an unstable

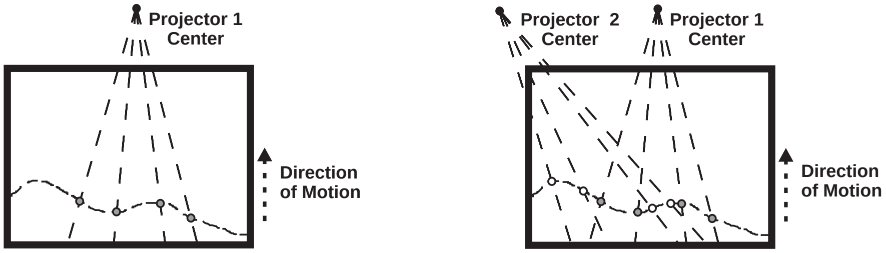


Figure 3.6: In the left image, the camera sees landmarks from only one projector. If the camera and surface move in the indicated direction with respect to the image plane of the camera, then the image of the landmarks will not change much. In the right image, the camera sees landmarks from two projectors. Now the same motion will produce a large change in the image of the landmarks from the second projector, allowing us to determine the camera pose reliably. It would be difficult, though not impossible, to find a direction of motion that would not yield much change in the image of either set of landmarks.

situation for measuring the camera pose. There are other motion paths for which the image does not change. We can improve this situation significantly by using a second projector and getting a very different direction for the lines mapped to the camera image plane. By a “different direction” we mean to imply that the camera (i.e. the user) is in some sense “between” the two projectors[5]. In that case, motion in the direction towards the first projector will create a large change in the image of the landmarks that correspond to the second projector (right image of Figure 3.6). There are still sets of 3D landmarks and motions for which the image will not change very much even for two projectors, but we believe these 3D configurations of landmarks are unlikely to occur once the system is implemented. The shape of such *critical surfaces* is discussed in Section 4.2.3.

Another nice property of using two projectors to determine the scale factor is that this measurement is most amenable to *autocalibration*. This idea is discussed in Section 5.4. Simply put, autocalibration estimates an internal system parameter in real-time. The problem is that there is an unknown interaction between errors in the one-time measurements, not only for the scale factor, but also for the internal camera geometry and internal projector geometry. This relationship is unknown, but autocalibration has the potential to refine the estimate of all the parameters simultaneously. To autocalibrate the scale factor with an object of known size, it would have to always be in the field of view, which reduces us to the primary problem that this dissertation tried to avoid—detecting a physical object in the camera’s view of a small, dynamic environment. To autocalibrate the relative pose between two cameras would be possible, but sensitive to errors since the two head-mounted cameras will see very similar images of the corresponding landmarks and lines.

One question regarding the use of two projectors for resolving the scale is what effect, if any, this has on the *sensitivity* of the algorithm to errors in the one-time measurement (assuming that the autocalibration scheme mentioned above is not enabled). The one-time measurement when using two projectors to resolve

[5] Note that Figure 3.3 was laid out to best express the geometric quantities, not to represent the “best” configuration.

scale would be the relative pose of the projectors. The one-time measurement when using two cameras would be the relative pose of the cameras, and the one-time measurement when using an object of known size would be the localization within the image plane and the size of the object.

### 3.2.5 Integrating Time-Sequential Measurements

In Section 3.1.2, we argued that, for reasons of simplicity, we plan in the first implementation to project a single landmark at a time in order to easily establish the correspondence between projected and detected landmarks. We have said that we want to use multiple landmarks, since we have decided to use two projectors to determine the scale of the world, and it takes five landmarks from each projector. At standard video rates, however, detecting even the minimum number of landmarks (ten) one at a time would require a long time. For example, at the standard video rate of 30 Hz, acquiring ten frames requires one-third of a second. It is unreasonable to expect the user to be still for this long. Even with higher-speed cameras, we could never really eliminate this problem except by solving the correspondence of ten projected and ten detected landmarks. One could also imagine strategies in which a variable number of landmarks, based on confidence in the current estimate, are projected. Correspondence could then be solved with the assumption that the error in the current estimate is bounded. We've already noted that solving correspondence in any form can be a very difficult problem, and thus we have chosen to implement the simplest possible situation of a single projected landmark and "automatic" correspondence.

Any motion of the user while these measurements are taken will introduce error into the computed pose if we compute the pose with an algorithm that assumes the camera does not move while the measurements are taken. (Many algorithms do make this assumption.) Also, recall from Section 2.2.1 that a high update rate is among the features that we desire from a tracker. In VST AR applications, we clearly want the most up-to-date estimate of the pose whenever we are to begin rendering synthetic imagery onto a new frame of video. We thus need a way to combine multiple previous measurements with new measurements that allows for motion of the camera between measurements and computes an estimate for the current time.

A common strategy to performing incremental computations of all kinds is to maintain a current estimate of the solution, then at each step compute an update to the current estimate. A specific class of algorithms that use this strategy is known as *predictor-corrector* algorithms. This strategy appears to be applicable to the problem as described above. Refer back to Figure 3.3. The right side of that figure showed the camera's view of the structured light patterns. This is a view that would occur only if the parameters of the camera pose are correct; only in this case would the detected landmarks lie exactly on the corresponding lines. We can make this our prediction, however, and compute a correction based on the distance of the point from the line. This cycle repeats in order to either further refine the estimate or simply keep the estimate updated while the camera is moving.

We are trying to estimate a set of parameters that could be changing over time. The *extended Kalman filter* (EKF) is a good tool for this type of estimation, in part because it is a predictor-corrector algorithm. There are other good tools available, such as Covariance Intersection [CIWG97]. We chose the EKF because we have had success using an EKF in the tracking algorithm for the UNC optical ceiling tracker (discussed in Section 2.2.3.3). We refer to this as the single-constraint-at-a-time method (SCAAT) [Welch96]. This framework allows us to project and detect a single landmark each frame and compute an updated estimate for the camera pose.

We need a basis for combining the previous information with new information at each cycle of the iteration. To do this requires maintaining certain information from one time step to the next.

- the current estimate of the pose of the object
- estimates of any parameters necessary to predict its behavior over time
- a measure of confidence in the estimates of pose, of velocity, and, if present, of acceleration
- a measure of confidence in the measurements of point-to-line distance in the camera image plane

The measures of confidence are new items; their roles are discussed below. In addition to these quantities, we need to know some other fixed quantities and some relationships between the various quantities.

- a "forward" mapping from the camera pose, location of the projected landmark, and the detected location of the landmark to the measurement of the point-to-line distance

  The mapping we need for projected landmarks is to compute the line equation with the geometry of rays in space, as described in Section 3.2.1.

- geometric information about the projectors and the camera that enables us to compute this mapping of point-to-line distance

  Methods for acquiring this information are discussed in Section 3.3.2. Methods for real-time acquisition of these values are discussed in Chapter 5.

- a mapping from distance between the detected landmark and the predicted line to error in the pose and velocity

  This "inverse" mapping is derived from the forward mapping of the pose and velocity to the distance in the camera image plane, in a way that is defined by the EKF.

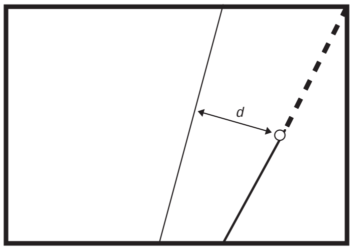


Figure 3.7: In the camera image, we see the projected line (solid line near right) based on the correct camera pose, which is drawn in 2D but aligns with the 3D drawing of the projector ray (dashed line near right). The camera detects the landmark (circle) and predicts the equation of the line based on the current estimated pose (left). The distance $d$ is computed as the shortest distance between the detected landmark and the predicted line. If the estimate of the camera pose were correct, then $d$ would be equal to zero. Contrast this with the camera view depicted in Figure 3.3.

We now begin the discussion of how these quantities and relationships combine to yield a tracking algorithm. We noted above that the predictor-corrector framework has demonstrated successful estimation of pose for the UNC ceiling tracker. Let us now examine the “predictor” phase of one cycle. Simple prediction of the future location of moving objects is something that people do quite naturally, since the mechanics of motion are everyday occurrences. Even when performing an action as simple as crossing the street, we estimate (without consciously thinking) the current location and velocity of an approaching car. We judge very quickly and perhaps to a reasonable degree of accuracy whether we can safely cross the road before the car passes, or whether we are not quite certain that we will make it across, and thus should wait.

In introductory physics courses, this simple model is known as rigid body kinematics. It says in general that from the last estimate of pose and velocity we can estimate the current pose. Returning to the problem of tracking the camera, this implies that if we maintain an estimate of the most recently observed pose and velocity, or perhaps the pose, velocity, and acceleration, we can compute a good estimate of the current pose. We can combine this with our understanding of the geometry of the rays in space emanating from the projector to determine the line on which the currently projected landmark should lie, as described in Section 3.2.2. We compute the distance $d$ of the detected landmark from the computed line, as shown in Figure 3.7.

The following describes the way the EKF works at each time step.

1. Predict the camera pose based on the previous pose and velocity and on the “forward” mapping of the pose and the projected landmark locations.

2. Measure the distance of the detected landmark from the line.

3. Map the distance onto the parameters of pose with the “inverse” mapping to get potential error in the current estimate.

4. Combine the previous estimate with the newly computed potential error.

Note that the computation produces *potential* error in the estimate. “Potential” implies that we are not yet ready to attribute the error we saw between the predicted and actual measurement to error in the parameters we are estimating. We are only stating what might be the case. In the “corrector” phase of the cycle, we introduce the measures of confidence in our estimate into the merging of the potential error with our previous estimate. The potential error indicates the parameters to which the distance measurement is currently sensitive. This sensitivity is measured by the partial derivatives of the distance measurement function. At each step, we know that if we are sensitive to a parameter, then a small change in that parameter will make a large difference in the measurement. If we are unsure about a parameter’s value, then we know that we want to update that value with new information. If we are confident that our current estimate of a given parameter is accurate, then we can dismiss the computed potential error as due to either noise in the detected landmark location or to error in another parameter. We can thus provide a weighting (based on the relative confidence) between the old data and the new data, in which the new data consists of potential changes to the parameters (based on the sensitivity data) and the old data consists of our current estimate updated for time according to the model of dynamic behavior.

One final note about this algorithm is that given the above description, one could imagine complementing the data acquired via detected landmarks with other sources of data, such as accelerometers. Incorporating accelerometers would provide a direct measurement of a quantity that our vision-based system can only estimate. An EKF framework provides an excellent method for integrating measurements from such qualitatively different sensors [Welch96, Foxlin98].

The preceding description gave an general overview of the method chosen to implement the proposed tracking system, in order to enable understanding of how and why the algorithm works. The next section will give the equations of the geometry and of the EKF, with the intention of enabling the reader to implement the algorithm.

## 3.3 A Detailed Description of the Algorithm

With the overview of the tracking algorithm we propose using structured light and the intuition behind it as an introduction, we need to describe the algorithm in sufficient detail that the reader could implement it.

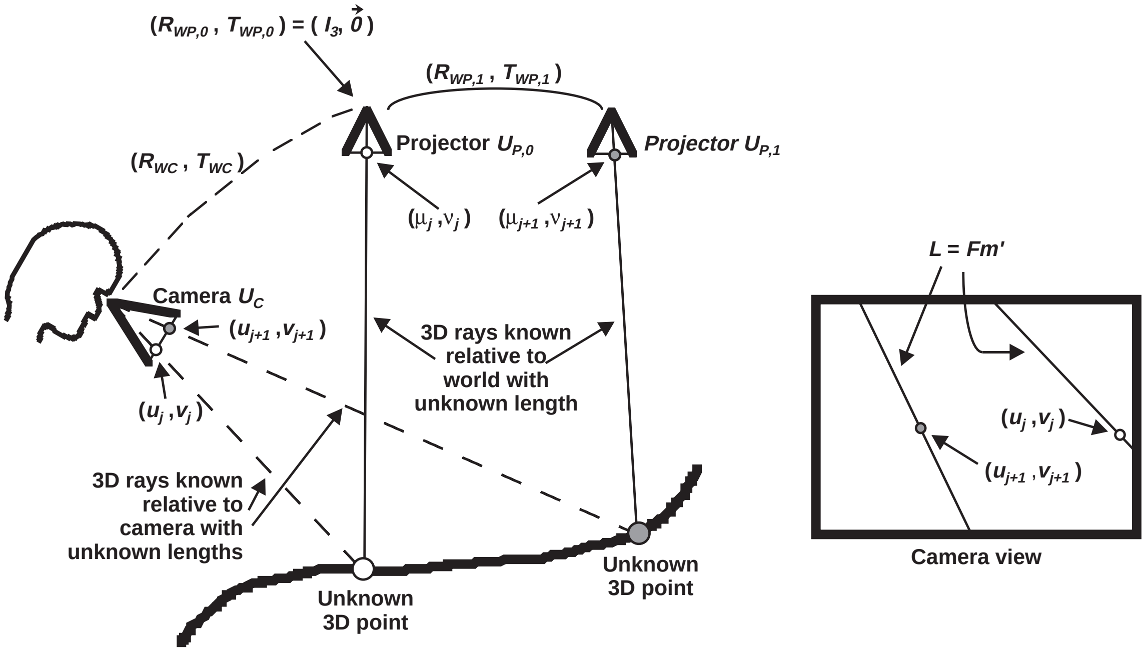


Figure 3.8: The geometry of rays in space, as depicted in Figure 3.3, with the notation introduced in Section 3.3.

### 3.3.1 Geometric Quantities

The quantities and notation that describe the geometry of rays in space as depicted in Figure 3.3 are labeled in Figure 3.8. The system consists of two projectors (although the description scales to any number of projectors), labeled $P_0$ and $P_1$, and the camera mounted on the user's head.

Whenever you want to relate image plane coordinates of a camera or projector that to the 3D geometry of the environment, you need to calibrate the parameters of the projection operation performed by these devices. Methods to do so are discussed in Section 3.3.2.1. This calibration can be expressed with a $3 \times 3$ matrix [Faugeras93, McMillan97] that maps image plane coordinates to 3D coordinates relative to some Euclidean coordinate system that travels with the camera or projector. Denote the projection matrix associated with the camera by $U_C$ and the projection matrix associated with the $i^{\text{th}}$ projector by $U_{P,i}$.

The projectors are mounted statically in the environment. We can express the pose of these projectors in the world with rotation matrices $R_{WP,i}$ and translation vectors $T_{WP,i}$. For simplicity, the first projector can be considered to define the world coordinate system; thus $R_{WP,0}$ is the identity matrix and $T_{WP,0}$ is the zero vector. For simplicity of description, let us assume that all translation vectors $T_{WP,i}$ are expressed in the world coordinate system of the first projector. Measuring these quantities will be discussed in Section 3.3.2.2.

Once we have placed the projectors in the environment, we can begin to project landmarks. We express the landmarks in two coordinate systems, that of the projector image plane and that of the camera image plane.

$R_{WP,i}$ = $3 \times 3$ rotation matrix from the world to $P_i$

$T_{WP,i}$ = translation vector from the world to $P_i$, measured in meters

$U_C$ = $3 \times 3$ matrix description of camera

$U_{P,i}$ = $3 \times 3$ matrix description of $P_i$

$(\mu_j, \nu_j)$ = the (selected) projector coordinates of the $j^{\text{th}}$ landmark, measured in pixels

$(u_j, v_j)$ = the (detected) camera coordinates of the $j^{\text{th}}$ landmark, measured in pixels

$T_{WC}$ = translation offset from $P_0$ to camera, measured in meters

$(\theta, \phi, \rho)$ = incremental orientation offset of camera this time step, measured in radians

$R_{WC}$ = $3 \times 3$ rotation matrix from the world to the camera

Figure 3.9: Symbols used in expressing the epipolar geometry between the camera and the set of projectors.

For the $j^{\text{th}}$ landmark, denote the projected coordinates by $(\mu_j,\ \nu_j)$ and the detected coordinates by $(u_j,\ v_j)$.

The only remaining quantity involved in the operations is the pose of the camera. We can express the camera pose relative to $P_0$ with the rotation matrix $R_{WC}$ and translation vector $T_{WC} = [\ x\ y\ z\ ]^t$, where the superscript $^t$ denotes transpose and vectors are expressed as column vectors. For reasons that will be made clear in Section 3.3.3.4, we will also want to be able to express the detected rotation of the camera since the last update to our estimate. This is a small offset that we can express with Euler angles. Denote the Euler angles of yaw, pitch, and roll by $\theta$, $\phi$, and $\rho$.

These quantities are summarized in Figure 3.9. Using these raw quantities, we will be able to write formulas (in Section 3.3.3.4) for the epipolar geometry of the camera and the various projectors.

### 3.3.2 Calibration of the System

Several of the quantities in Figure 3.3 are listed as known. What this means is that these values must be measured prior to running the system. There are two classes: intrinsic parameters of the camera and projectors, and extrinsic parameters of the projectors relative to each other. Note that the parameters associated with the projectors are not likely to change while the tracking system is running. The intrinsic camera parameters also do not usually change during operation of an AR system. This may be due more to the difficulty of re-estimating the camera intrinsic parameters rather than a lack of applications that need to be able to change the camera intrinsic parameters. One could imagine AR applications in which the camera intrinsic parameters must be estimated in real time. One of the interesting benefits of the algorithmic framework using the EKF is that it provides a method to estimate these parameters in real time. We do not envision this in the first implementation, however. These possibilities will be discussed further in Sections 5.3 and 5.4. For now, we need methods to measure these parameters once in an off-line calibration step.

#### 3.3.2.1 Calibration of Camera and Projector Intrinsic Parameters

The matrix descriptions of camera models can be analyzed in a variety of ways [McMillan97]. One method to describe a symmetric frustum with the view direction perpendicular to the image plane is to build the matrix from three column vectors. The first column represents a step in the image-plane $u$ direction, the second column represents a step in the image-plane $v$ direction, and the third column represents the translation from the camera's center of projection (camera's origin) to the image-plane origin. This method requires three parameters, frequently considered to be field of view, aspect ratio, and focal distance, though other formulations are equivalent. Other parameters may include an offset between the central optical ray and the image plane, an angle between the "horizontal" and "vertical" coordinate axes of the image plane (allowing for deviation from perpendicular, due to manufacturing tolerance), and distortion coefficients.

For a simple three-parameter camera model that consists of the width $w$ and height $g$ of the image plane and the horizontal field of view $\xi$, the matrix $U_C$ is written as follows.

$$U_C = \begin{bmatrix} 1 & 0 & -\frac{w}{2} \\ 0 & 1 & -\frac{g}{2} \\ 0 & 0 & -f \end{bmatrix} \text{ where } f = \arctan \frac{\frac{w}{2}}{\frac{\xi}{2}}$$

In this simple camera model, the only thing we need to compute is the focal length, which we compute from the field of view. We assume that the aspect ratio is equal to the ratio of the image width to image height—i.e. that the pixels are square. Camera calibration is a well-studied and much larger topic in computer vision [Davies97, DePiero96, Faugeras93]. Since the projector is geometrically equivalent, we can use the same principles to calibrate a projector as well. For our structured light system, we selected from the literature calibration techniques based on vanishing points [Wang90, Caprile90].

#### 3.3.2.2 Calibration of Projector Extrinsic Parameters (Pose)

There are many methods that can be applied to measure the pose of the second projector (and succeeding projectors) in the system. Two are suggested here[6]. The first is based on the idea of motion from point correspondences, just like the algorithm for tracking, and the second is similar to the triangulation algorithms used in conventional optical tracking techniques.

To calibrate the rotational and translational offsets between the second projector and the first is an equivalent problem to computing the offset between a camera and a projector, and algorithms such as the Eight-Point

[6] As noted in Chapter 4, the proposed system has not yet been implemented. Thus neither of these two calibration techniques have been implemented.

Algorithm and its variations can compute this offset [Longuet-Higgins81, Hartley95]. The problem is made more difficult by the fact that neither device can be used as a sensor, but the same basic technique of generating correspondences between landmarks in the two image planes can be applied. This implies that the projectors must have some overlap of surfaces that are in their respective fields of view.

In order to generate correspondences, the following procedure can be used. Illuminate a single landmark in both projector image planes. Hold the landmark fixed in the image plane of the first projector and move the landmark in the image plane of the second projector until the centers of the two 3D landmarks are precisely aligned. This will have to be done visually, but once near alignment has been achieved, it can be refined by alternating between projecting only the landmark from the first projector and projecting only the landmark from the second projector. It may be that the number of pixels used to create the second landmark needs to be changed in order to precisely align the centers of the two landmarks, and a temporary physical marker may also be of assistance.

This generates one correspondence. As noted previously, at least five such correspondences are needed to determine the pose to within a scale factor. More landmarks should be used in this one-time calibration to reduce the effects of noise. The scale factor can be determined by measuring the 3D distances between some of the landmarks with a digitizing wand or a conventional measuring device such as a ruler. These measurements should be between two landmarks that are far apart. Again, several such measurements of the scale factor can reduce the effects of noise.

The second suggested method for calibrating the pose of the projector is to use a single digitizing wand as an intermediary device. With such a device, the 3D locations of the landmarks can be determined. Then a standard triangulation-based pose computation [State96a] can be independently performed for each projector. These algorithms generally require three landmarks to compute the pose to within a sign, but again more landmarks can reduce the effects of noise. These independent computations will each yield the pose of the projector with respect to the intermediary device, and then the pose from one projector to the other can be computed as the composite transformation from one projector to the intermediary device then to the other projector.

### 3.3.3 Operation of the System

We are now ready to examine the operation of the tracking system. There are two basic components to this, obtaining an initial estimate, and then updating this estimate in real time. As noted in Section 2.2.3.3, the UNC opto-electronic ceiling tracker has had success with using an EKF to perform this type of update to a set of parameters. I have adopted the EKF as the method of estimating the set of parameters that describe the pose of the camera.

In KF literature, the set of parameters to be estimated is known as the state. With any estimate, there is some amount of uncertainty about the estimate. The EKF maintains an estimate of the uncertainty of each element of the state. The EKF requires the following for operation:

- an $n \times 1$ *state vector* $s$ of the parameters to be estimated
- an $n \times n$ *state transition* matrix $A$ that describes the dynamic behavior of the state $s$ (linear for a KF, Jacobian of a non-linear function in an EKF)
- a measure of uncertainty in the state, i.e. an $n \times n$ process covariance matrix $P$
- a model of the noise in the dynamic process model, an $n \times n$ process noise matrix $Q$
- an initial estimate $s_0$ of the state vector and $P_0$ of the process covariance matrix
- a function $h$ (linear for KF, can be non-linear for EKF) that maps the state vector to the $m \times 1$ output vector of the measuring device
- a map from the measurement space to state space; in the EKF, the Jacobian $H$ of $h$, i.e. the $m \times n$ matrix of partial derivatives of $h$ with respect to the elements of $s$
- an $m \times m$ measurement noise covariance matrix $R$

The assumption is that the noise in both the measurements and the dynamic process model is Gaussian and zero-mean, and thus our model consists of the variances of these respective Gaussians.

#### 3.3.3.1 State Vector and System Behavior

We maintain the position, orientation, translational velocity, and angular velocity of the camera as the state vector. We will denote the state as

$$s = \begin{bmatrix} x & y & z & \theta & \phi & \rho & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho \end{bmatrix}^t$$

Following previous works [Azarbayejani95, Welch96], we maintain the complete orientation externally and maintain the incremental orientation as small rotations $\theta$, $\phi$, and $\rho$. This allows us to linearize the incremental orientation for the EKF without introducing significant error. We want to ignore the equivalence of $360°$ to $0°$ and create a linear orientation description. The model of dynamic behavior is a simple, first-order model of rigid-body kinematics. That is, the camera is predicted to rotate and translate with constant velocity in all six dimensions of motion. We could also augment the state vector with acceleration in each of these six dimensions and assume, for example, constant acceleration.

Note that this assumption of "constant" velocity does not imply that we always have the same numerical value for the estimated velocity, only that our prediction is that this velocity does not change during the interval between measurements. Any change will be estimated, however.

At each time step, the EKF predicts the upcoming measurement. In order to do this, it must predict the value of the parameters in the state. With our model for the dynamic behavior of these parameters, this becomes straightforward. We need the following matrix that describes the rigid-body kinematics equations, given the state vector above.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus our prediction for the new state after time $\Delta t$ has passed since our last update is $s^- = As$, where the superscript $^-$ denotes a predicted value. With this recursive formulation, we need an initial estimate $s_0$ of the state vector to initialize the computation.

#### 3.3.3.2 Noise in the Model of the Dynamic Process

We use the $n \times n$ process noise covariance matrix $Q$ in the EKF to represent the uncertainty in the behavior model. Using the equations used for the uncertainty of user motion in the UNC opto-electronic ceiling tracker [Welch96], we assume that the process noise is a time stationary random process, and thus a function only of the inter-sample time. We further simplify the matrix by assuming that the translation offsets have the same variance and that the rotation offsets have the same variance. This reduces the number of free parameters in the matrix to two variances and the sample time.

$$
\begin{aligned}
Q_{xx} = Q_{yy} = Q_{zz} &= \frac{1}{3} q_p (\Delta t)^3 \\
Q_{\theta\theta} = Q_{\phi\phi} = Q_{\rho\rho} &= \frac{1}{3} q_o (\Delta t)^3 \\
Q_{v_x v_x} = Q_{v_y v_y} = Q_{v_z v_z} &= q_p \Delta t \\
Q_{v_\theta v_\theta} = Q_{v_\phi v_\phi} = Q_{v_\rho v_\rho} &= q_o \Delta t \\
Q_{x v_x} = Q_{y v_y} = Q_{z v_z} &= \frac{1}{2} q_p (\Delta t)^2 \\
Q_{\theta v_\theta} = Q_{\phi v_\phi} = Q_{\rho v_\rho} &= \frac{1}{2} q_o (\Delta t)^2
\end{aligned}
$$

All other elements of $Q$ are 0.

We also need to express the uncertainty each time we update the state according to our model of dynamic behavior. This projection of the error covariance forward in time is done with the following formula.

$$P^- = APA^t + Q$$

The first term merely projects the error covariance in time, while the second term introduces new uncertainty that is due to noise in the dynamic model. Again, this is a recursive formulation of $P$; thus we need an initial estimate $P_0$ of the covariance matrix in order to begin the algorithm.

#### 3.3.3.3 Initial Estimates and Re-initialization

We can compute an initial approximation by placing the camera at an arbitrary but fixed pose and determining the offset to each projector independently. This can be done with a number of algorithms in computer vision, including the Eight-Point Algorithm [Longuet-Higgins81] discussed in Section 2.3.1. These offsets can of course only be determined to within a scale factor using a single projector. However, since we know the relative poses of the projectors, we can determine the absolute scale (of each offset) by simple vector geometry. This gives a method to completely determine an initial approximation. The uncertainty of an initial estimate obtained with such an algorithm is a property of that algorithm, and analyses of these algorithms are available [Maybank92a, Faugeras93, Kumar94, Maybank98]. We have not implemented any of these algorithms for use in the simulations described in Chapter 4.

#### 3.3.3.4 Updating the State and Error Covariance from Measurements

The EKF requires a *measurement function* $h(s)$ that predicts the upcoming measurement given the current state. We define a function that determines the distance of the detected landmark from the epipolar line, shown in Figure 3.7. It uses the projector's pose in the world, the camera's estimated pose in the world, and the image-plane coordinates of the projected landmark to determine the epipolar line. Simple 2D analytic geometry gives the distance from the detected landmark to the line. The predicted measurement is never really computed in this function; $h(s)$ is identically 0. That is, we expect the detected landmark to lie on the epipolar line. The distance then constitutes the error (*residual*, in KF literature). The following set of equations computes the distance from the detected landmark to the epipolar line. Included in this list are the formulas we need for the epipolar geometry, specifically, for computing the epipole $e$ and the fundamental matrix $F$. The $j^{\text{th}}$ measurement $z_j$ taken using the $i^{\text{th}}$ projector is computed with the following set of equations [Luong95, Faugeras93].

$$z_j = h(s_j) = m^t F m'$$

where

$$\begin{aligned}
m &= \begin{bmatrix} u_j & v_j & 1 \end{bmatrix}^t \\
F &= E\, U_C^{-1}\, R_{WC}^t R_{WP,i}\, U_{P,i} \\
m' &= \begin{bmatrix} \mu_j & \nu_j & 1 \end{bmatrix}^t \\
e &= U_c^{-1}(T_{WC} + R_{WC} T_{WP,i}) = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \\
E &= [e]_\times = \text{ matrix } E \text{ such that } Ex = e \wedge x = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \\
U_C &= \begin{bmatrix} 1 & 0 & -\frac{w}{2} \\ 0 & 1 & -\frac{h}{2} \\ 0 & 0 & -f \end{bmatrix}
\end{aligned}$$

Note that the symbol $\wedge$ denotes the vector or cross product of two vectors. The only units in the above equations are pixels, associated with $m$, $m'$, and $e$. However, the units are obscured by the fact that $E$ and therefore $F$ are defined only to within a scale factor. Thus the distance computed by the above equations must be normalized such that the units in which the distance is expressed are pixels. The normalizing factor for the epipolar line

$$Fm' = \begin{bmatrix} A \\ B \\ C \end{bmatrix}$$

is $\sqrt{A^2 + B^2}$.

The correction to the state is based on the residual and a factor $K$, known as the *Kalman gain*. The gain factor is computed as a weighted average of the old and new data, using the relative uncertainties for weights, and is computed with the following equation.

$$K = P^- H^t (H P^- H^t + R)^{-1}$$

$R$ reflects the noise of the measurements. In this case, it is simply the variance of the landmark localization on the camera image plane. $HP^-H^t$ can be thought of as mapping the uncertainty in the current estimate into the measurement space. Thus the denominator gives a total confidence measure, and the numerator shears the potential correction according to the uncertainty in the current estimate and in the measurement.

$H$ maps the error between the predicted distance $h(s) = 0$ and the actual distance $z_j$ to the elements of the state vector. For the EKF, it is the Jacobian of $h(s)$, or the matrix of partial derivatives of $h(s)$ with respect to each of the elements of $s$. This means that the correction term is a linear approximation to the change in the state.

It is a daunting task to write an analytical expression for the Jacobian of $h(s)$. We thus compute the value of the partial derivatives only at the current state vector. These derivatives are computed numerically with PDV C++ classes written by Prof. Gary Bishop. The code re-implements (overloads) the primitive operations of addition, subtraction, multiplication, division, sine, cosine, and square root. For each operation, the code computes the numerical value of the analytical partial derivative that results from the given input values. Each partial derivative is initialized to 1.0, since the derivative of any variable with respect to itself is 1.0. The analytical formula for each primitive operation is easily implemented. Each variable for which partial derivatives are to be computed must be assigned a slot in the array of all partial derivatives. For example, to compute the partial derivative of $f = a \cdot b$ with respect to both $a$ and $b$, one would create the PDV instance for $a$ and assign the partial derivative in the first array slot to be 1.0, and all others to be 0.0. Then one would create the PDV instance for $b$ and assign the partial derivative in the second array slot to be 1.0, and all others to be 0.0. When the multiplication operation occurs, the function value is computed. Then for each slot in the array of partial derivatives, the product rule is used to compute the partial derivative.

$$\frac{\delta f}{\delta a} = b, \frac{\delta f}{\delta b} = a$$

Thus the array of partial derivatives would be $[\ b \quad a\ ]$. But the code cannot assume that $a$ and $b$ were scalar variables; they could have been functions of other variables. Thus the chain rule dictates that we must multiply the partial derivative of $f$ with respect to each of these functions by the derivative(s) of $a$ and $b$ with respect to their component functions and variables. Effectively, this requires that the PDV class copy both arrays. The initialization has taken care of this bookkeeping, however. Similarly, we can implement the constant rule, sum rule, quotient rule, and rules for trigonometric functions and the square root function.

We can now add a correction term to the state. The distance along the direction in state space that we are going to move (given by $K$) is determined by the residual, which since our predicted measurement is zero distance between the detected landmark and the epipolar line, is equal to the measurement.

$$\hat{s} = s^{-} + Kz_j$$

where $\hat{}$ denotes an updated value. This provides an update to the translation components and velocity components in the estimate, and computes an incremental rotation that we must apply to our current estimated orientation. We update this portion of the estimate by computing the rotation matrix determined by the incremental rotation parameters in the state.

$$\hat{R}_{WC} = R_{WC}\ \text{Rot}(\theta, \phi, \rho)$$

where Rot$(\theta, \phi, \rho)$ is the $3 \times 3$ rotation matrix determined by the Euler angles [Foley90]. Note that the computation of the Jacobian must account for the current total orientation, and thus must reconstruct $R_{WC}$ with the correct inverse for the chosen definition of Euler angles. After this update, the incremental Euler angles $\theta$, $\phi$, and $\rho$ are reset to 0.0.

Since we've updated our estimate, we must again update the error covariance associated with the estimate. This update is given by the following equation.

$$\hat{P} = (I - KH)P^{-}$$

where $I$ is the identity matrix. The updated values are used to continue the iteration at the next time step.

#### 3.3.3.5 Filter Tuning

It is difficult to analytically determine the correct variances that constitute the matrix $Q$ for a motion path of a human user. Thus we tune the EKF by adjusting the parameters that determine the non-zero values of $Q$. To do this, we need a metric measurement for the error. We choose a set of 16 points distributed over the environment. We run the algorithm with a pre-programmed motion path for the true camera. For each estimated camera psoe over this run, we compute the total squared error between the projection of this set of points into the true camera image plane and the projection of this set of points into the estimated camera image plane. At the end of the run, we compute the root-mean-square $(\mathrm{RMS})$ error for the entire run. This value gives a metric measure of the filter performance. We minimize this error measure with respect to the two variance parameters, $q_p$ and $q_o$. If our pre-programmed motion path is realistic (i.e. closely reflects the path that the physician will take during operation), then the parameters found with this method will yield the best performance of the EKF for the motion. The experiments to perform this tuning are described in Section 4.1.3. We could implement this procedure with a metric other than RMS error. We chose RMS error so that consistent behavior from one frame to the next with a slightly higher average would be preferred to a sequence of estimates in which occasional frames exhibited high error.

## 3.4 Visualizing the Algorithm using Epipolar Geometry

We can think of the algorithm as constraining the camera to lie on the intersection of epipolar planes. In this section, I will expand on the description of our approach as a solution using epipolar geometry to create a visualization of the algorithm. This visualization also underlies the construction in Figure 2.8. Any plane is determined by a line and a point not on that line. An epipolar plane is defined by the line which contains the optical ray through which one camera sees a given point and the center of projection of the other camera. Notice that the line containing the optical ray necessarily contains the center of projection of the first camera. The same definition applies to a projector and a camera, except that the direction of the optical ray is reversed. The line, however, remains unchanged.

We can equivalently think of the epipolar plane being defined by the line that contains the two centers of projection (camera and projector) and the point in which the optical ray emanating from the projector intersects the scene. This more directly represents the behavior of the algorithm, since the centers of projection are always the same point on the camera and projector, respectively, while the third point changes as we select a

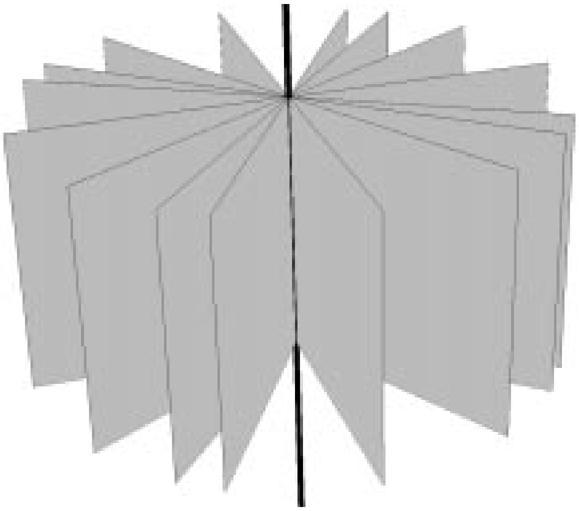


Figure 3.10: A pencil of planes can be visualized by a rectangle rotated around a line contained in that rectangle.

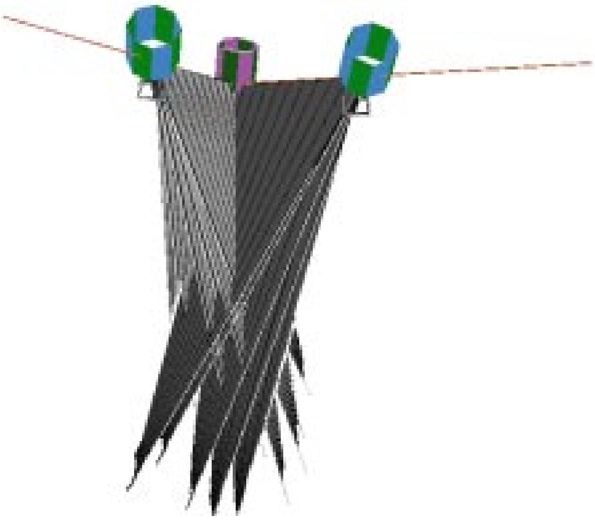


Figure 3.11: With two projectors, we construct two pencils of epipolar planes. We can thus constrain our estimate of the camera pose to a single point, which the intersection of those two pencils of planes. That intersection point must be the true pose of the camera.

new ray to project from the projector. For each ray, then, we form an epipolar plane.

The series of epipolar planes forms a *pencil of planes* (Figure 3.10). A pencil of planes is the set of planes that share a line of intersection and can be visualized by rotating a square around some line that it contains. The pencil is defined by this line of intersection. In the configuration of our tracking system, this line contains the centers of projection of both the camera and the projector.[7] Thus with one projector we can constrain the camera pose to a line in space and fully constrain the camera orientation. With two projectors, we constrain the camera pose to the intersection of two lines–a point, since the projectors are at distinct poses. Thus we can fully constrain the six-DOF camera pose (Figure 3.11).

[7] This line of course changes as the camera moves, but for purposes of explanation, visualization of a single line is sufficient.

```
Create initial estimate of pose, velocity, and covariance.
while( TRUE )
    Choose projector and pixel and project a landmark.
    Acquire camera image and measure elapsed time.
    Predict pose of camera at this time.
    Update covariance matrix for time step.
    Search camera image for the landmark.
    if( landmark successfully detected )
        Compute epipolar line corresponding to projected landmark.
        Measure the distance between the detected landmark and the epipolar line.
        Compute correction factor with extended Kalman filter.
        Update estimated pose and velocity.
        Update covariance matrix to reflect new measurement.
    end if
end while
```

Figure 3.12: Pseudocode for the new tracking algorithm.

## 3.5 Summary

The algorithm can thus be summarized as follows. We begin with an estimate of the pose and our confidence in that. At each step, we project a landmark and compute the corresponding epipolar line. We correct the estimated pose by a distance that is proportional to the distance of the detected landmark from that epipolar line. The direction for correction is computed through the Kalman filter mechanism using the Jacobian and the confidences of the prediction model for dynamic behavior and the detection algorithm. Figure 3.12 presents the algorithm in pseudocode.

# Chapter 4

# Experimental Performance and Analysis

Now that we have specified the algorithm, we want to assess its performance. There are two approaches we take, the first a more practical approach grounded in theory, and the second a purely theoretical analysis of conditions which, if they occur, will make it difficult for the algorithm to produce accurate estimates of the camera pose. To perform the first of these, we implement a simulator and test the algorithm with user head motion data recorded from our primary intended application, the ultrasound augmented reality system described in Section 2.1.2.4. The simulator and user experiments are described in this chapter. The second approach for analysis examines geometric configurations of the cameras and projector under which the algorithm's performance could suffer or under which the algorithm could fail. The simulations reveal that the algorithm as specified in Chapter 3 works well for the intended application with fast cameras, while performs below our requirements for normal-speed cameras. Methods to overcome this limitation will be discussed in Chapter 5.

## 4.1 Performance in Simulation

### 4.1.1 Implementation of Simulator

The simulator consists of two pieces of code: code to run the algorithm as specified in Chapter 3, and code to simulate the actions of the projectors and cameras in the system. The former is a direct implementation of the pseudocode in Figure 3.12 and the equations presented in Section 3.3. The latter requires some simple geometric processing which we now describe.

Refer again to the list of symbols in Figure 3.9. The list consisted of poses for the projectors in the world (rotations $R_{WP,i}$ and translations $T_{WP,i}$), calibration matrices for the camera ($U_C$) and projectors ($U_{P,i}$), image-plane coordinates of the projected landmarks ($\mu_j, \nu_j$) and detected landmarks ($u_j, v_j$), and the estimated pose of the camera (rotation $R_{WC}$ and translation $T_{WC}$). In order to compute the data necessary for the algorithm and to measure the performance of the algorithm, the simulator must keep track of the true pose of the camera. Fundamentally, we define system performance by how well the estimated camera pose matches the true camera pose. Let us now discuss how we select or compute each of these values in the simulator.

One of our goals for the simulator is to get results that will directly apply to an implementation for our primary intended application, that of ultrasound-guided needle biopsy. This implies that we want to select the poses ($R_{WP,i}$, $T_{WP,i}$) of the projectors in the simulator to be projecting onto the surgical environment depicted in Figure 3.1. Recall that this configuration had the projectors mounted on the ceiling, pointing down at an operating table. This goal also implies that we want to select values for the intrinsic parameters of the camera ($U_C$) that match the cameras that we currently use in our AR system (and that are similar to what we envision using for the foreseeable future) and values for the intrinsic parameters of the projectors ($U_{P,i}$) to match the DMD-based projectors mentioned in Section 1.3.1 that we envision using in the first implementation[1]. The dimensions of both image planes are $640 \times 480$. The horizontal field of view for the camera is $28.64°$ and for the projector is $40.1°$.

The selections of which projector to use for the next projected landmark and the image-plane coordinates ($\mu_j$, $\nu_j$) for that projected landmark are part of the algorithm, as shown in Figure 3.12. For the simulator, we use a simple pseudo-random distribution over the portion of the image plane that projects onto the approximate volume over which we know we will want to view augmented imagery. More sophisticated strategies for these selection algorithms are discussed in Section 6.2.

To simulate the detection of landmarks at location ($u_j$, $v_j$) in the camera image plane, we must project the pixel from the projector into the environment, then project the intersection point into the camera image plane. The former operation is a simple ray casting operation, similar to that implemented in ray tracing software. This assumes a model of the geometry of the environment. This is exactly the information we do not want to know in the real system, but the simulator must know it in order to accurately depict the interaction of light with the scene. The second step requires a transformation of the coordinates of the intersection point from the world coordinate system to the camera coordinate system. Note that this operation uses the true camera pose, not the estimated pose ($R_{WC}$, $T_{WC}$). This point is projected into the camera image plane. The simulator computes the estimated pose of the camera with the algorithm in Section 3.3. This completes the list of quantities in Figure 3.9.

There are still other quantities that we need to begin a simulation, however. We next turn to the initial values for the EKF. Recall that the filter requires an initial estimate for the state—the camera pose in the algorithm—and for the covariance matrix associated with those estimates. Briefly, the covariance matrix contains along its diagonal the variance of the estimates of the elements of the state vector and off its diagonal, the

[1] All these parameters could of course be chosen arbitrarily, but we opted for as much realism as we could achieve for these parameters.

covariance of the estimates. These numbers reflect the presumed distribution of the value, or more informally, our confidence in the current estimate.

The true initial pose of the camera with respect to the projector can be used to create the initial estimate with perfect accuracy, or an arbitrary set of parameters can be used to create an initial estimate with some error. In the latter case, the filter converges after a small number of measurements. How many measurements are required to converge depends on the initial error and whether (and how much) the camera moves during this interval. We choose the initial covariance matrix to give a small amount of uncertainty, for example one millimeter for position and one milliradian for orientation in each Euler angle, to each parameter with no correlation. In this case, the initial covariance matrix is $10^{-6}I$, where $I$ is the identity matrix. Recall that the elements of the diagonal of $I$ are variances, and thus are the square of the standard deviation of the Gaussian that represents our initial estimate of the distribution of the true parameter value. This distribution frequently does not enclose the true pose, but since the filter injects noise in the error covariance matrix $P$ through $Q$ at each time step, the filter will not ignore measurements. This enables the filter to overcome poor initial estimates of both the state vector $s$ and $P$.

In any real implementation of such a system as we propose, there will be noise in the detection of the landmark. We simulate this by adding Gaussian white noise to the $(u_j, v_j)$ coordinates computed for the detected landmark. The variance of this Gaussian distribution is the quantity that we call $R$ in the EKF algorithm.

Another issue in the simulator is the control of the true camera pose. We gathered real user motion data from the ultrasound visualization system, using the UNC optical ceiling tracker described in Section 2.2.2.6. All the application can do is sample the user's head pose at discrete intervals. There is some noise present in this data that we eliminate by resampling with a low-pass noncausal filter [KSC-TR]. In order to provide a continuous motion path, we build a series of Catmull-Rom curves that interpolate these samples smoothly. This is the path along which the camera moves during experiments.

### 4.1.2 Performance Metrics

As noted in Section 2.2.1, we generally want as much accuracy as possible from a tracking system. For most tracking systems, the only way to measure the accuracy is in millimeters of position error and degrees of orientation error. Certainly, those metrics apply to this system as well.

But since this algorithm is designed to be applied to a VST AR system, there is another error metric that we want to consider. That is pixels of registration error on the image plane of the camera. After all, registration error is the manifestation of tracking error that affects the user of a VST AR system, not the position error and orientation error directly (although they of course determine the registration error). This is a more tolerating error metric, since position error and orientation error can frequently cancel out registration error in the critical portion of the field of view, a phenomenon we have witnessed with our current vision-based tracking system [State96a]. Only in the worst case will the errors be additive.

Also, we want to examine the rate of convergence and its implications for the latency of the new tracking algorithm. Latency is one the largest sources of registration error for an AR system. A detailed analysis of the sources of registration error [Holloway95] revealed the following "rule of thumb" relationships that apply to our VST configuration.

1. One millimeter of position error results in one millimeter of registration error at arm's length (500 mm).

2. One tenth of a degree of orientation error results in one millimeter of registration error at arm's length.

3. One millisecond of latency results in one millimeter of registration error at arm's length.

Using these results, we will be able (in Section 4.1.6) to bound the registration error that will result for given amounts of position error, orientation error, and latency. We will then compare this value to the registration error measured in the experiments described in Section 4.1.5).

### 4.1.3 Tuning Experiment

In Section 3.3.3.2 we discussed the noise in the model of the dynamic process. The process noise covariance matrix $Q$ was specified in terms of two variances, one associated with the translation components of the camera pose and one associated with the rotation components. Since our dynamic behavior model represented by the state transition matrix $A$ does not account for acceleration, we are effectively saying that we can treat the acceleration as a source of noise in the dynamic process, and then we re-estimate the current position and velocity taking this into account. The accounting is done by adding uncertainty into the process covariance matrix $P$ each time we perform an update of the state for the passing of time.

$$P^- = APA^t + Q$$

We need a method to determine the best parameters for these two variances. To do this we select a "typical" motion path from the set of user head motion data **from the application of interest**. We then run an optimization procedure to minimize the error with respect to those parameters. We can think of this process as finding the best balance between "believing" the measurements of the point-to-line distance and "believing" in the dynamic process model.

This user study is described below. Prior to that description, let us note two important points. First, the fact that the data comes from the application to which we are targeting the system is important; it implies that the optimal values we find for that path will apply when we implement the system. The primary physician user of the ultrasound-guided needle biopsy system was not among the users in the user study. It may be that the motion path she takes while performing the procedure differs in a significant manner from the motion recorded and the parameters would change significantly, but this is unlikely and not crucial to the theory behind the algorithm.

The second important note is that the best values for these parameters depend on the acquisition speed of the camera. This is because the interval over which the simple dynamic model (rigid-body kinematics with only position and velocity) is equal to the time between acquisition of images from the camera. An analysis of head-motion prediction [Azuma95] shows how the error in this prediction grows with the length of the time interval. Thus with an increase in the interval over which we must predict the user motion, we know that we will want to decrease our confidence in the accuracy of the dynamic model and increase our belief that the measurements are giving us accurate information.

The tuning experiment consisted of two phases. In the first phase, we recorded several users' head motion while performing an ultrasound-guided needle biopsy task. The biopsy procedure requires the physician to insert a needle into a small lesion. We simulated this procedure by asking the user to insert a tracked needle into a medical phantom[2] with simulated lesions (Figure 4.1). The view presented to the user was entirely virtual (Figure 4.2); however, the calibrated scene geometry from the AR ultrasound visualization system (described in Section 2.1.2.4) was used so that the imagery was accurately displayed to the user. The user saw the patient torso, a sphere placed at the location of the lesion in the medical phantom, and the tracked needle. In addition, the physical model used for the AR system was registered to the visual feedback, so that the tactile feedback the users in this experiment received was the same as that which the physician receives while performing the procedure. The use of only a synthetic view (as opposed to an AR view) also allowed the system to run faster, which we hoped would allow the user to gain the full effect of head motion on a more timely basis than if the system had to wait for the video digitization into the frame buffer. We hope the increased speed produced more realistic motion paths for a fast, unobtrusive tracking system such as we propose to build with the new algorithm.

The second phase of the tuning experiment consisted of running an optimization procedure to find the best

[2] These and similar phantoms are used by medical schools for training physicians. The phantom consists of a gel-like substance that appears similar to human tissue when imaged with ultrasound. Embedded in this gel are liquid-filled and solid-filled sacs that emulate the appearance of cysts and tumors, respectively, in ultrasound. We mount this phantom in a store mannequin to simulate the geometry of the human torso.

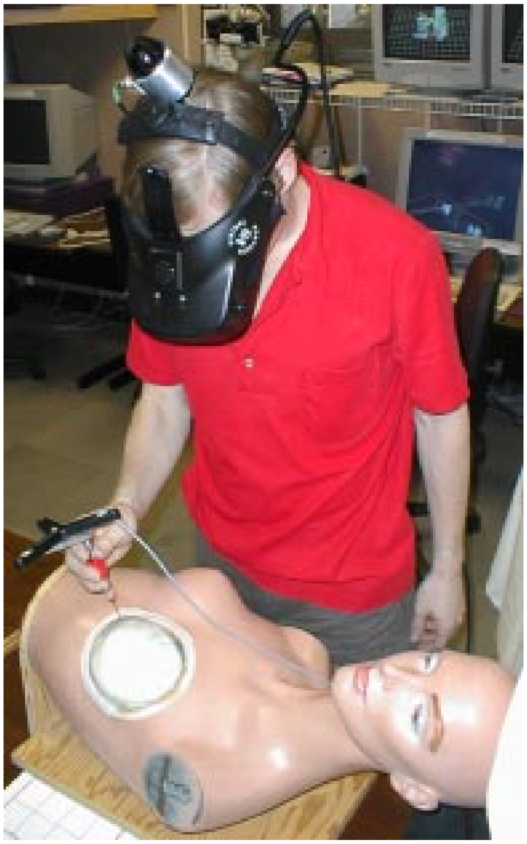
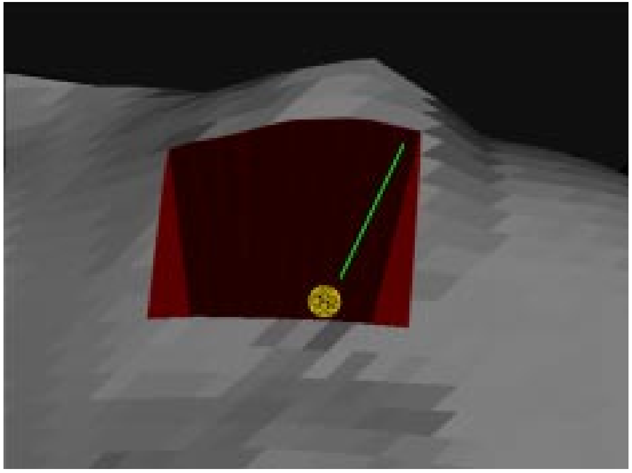


Figure 4.1: (*left*) A re-creation of the scene of the user experiment. The user wears an HMD that lacks see-through capability. The user inserts a tracked needle into a target within the medical phantom in the mannequin torso. The location of the target and the needle were both depicted in the virtual world presented to the user (Figure 4.2).

Figure 4.2: (*below*) The view presented to the user during the user study of head motion while simulating a needle biopsy. The line was aligned with the tracked needle and the sphere with a lesion in the medical phantom. This gives correct tactile feedback, helping overcome the lack of visual feedback from the real world.



values for the parameters. We need a function to minimize. Since our primary application for this tracking system is an AR system, we want to minimize the registration error metric described in Section 4.1.2 over the duration of the user's motion. We selected one motion path to represent the set and computed the RMS registration error for the complete motion path. This error metric was returned to the optimizer as the function value. Using the multi-dimensional minimization strategy known as Powell's method [Press88], we searched the parameter space for the best values for that path. By assuming that path to be representative, we assumed that we found the best parameters for all paths. We tested the algorithm's performance with the remaining paths. Results from those tests are presented in Section 4.1.5 and analyzed in Section 4.1.6.

The optimizer needs to have the range in which to search. We choose the range by selecting a reasonable value for $R$, the uncertainty of the measurements, and then examining the relative magnitudes of $R$ and $HP^-H^t$. These two terms determine the total weight of the process model and the measurements when the Kalman gain $K$ is computed.

$$K = P^-H^t\left(HP^-H^t + R\right)^{-1}$$

Knowing the possible magnitude for $H$ gives us a range in which $P^-$ should fall so that the order of magnitude of $HP^-H^t$ does not vary greatly from the order of magnitude of $R$. If it does, then one of the terms is rendered irrelevant in the computations, which implies that either the EKF is ignoring the measurements (when $HP^-H^t$ is large) or ignoring the dynamic behavior model (when $HP^-H^t$ is small). Neither of these situations is desirable.

We selected $R = 1.0$, which says that we believe we can find the centroid of the detected landmark to within 1.0 pixels on the camera image plane. This is pessimistic for many sophisticated image processing algorithms, but preliminary experiments with simple implementations of this system did not achieve the accuracy of those algorithms; further experimentation will be necessary when the proposed system is implemented to determine the proper value. The elements of the Jacobian are in pixels-per-meter or pixels-per-radian since we express the translation in meters and the rotation in radians. We experimentally determined the range of the order of magnitude of the elements of the Jacobian to be $10^{\pm 3}$. When the EKF converges, we see in the simulator that the variances and covariances in the $P$ matrix are of order $10^{-6}$. This steady-state value of $P$ depends largely on the order of magnitude of elements of $Q$, which we need to determine. To do this, we write the total weight factor in $K$ in terms of the $P$ matrix that begins each predictor-corrector cycle.

$$HP^-H^t + R = H(APA^t + Q)H^t + R$$

Now we write out the order of magnitude of each matrix as powers of ten, and recall the formulas for the elements of $Q$ from Section 3.3.3.2. Since time is measured in seconds and is on the order of milliseconds, the elements of $Q$ can range in order of magnitude from $10^{-3}$ to $10^{-6}$.

$$H(APA^t + Q)H^t + R \approx 10^3(10^0 10^{-6} 10^0 + [q_p, q_o] \times 10^{-6})10^3 + 10^0$$

Despite the abuse of notation, it follows from the above expression that the order of magnitude of both $q_p$ and $q_o$ should be between $10^{-2}$ and $10^2$ to keep the weights within a ratio of 100:1. At greater ratios, either the measurements or the dynamic process model will not affect the estimate. Initial values in the optimizer's search procedure for the variance parameters in the $Q$ matrix were selected to be $q_p = 1.0$ and $q_o = 1.0$. The optimizer was run for measurements at several camera acquisition rates, since as noted above, this speed determines the interval over which we predict with the dynamic model, which in turn determines the error in the prediction for a given user motion. The results are in Table 4.1.

With these values in hand for the parameters of the $Q$ matrix, we now have everything we need to simulate the algorithm.

| Camera speed (Hz) | $q_p$ | $q_o$ |
|---|---|---|
| 200 | 1.304400 | 3.37088 |
| 150 | 1.078860 | 2.04517 |
| 120 | 0.901545 | 1.55981 |
| 90 | 0.905117 | 1.31833 |
| 60 | 0.598406 | 1.24759 |
| 30 | 0.376259 | 1.13971 |

Table 4.1: Table of variance parameters computed for the process noise for several camera acquisition rates.

### 4.1.4 Tests of Convergence and Stability

First we test whether the algorithm converges to the correct answer from an initial error. To test this, we set the initial estimate to a pose with moderate initial error and run the simulator. In this experiment, the true camera pose remained fixed. The results are shown in Figure 4.3. We observe the algorithm reduces the error in position, orientation, and registration to zero. Note that these values do not decrease monotonically. An EKF cannot guarantee that at each step it will reduce the error in the estimate. However, the graphs show that the general trend of the algorithm is to reduce the error. The algorithm oscillates around the correct pose briefly because the estimated velocity built up in the state causes the algorithm to overshoot the correct pose. This could be alleviated (but not eliminated) by adjusting the process noise parameters to place less confidence in the dynamic process model during the time when the algorithm is converging from a poor initial estimate to a fixed camera pose. This could also be alleviated by changing the model of dynamic behavior using semantic knowledge supplied by a system operator or using statistics about the error in prediction [Brown92].

Figure 4.4 shows simulated VST views as the algorithm converges from a poor initial estimate ($1.8$ meters of position error and $28^\circ$ of orientation error) to the correct solution, although again the algorithm overshoots the correct answer and then converges. The real camera then moves through the scene, and the algorithm maintains good registration on the image plane. When the camera comes to a sudden stop, the algorithm again shoots past the correct answer, but only to a registration error of $6.2$ pixels, then quickly converges again.

A more advanced and less obvious consideration applies the general conditions for stability of an EKF to the algorithm. The following conditions imply that the EKF will be globally observable and stable [Deyst68, Maybeck79, Welch96].

- the set of measurements must form a complete set of constraints
- the system must sample at a rate that is twice the highest frequency of the true signal (the camera motion path)

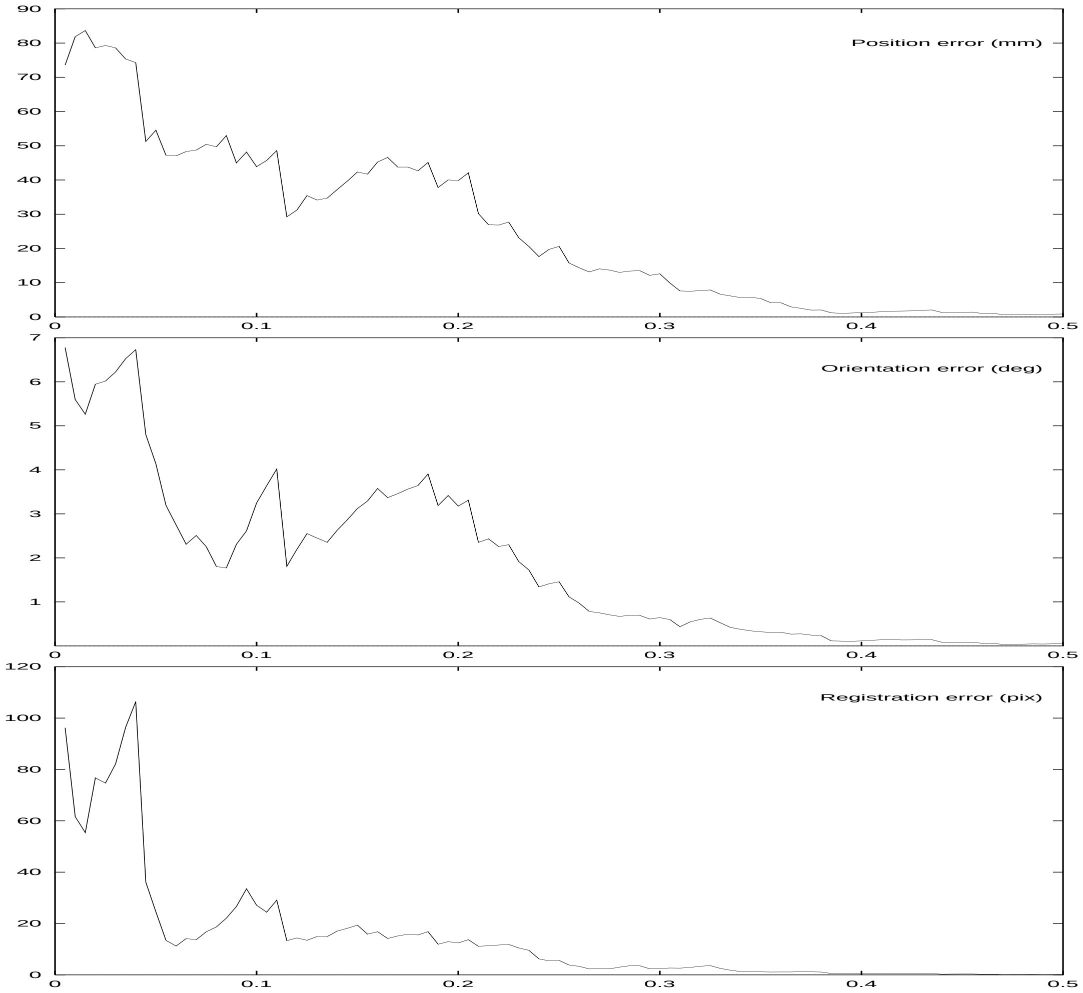


Figure 4.3: The algorithm converges from an initial pose with moderate error. Although the algorithm does not (and cannot guarantee) to reduce the error on each step, the general trend is clearly to reduce the error. Note that after initially approaching the correct answer, the algorithm oscillates briefly before converging to the correct answer. This is because the estimated velocity that the EKF implementation computes is prone to having noise when the true camera pose is fixed, but we did not adjust the process noise parameters.

- the dynamic change over the time between estimates and the corresponding uncertainty is bounded
- the measurement noise is bounded

The first condition is relatively straightforward. It merely says that the problem must be solvable in any form before the EKF can solve it. There are geometric configurations in which the constraints applied in the new algorithm are not sufficient; they will be addressed in Section 4.2.

The second condition gives a version of the condition on the minimum sample rate, known as the *Nyquist rate* in sampling theory [Foley90, pgs. 627–628]. This implies that we have a minimum sampling rate to be able to accurately estimate a given motion path. Indeed, we shall see in a moment that the algorithm's per-

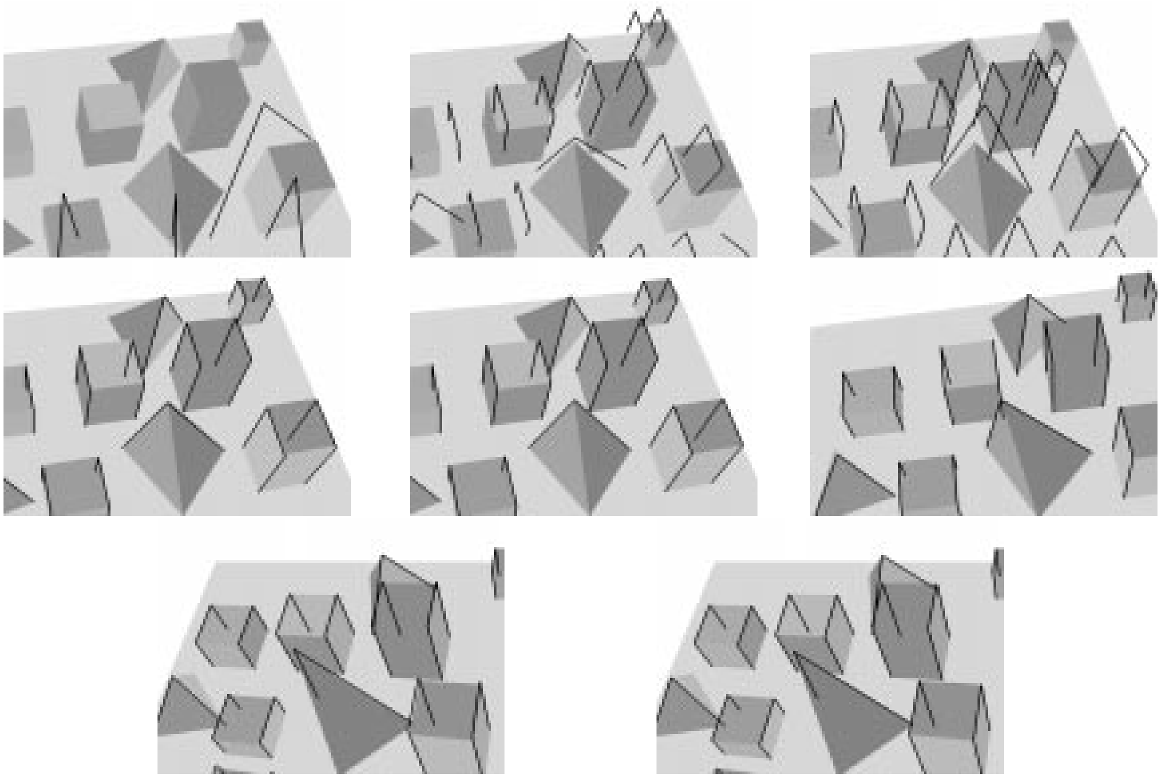


Figure 4.4: The camera view of a simulated VST system while the algorithm runs. The camera image plane is $640 \times 480$ pixels. The black lines simulate the synthetic imagery; they are rendered from the estimated pose. The gray shapes simulate the real imagery; they are rendered from the true pose. The initial estimate (upper row, left) is poor, but the algorithm quickly begins to converge (upper row, middle), and although it overshoots the true answer (upper row, right), it does converge to the correct solution (center row, left). This required 1.5 seconds of simulated time (about 300 measurements). Then the camera begins to move through the scene (center row, middle) and the algorithm maintains good registration (center row, right). When the camera stops suddenly, the algorithm again overshoots the correct pose, but only to $6.2$ pixels of registration error (bottom row, left) and then quickly reconverges (bottom row, right).

formance degrades when the sampling rate drops. However, those same experiments will also justify that we are sampling at a sufficient rate, or at least close to it. Of course, we can never know the maximum frequency for all possible paths. In the simulator we measured the maximum frequency for the paths in the user study to be 10 Hz. We assume the minimum sampling rate we will use is 30 Hz, the standard video rate for NTSC cameras. Thus we are sampling at a sufficient rate to reconstruct the motion path.

The third condition says that the error in the dynamic process must be bounded both above and below. A bound from above limits the amount of error in the predictor phase, which is analogous to requiring an iterative algorithm to be given an initial estimate that is by some definition “close enough” to a solution in order to converge. A bound from below implies that we never believe that the dynamic process is perfect; if the EKF were to be in that situation, then it would ignore future measurements.

The fourth condition says that over some finite set of measurements, there must be sufficient information content (signal, in the sense that we talk about signal-to-noise ratio) in the measurements for the EKF to extract the correct answer.

We found bounds to satisfy the third and fourth conditions for the motion paths in the user study, implying that these conditions are satisfied. These bounds may not be the tightest that could be found, but all that the condition requires is that the bound exist, not that it satisfy some closeness criterion.

### 4.1.5 Numerical Results

With the filter tuned to its best performance on the path selected for tuning and with some confidence that the filter will indeed converge to the correct answer, we can now test the filter's performance on the motion paths of the other users in the study. This consists of simply running the filter using the process noise parameters determined by the optimization process described in Section 4.1.3.

We can use any of the error metrics listed in Section 4.1.2 to measure the performance of the algorithm in simulation, but since the proposed system is aimed at VST AR, we are primarily interested in the registration error on the image plane.

The simulator adds random noise to $u_j$ and $v_j$ at each camera frame, distributed on a Gaussian with a standard deviation that represents the amount of noise we can expect in the location of the landmark on the image plane of the camera. We conducted an experiment to determine how much noise we can expect in these measurements. The experiment consisted of fixing a projector in the environment to project onto a surface. A camera was then fixed in the environment to view this surface. Four trials were conducted to measure the variance of the $u$ and $v$ coordinates of the landmark location. The same camera acquired all the images used for all trials. The images in this experiment consisted of one field of video, which reduces the vertical resolution to 240 pixels. For the first trial, the surface was a white cardboard plane and the shutter time was set to normal speed, approximately 16 ms for one field of video. For the second and third trials, the shutter time of the camera was set to 4 ms and the surface remained the white cardboard plane. For the fourth trial, the shutter time remained 4 ms, and the surface consisted of a hanging beige curtain and a medical training model with approximately human (Caucasian) skin color, although with brightly colored models of internal anatomy exposed. In the first three trials, thirteen landmarks were imaged. In the fourth trial, ten landmarks were imaged. The reduction in the number of landmarks was due to a refinement in the experimental design, selecting ten landmarks which were in view in the first three trials. (In the first three trials, fifteen landmarks were projected, in order to improve the chances of at least ten landmarks being within the field of view.)

The resulting images, when composited as in Figure 4.5, show clusters of landmarks. The variances of the clusters of the landmarks are given in Table 4.2. The important result here is that the variance in the horizontal direction is under one pixel, and the variance in the vertical direction is generally under 0.65 pixels, which may well double at higher resolution, but is still close to within one pixel of noise. The threshold for what intensity is considered "white" must be adjusted for the lower shutter time. For a 16 ms shutter time, an intensity of more than 240 (on a scale of 0–255) was considered white. For a 4 ms shutter time, an intensity of more than 48 was considered white. This is where the fact that we can largely control the lighting in the environment helps in creating structured light patterns. The room was darkened for this experiment, although not all the light was turned off. We did not subtract background (ambient) light in the environment from the image before processing. This may improve the accuracy and robustness of landmark location.

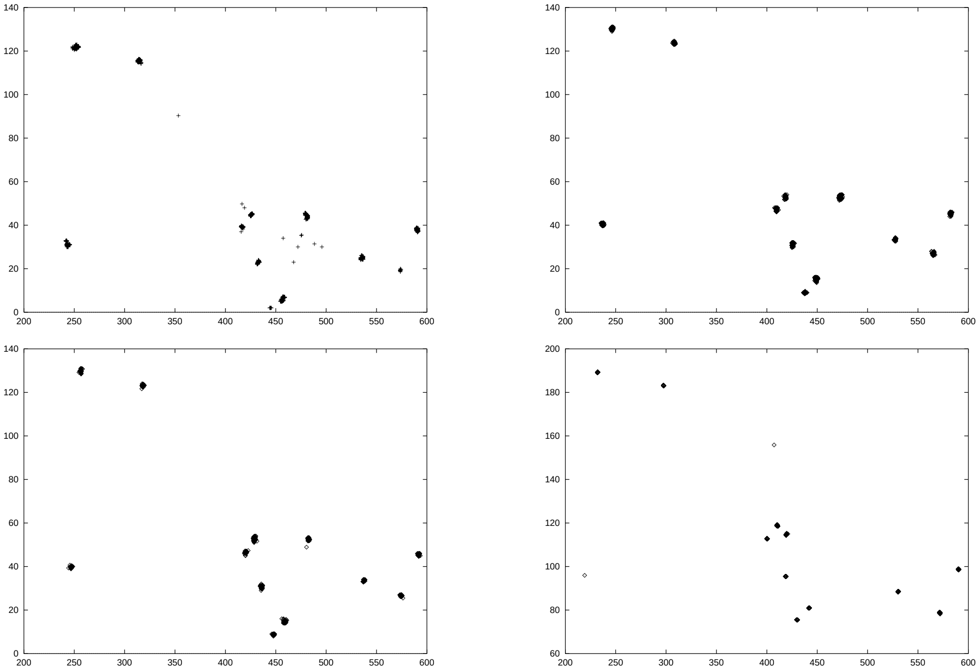


Figure 4.5: Clusters of landmark locations within the camera image plane in the experiment to verify the amount of noise in landmark locations. The axes represent the principal axes of the camera image plane, measured in pixels. On the upper left, the clusters of landmarks projected onto a white planar surface for a camera shuttered at 16 ms. In the upper right, a trial with the camera shutter time set to 4 ms with landmarks projected onto a white planar surface. On the lower left, a second trial with the shutter time equal to 4 ms with landmarks projected onto a white planar surface. In the lower right, a trial with a 4 ms shutter time and landmarks projected onto a beige curtain and medical training model. These models are visible in Figure 4.6.

The first three trials (using the white planar surface) may seem to be performed in an environment that is greatly to the advantage of the system. Thus we set up an environment more similar in reflectance to the environment in the ultrasound-guided needle biopsy application. We draped a diffuse beige curtain on the wall such that its surface was non-planar. Its color is closer to that of light-colored human skin. We placed in front of this curtain a standard anatomical training model with exterior (skin) color matching that of a fair-skinned human and interior anatomy colored with bright colors (Figure 4.6). The shutter time remained 4 ms in this trial. We then ten selected landmarks from the first trials that were within the field of view of the camera and ran the experiment a fourth time. In this experiment, we also measured the percentage of attempts to detect the landmark that were successful. The results show that the landmark detection is still quite robust, with the noise in all trials below 0.32 pixels horizontally and 0.20 pixels vertically. Most landmarks were found on every attempt, and all were found on at least the vast majority of attempts. (Table 4.3).

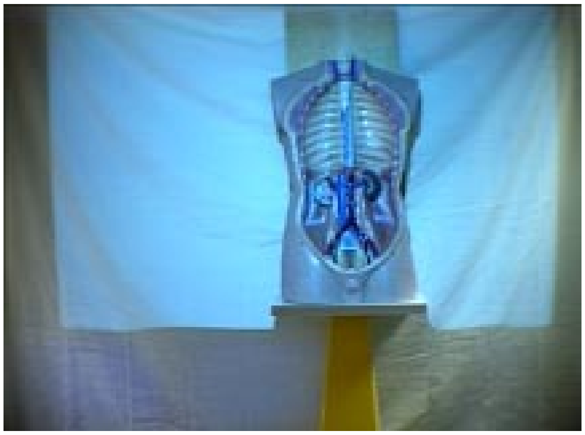


Figure 4.6: Scene of the experiment to measure landmark detection noise in an environment similar to the primary intended application, ultrasound-guided needle biopsy. A curtain hangs in the environment, yielding a non-planar, diffusely reflective surface. The medical training model in the foreground is life-size and has an exterior color of a fair-skinned human. Its interior consists of brightly colored models of interior anatomic structures.

In the simulator, we approximate this noise by setting $R = 1.0$ in the EKF. Adding the noise gives a more realistic simulation of the measurements an implementation would get from a camera. Table 4.4 lists all three types of errors.

The results shown in Table 4.4 reveal two things upon first glance. First, with 200 Hertz cameras, we get excellent performance from the algorithm—under 1.0 pixels RMS registration error for all users.

Table 4.4 also reveals that for each user, the performance degrades by more than an order of magnitude when the speed of the cameras equals the standard rate for NTSC video, 30 Hertz. We know from experience that the registration error listed for the algorithm running with 30 Hertz cameras will be too much for use with the ultrasound visualization AR system described in Section 2.1.2.4. Since this is also the system from which the user motion data comes, we need to look for explanations for the error. This will be done in Section 4.1.6, but we look for some clues to the answer first.

We can illustrate the performance of the algorithm by graphing the error over time as the algorithm performs its real-time estimation. We select User 6 running at 200 Hertz, for reasons that will be made clear momentarily. Figure 4.7 shows the registration error, position error, as well as the translational velocity and translational acceleration. Figure 4.8 show the registration error, orientation error, as well as the angular velocity and angular acceleration. Both graphs show the variables as a function of time, and the individual graphs are aligned vertically so that the ordinate axes are synchronized.

With the synchronization of the time axis in these graphs, we can see that the registration error grows just after the velocity and acceleration in either position or orientation become large. This phenomenon also appears in the same performance graphs for the algorithm running with the motion path for User 1 with 30 Hertz, shown in Figures 4.9 and 4.10.

Now that we have noted this phenomenon, we need to review the method of the EKF and the equations in our implementation to understand why there would be a connection between the registration error and the velocity or acceleration.

### 4.1.6 A Performance Analysis Model

Section 3.2 discussed the intuition behind the new tracking algorithm. Summarizing that discussion, we see that the algorithm works by integrating constraints over time into a single estimate of the camera pose. The advantage this algorithm has over many algorithms presented until recently in the computer vision literature (where the constraint is well-known) is that it uses the EKF, in which there is an explicit model of noise. Since in the simulator this noise model holds, the performance is excellent. We have also given evidence that the noise model will continue to hold in an implementation of the system. Once the filter is integrating numerous constraints, the performance is similar to the performance of batch algorithms in computer vision literature to determine camera pose, when used with similar numbers of constraints with low noise (the noise model in those algorithms).

However, we want to have a deeper understanding of precisely how the new algorithm and system performs. As discussed in Section 3.2.5, the EKF is a predictor-corrector algorithm. One straightforward way to analyze the filter performance and its behavior against various internal system parameters, then, is to analyze the error incurred in the predictor phase and the error overcome by the corrector phase.

This is a general strategy that can in theory be applied to any implementation of an EKF. There are some advantages that make this strategy for analysis appropriate, some particularly for this dissertation.

1. It gives a direct method of measuring exactly those quantities that are important in an AR system: registration error and latency.

2. The analysis is intuitive for the understanding the performance of the EKF.

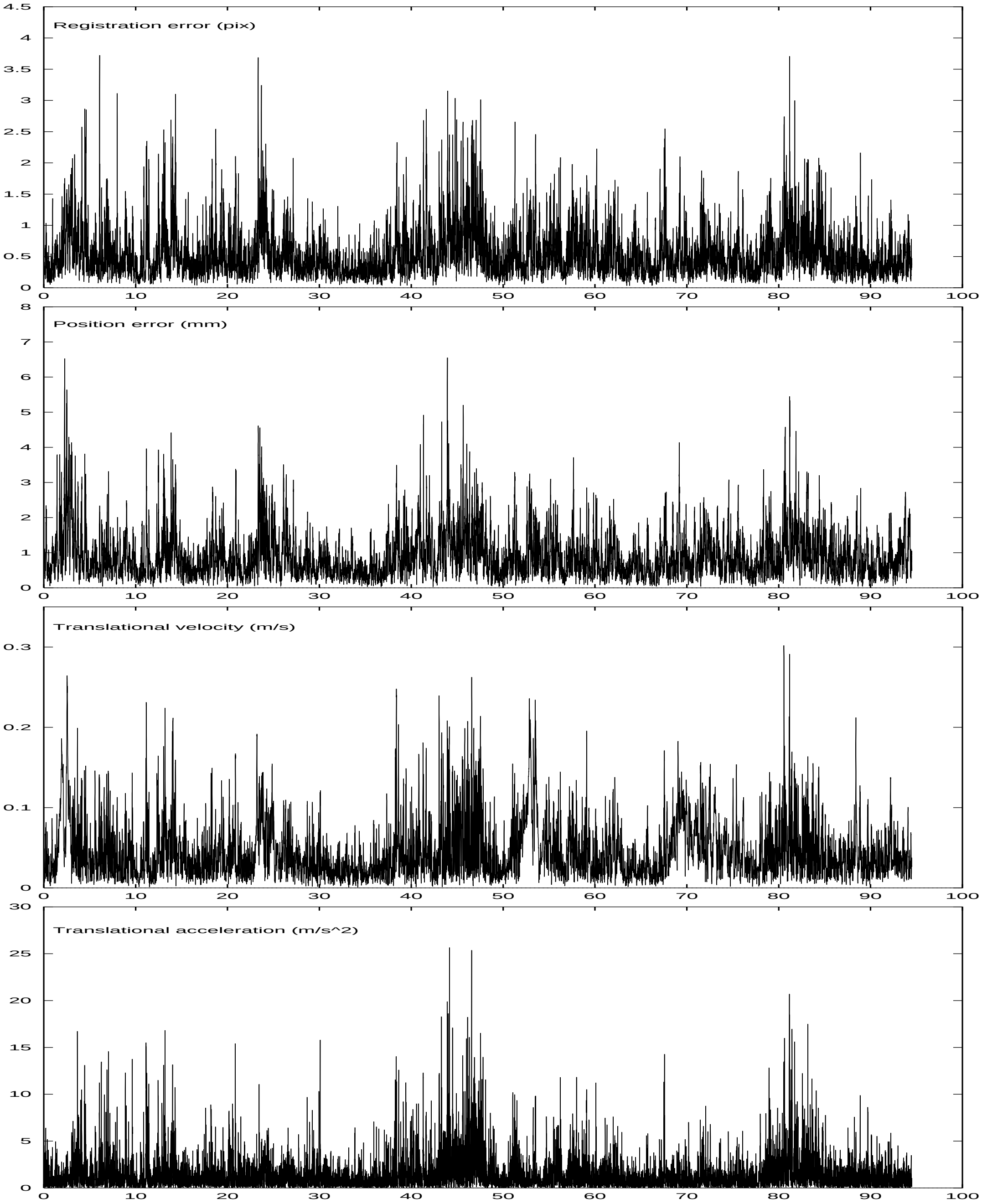


Figure 4.7: Position statistics from User 6 in the user study. All four graphs have the time of the motion path as the abscissa. The top graph shows registration error, measured with the metric described in Section 4.1.2 on a per-frame basis. The second graph shows position error in millimeters. The third graph shows the translational velocity in meters per second, and the fourth graph shows the translational acceleration in meters per second squared.

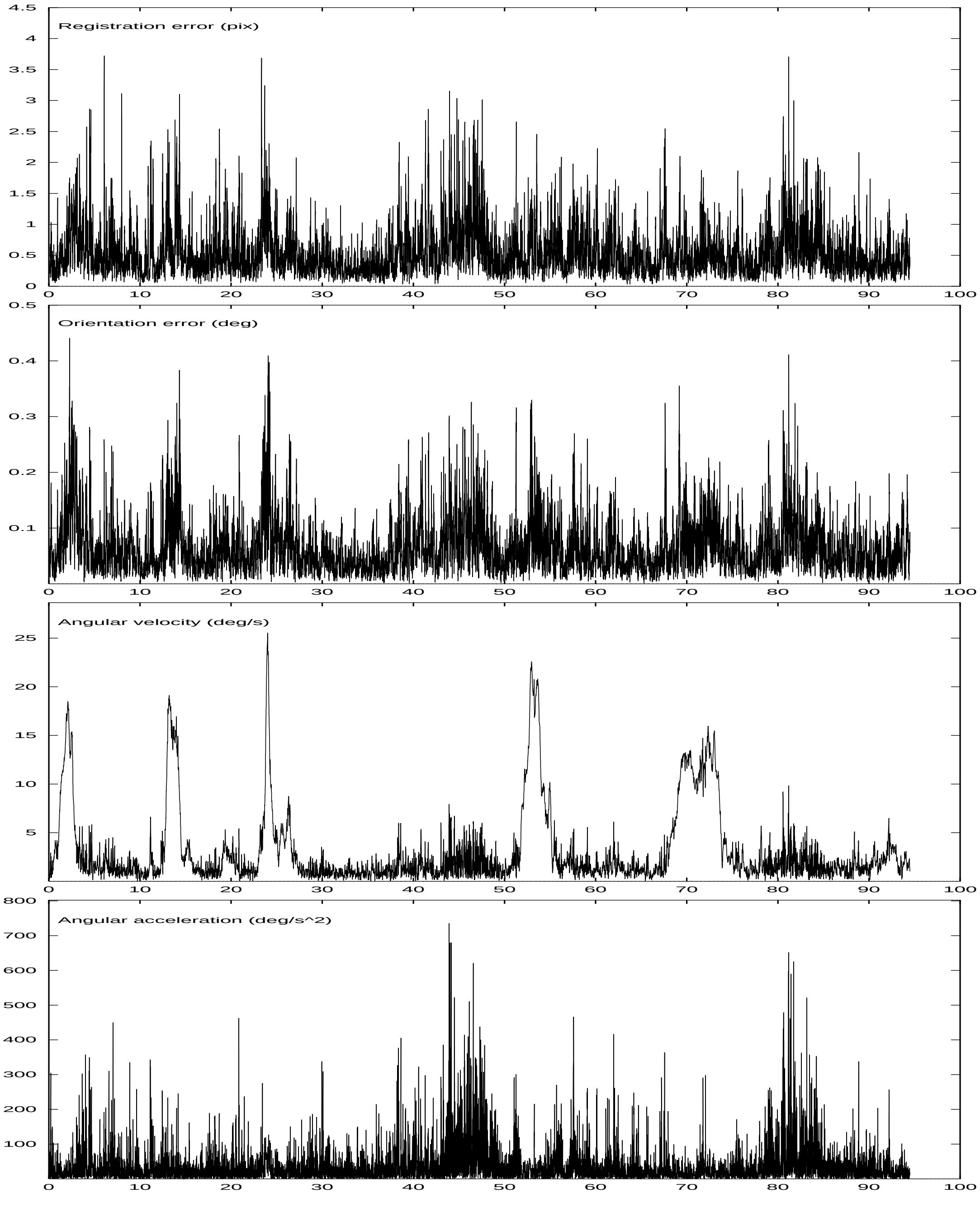


Figure 4.8: Orientation statistics from User 6 in the user study. All four graphs have the time of the motion path as the abscissa. The top graph shows registration error, measured with the metric described in Section 4.1.2 on a per-frame basis. The second graph shows orientation error in degrees. The third graph shows the angular velocity in degrees per second, and the fourth graph shows the angular acceleration in degrees per second squared.

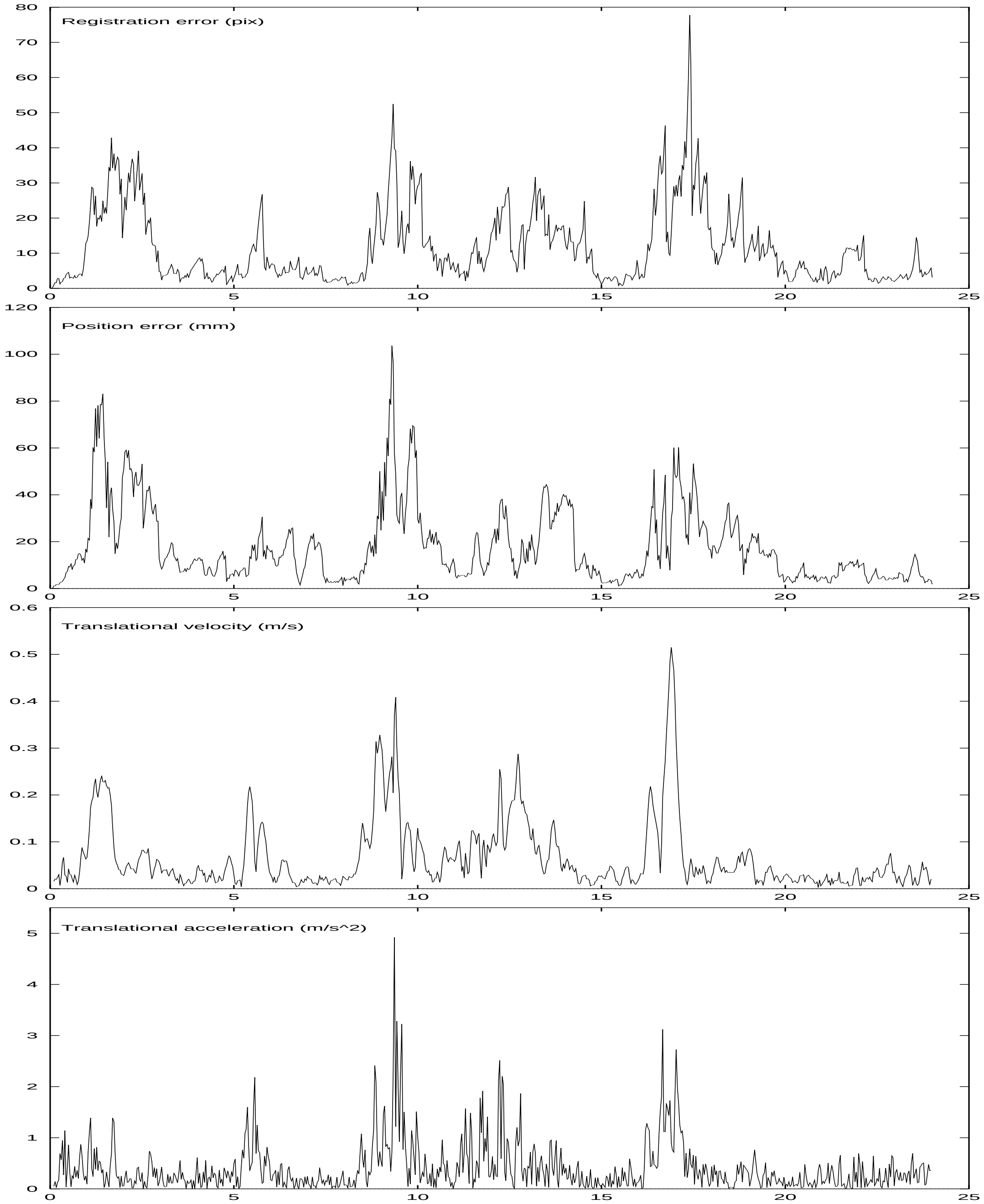


Figure 4.9: Position statistics from User 1 in the user study. These graphs use the same format as those in Figure 4.7.

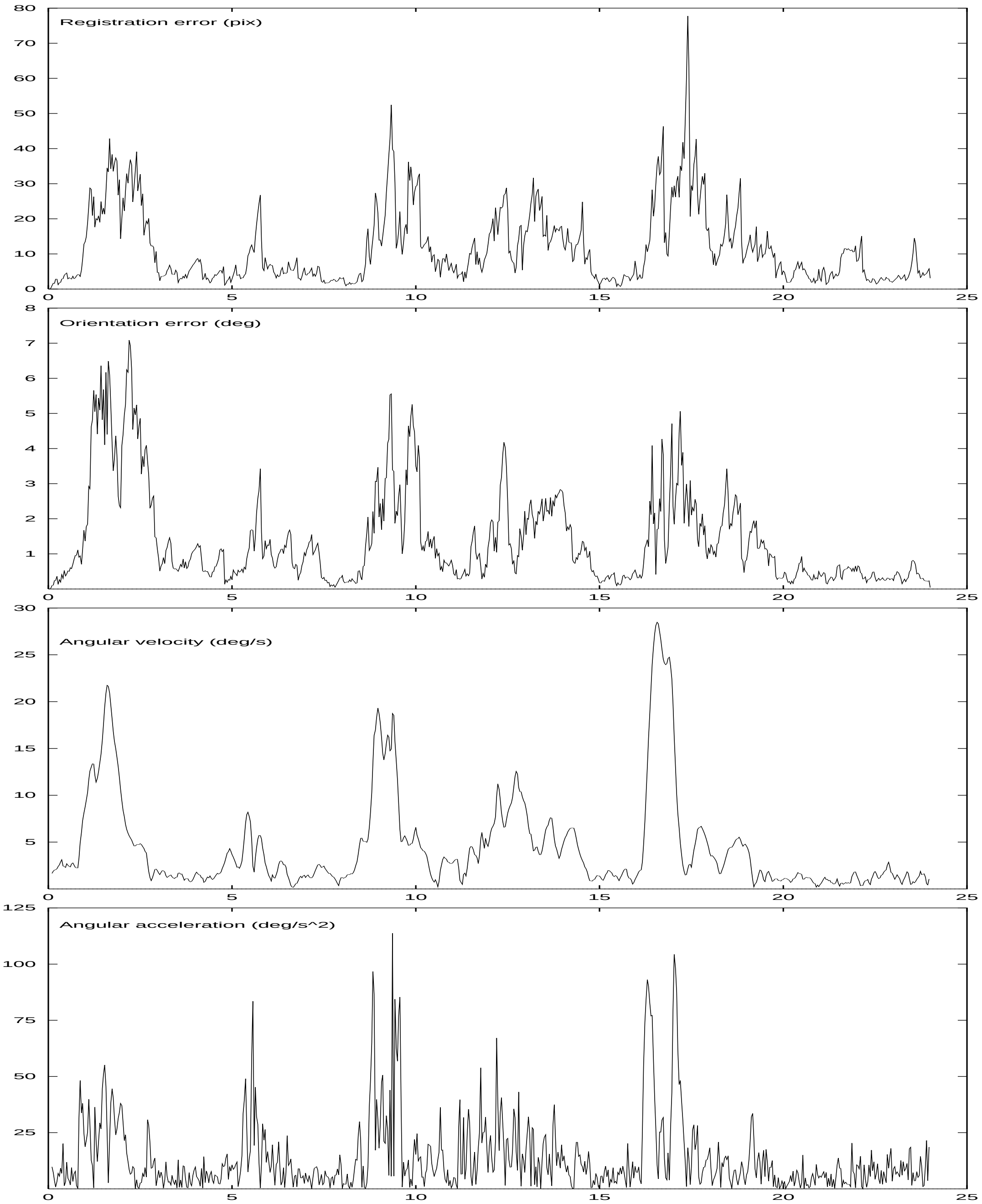


Figure 4.10: Orientation statistics from User 1 in the user study. These graphs use the same format as those in Figure 4.8.

3. It requires knowledge of only the equations of motion of a rigid body in 3D. These can be used to bound the pose error incurred in the predictor phase. We can also alter the prediction scheme without requiring significant additional effort to analyze the extended algorithm.

4. We can analyze the performance of the corrector phase numerically much more readily than we can analytically.

5. We know how head tracking errors and latency contribute to registration error in AR [Holloway95]. These error models allow us to convert the pose error bounds to registration error bounds.

In order to bound the pose error that results from the predictor phase, we need to know several things. We would really like to know the true motion path, but of course, this is unavailable. We can sample it in the simulator. We must, however, understand the error that can result because our prediction model does not accurately reflect the behavior of the user's head (and the camera), even if the values we use in the model are correct. And there will naturally be errors in our estimates of these parameters. In the simulator, we measure the error in the velocity estimates that are generated in the EKF algorithm described in Section 3.3. We are not using acceleration in the predictor phase, so we measure the amount of acceleration. These will give us the information necessary to analyze the error in prediction.

The analysis also indicates how we might improve the algorithm by improving the predictor phase. For example, if we were to add accelerometers to the system, then all we would need to know to extend the analysis would be the accuracy of the accelerometers and the rate at which the acceleration may change. This latter quantity can also be easily measured in the simulator. This extension is presented in Section 5.1.

#### 4.1.6.1 Error Analysis of the Predictor Phase

In Section 3.3.3.1, we specified the state vector and the dynamic system model that we maintain in the EKF implementation. This model assumes that the user (and thus the camera) moves with constant velocity over the time interval for which we must predict the new pose. In this simple model, it will help if we separate the position and orientation into separate expressions. Thus our prediction for the new pose of the camera is

$$s_p^- = s_{0,p}^- + v_p^- \Delta t, \quad s_o^- = s_{0,o}^- + v_o^- \Delta t$$

where $s_{0,p}^-$ is the estimated current position, $s_{0,o}^-$ the estimated current orientation, $v_p^-$ the estimated current translational velocity, $v_o^-$ the estimated current angular velocity, and $\Delta t$ the time interval. These quantities determine the predicted position $s_p^-$ and the predicted orientation $s_o^-$.

We compute the true pose of the camera after a time interval of $\Delta t$ passes using the classical second-order equation rigid-body kinematics.

$$\begin{aligned} s_p &= s_{0,p} + v_p \Delta t + \frac{1}{2} a_p (\Delta t)^2 \\ s_o &= s_{0,o} + v_o \Delta t + \frac{1}{2} a_o (\Delta t)^2 \end{aligned}$$

The second-order equations include the translational acceleration $a_p$ and the angular acceleration $a_o$. Note that without the superscript $^-$, we denote the true values, not estimates of them. We then subtract to compute the error incurred by the prediction with this simple model of rigid body motion.

$$\begin{aligned} s_p - s_p^- &= (s_{0,p} - s_{0,p}^-) + (v_p - v_p^-)\Delta t + \frac{1}{2} a_p (\Delta t)^2 \\ s_o - s_o^- &= (s_{0,o} - s_{0,o}^-) + (v_o - v_o^-)\Delta t + \frac{1}{2} a_o (\Delta t)^2 \end{aligned}$$

Let us now look at the symbols that we have accumulated in these formulas and see what we can say about each of these. We know that all the estimated values are currently computed in the algorithm. This is exactly what the EKF maintains in its state vector, denoted simply by $s$ in Section 3.3. We don't have precise values for the true values of the pose, velocity, and acceleration (each of which are six values for motion in 3D). These are the variables for which we are trying to solve in the EKF. However, we can run the simulator with the motion paths acquired in the user study (described in Section 4.1.3) to bound the errors in estimating the position and velocity, and we can measure the maximum and average acceleration factors by sampling the acceleration along the paths. In the simulator, these values are the true values, and thus this analysis is sufficient to analyze the performance of the EKF in the application of interest for these motion paths. For reasons noted in Section 4.1.3, we assume these paths to be realistic for the ultrasound visualization application. Average and maximum values for the relevant quantities appear in Tables 4.5 and 4.6.

The only remaining variable is $\Delta t$, the time interval between measurements. This is a function of the speed of the camera and projectors. This is exactly what we wanted; we now have a mapping between an intrinsic (though adjustable) property of the hardware and the amount of error we can expect in an implementation of the system.

#### 4.1.6.2 Error Analysis of the Corrector Phase

The measure of the corrector phase is how much of the current error is eliminated by the update to the current estimate. This is very difficult to analyze in purely theoretical terms. The factors that affect this are

- the quality of fit that the Jacobian gives to the measurement function $h(s)$ given in Section 3.3.3.4
- the balance between confidence in the dynamic process model, expressed by the matrix $Q$, and confidence in the measurements, expressed by the matrix $R$
- the amount of information available in each measurement

The first of these is very difficult to determine in any form with the complex function $h(s)$ that consists of the projection operation from the projector to the unknown surface, then projection up to the camera image plane, in addition to the numerous parameters of the poses of the camera and projectors and the intrinsic parameters of the camera and projectors. The second depends heavily on the motion paths that were used to tune the EKF for the best performance. According to that experiment, these parameters are already the optimum in the sense that they minimize the objective function of registration error for the control motion path in the tuning experiment. The third depends on the mathematical constraints, a subject to which we will return in Section 4.2.

Since we already analyze the performance on the motion paths from the user study, a more accessible method of determining the amount of correction is to simply measure it in the simulator. Fortunately, this number is very consistent across all experiments, which gives us confidence that this type of measurement is an approximation of sufficient quality that we can perform the analysis we desire. Table 4.7 gives the percentage of the three types of error described in Section 4.1.2 remaining after the corrector phase for the user experiments.

Also note that all of the numbers in Table 4.7 are less than one. This is good evidence that the algorithm will converge in some finite number of predictor-corrector cycles so long as we can bound the velocity and acceleration of the user's head. This bound can be determined, as evidenced by the statistics from the user study. Formal proof of the convergence of the algorithm would require a more detailed analysis of the corrector phase, which we leave as future work. We next turn to the question of the number of cycles necessary for the algorithm to converge.

#### 4.1.6.3 Rate of Convergence

To predict the expected convergence of the algorithm, we express the error after a complete predictor-corrector cycle of the algorithm has run. We use the expression for the error after the predictor phase and multiply by the percentage of correction we expect in the corrector phase. The former requires numbers from Tables 4.5 and 4.6; the latter was presented in Table 4.7.

Take as an example User 6 running with 200 Hz cameras, which is the trial that had the highest acceleration. Using the expressions above for error in prediction, we can write the expected error after the predictor phase for both the position and orientation components of the pose. Also, for this trial, we have $\Delta t = 0.005$ seconds. Note that we must convert the units for position error to meters from millimeters and for orientation error to radians from milliradians, respectively, for the intermediate calculations. At the end, we convert to millimeters for position error and degrees for orientation error.

$$\begin{aligned} s_p - s_p^- &\approx 0.903\text{mm} \cdot 0.001\frac{\text{m}}{\text{mm}} + 0.0300\frac{\text{m}}{\text{s}} \cdot 0.005\text{s} + 0.5 \cdot 1.497\frac{\text{m}}{\text{s}^2} \cdot (0.005\text{s})^2 \approx 1.07\text{mm} \\ s_o - s_o^- &\approx 1.148\text{mr} \cdot 0.001\frac{\text{rad}}{\text{mr}} + 0.0537\frac{\text{rad}}{\text{s}} \cdot 0.005\text{s} + 0.5 \cdot 0.643\frac{\text{rad}}{\text{s}^2} \cdot (0.005\text{s})^2 \approx 0.082^\circ \end{aligned}$$

Borrowing ideas on the analysis of the causes of registration error [Holloway95], we can bound the expected registration error by examining the contribution of position error and orientation error to registration error. Since we have a $640 \times 480$ display with a $28.64^\circ$ horizontal field of view (and square pixels)[3], each degree of the field of view subtends $22.35$ pixels. At a working distance of $500$ mm, a $28.64^\circ$ horizontal field of view subtends $500$ mm$\cdot\tan 28.64^\circ = 273.1$ mm. Thus each millimeter at the working distance subtends $2.34$ pixels in the center of the field of view[4].

$$\text{Registration error } \leq 1.07 \text{ mm} \cdot \frac{640 \text{ pix}}{273.1 \text{ mm}} + 0.082^\circ \cdot \frac{640 \text{ pix}}{28.64^\circ} \approx 4.33 \text{ pixels}$$

Suppose we define convergence of the algorithm as achieving a registration error of under 2.0 pixels on the image plane. For User 6 at 200 Hz, Table 4.7 tells us that we expect to reduce the registration error after the predictor phase to 0.834 times its current value after the corrector phase. Then the number $n$ of corrector phases we can expect the algorithm to need before achieving this goal is determined by solving the following expression for $n$.

$$4.33 \cdot 0.834^n \leq 2.0 \Longrightarrow n \geq \frac{\ln\left(\frac{2.0}{4.33}\right)}{\ln 0.834} \Longrightarrow n \geq 4.3 \text{ frames}$$

Thus it will take five frames on average for the algorithm to converge for the motion path of User 6 if the measurements are taken at 200 Hz. But this analysis explains what would happen if the predictor phase incurred the error only in the first frame and were then perfect in the subsequent four frames. Only in that case would the algorithm actually converge to the correct value. Indeed, in the simulator, if the camera pose is frozen for a finite number of frames, the estimated pose does converge to the true pose, as seen in Section 4.1.4.

[3] These numbers are our best estimates for our current implementation of the AR ultrasound visualization system.

[4] We will use this number despite the fact that the tangent is a nonlinear function and lengths are stretched towards the edges of the field of view.

One way to view this analysis is as bounding the number of frames the algorithm requires estimate the pose with under 2.0 pixels of registration error. By the time the algorithm produces that estimate, however, the current pose will have presumably changed. Thus what this analysis really tells us is the number of frames of **latency** that we expect from the algorithm. Since in this example, the frames were at 5 ms intervals, this implies 40 ms of latency to reach an accuracy of 2.0 pixels of static registration error. This is comparable to the latency we get with several commercial tracking systems in our lab [Jacobs97]. However, according to the analysis of registration error [Holloway95], that could imply as much as 40 mm of dynamic registration error, which for our working distance would be almost 94 pixels of registration error. For such high registration error to occur under the conditions in this example, however, all three types of (position, orientation, latency) would have to create registration error in the same direction on the image plane, which is an unlikely event.

Another way we could view this analysis is as a way to determine what accelerations in position and orientation the camera can undergo before the algorithm's performance suffers. Suppose we are in a steady-state of 1.0 pixels of registration error. The error in registration due to acceleration experienced during the next measurement time is

$$0.5 a_p t^2 \cdot \frac{640 \text{ pix}}{0.2731 \text{ m}} + 0.5 a_o t^2 \cdot \frac{180^\circ}{\pi \text{ rad}} \cdot \frac{640 \text{ pix}}{28.64^\circ}$$

If we wish to have no more than 1.0 pixels of error remaining, then we must have the following expression be true.

$$[1.0 + \text{ error } ] \cdot \text{ correction } \leq 1.0$$

where "error" is given by the expression above and "correction" is given by the numbers in Table 4.7. For User 6 running with 200 Hz cameras, the left-hand side evaluates to

$$\left[1.0 + 0.5 \cdot 1.497 \cdot 0.005^2 \cdot \frac{640}{0.2731} + 0.5 \cdot 0.643 \cdot 0.005^2 \cdot \frac{640 \cdot 180}{28.64 \cdot \pi}\right] \cdot 0.834 = 0.879$$

and we see that performance is good, whereas for User 3 running with 30 Hz cameras, the left-hand side evaluates to

$$\left[1.0 + 0.5 \cdot 1.062 \cdot 0.033^2 \cdot \frac{640}{0.2731} + 0.5 \cdot 0.512 \cdot 0.033^2 \cdot \frac{640 \cdot 180}{28.64 \cdot \pi}\right] \cdot 0.854 = 2.346$$

and we see that performance is poor. By selecting an intersample time $t$, one could compute the maximum acceleration. This would require selecting a priori the amount of registration error due to position error and the amount of registration error due to orientation error, which is, at best, a difficult task. Still, this calculation would give some insight to the maximum acceleration that the algorithm could tolerate without degrading performance.

These registration error calculations for User 1 with 30 Hz cameras will serve to motivate discussion of future directions in Section 5.1.

$$\begin{aligned}
s_p - s_p^- &\approx 17.553\text{mm} \cdot 0.001\frac{\text{m}}{\text{mm}} + 0.0591\frac{\text{m}}{\text{s}} \cdot 0.0333\text{s} + 0.5 \cdot 0.406\frac{\text{m}}{\text{s}^2} \cdot (0.0333\text{s})^2 \\
&\approx 19.75\text{mm} \\
s_o - s_o^- &\approx 24.226\text{mr} \cdot 0.001\frac{\text{rad}}{\text{mr}} + 0.0797\frac{\text{rad}}{\text{s}} \cdot 0.0333\text{s} + 0.5 \cdot 0.244\frac{\text{rad}}{\text{s}^2} \cdot (0.0333\text{s})^2 \\
&\approx 1.55^\circ \\
\text{Registration error} &\leq 19.75 \text{ mm} \cdot \frac{640 \text{ pix}}{273.1 \text{ mm}} + 1.55^\circ \cdot \frac{640 \text{ pix}}{28.64^\circ} \approx 80.9 \text{ pixels} \\
80.9 \cdot 0.878^n &\leq 2.0 \\
\Longrightarrow n &\geq \frac{\ln\left(\frac{2.0}{80.9}\right)}{\ln 0.878} \Longrightarrow n \geq 28.4 \text{ frames}
\end{aligned}$$

This analysis ignored two important factors in determining dynamic registration error. When the camera is moving, the algorithm will try to push the estimated pose towards the current pose, not poses from which past measurements were taken. Also, this example used average errors in the factors affecting predictor error. The registration error at each frame depends on the distribution of these errors, which we have not measured.

This analysis does not offer formal proof of the latency of the filter. To do this, determine the transfer functions for the filter and examine the phase difference as a function of the signal frequency [Maybeck79]. Still, this analysis sheds some insight on the convergence of the algorithm. We have shown how to bound the latency with which the algorithm will converge to the correct solution on average. A similar analysis using maximum errors in the factors affecting prediction would yield a bound for that case. This will be very pessimistic, since we are unlikely to reach the maxima in these errors at every frame. The number of measurements required for each of the trials in the user study is given in Table 4.8.

## 4.2 Theoretical Analysis of Unobservable Configurations

In KF literature, a system is referred to as *unobservable* if one or more parameters cannot be estimated from the constraints. In the linear system $Ax = b$ (where $A$ is an $n \times n$ matrix and $x$ and $b$ are $n \times 1$ vectors), the analogous condition would be that $A$ has rank less than $n$, and thus has no inverse. There are a variety of situations from which such a condition might arise. It may be that a parameter included in the state does not influence the measurements. Such a parameter cannot be estimated from the measurements, which should be an intuitive statement. This argument will be applied to one situation below.

Also of interest in this section, however, will be situations in which the parameters may all be affecting the measurement, yet still there are multiple solutions possible. It is one of the big advantages of the SCAAT formulation of tracking systems [Welch96] that at each time step, the constraint is simple, which will in general also mean that the constraint is not enough to fully constrain the six DOF of pose in the world of the tracked object. It may also be, however, that **no** amount of constraints will be sufficient to constrain the object's pose. In this condition, the tracking problem is underconstrained—that is, unobservable.

There are five specific situations that will be examined in this section. The first is a configuration in which the epipolar line degenerates to a point. The second is a configuration of cameras and projectors for which there is an infinite number of solutions. The third is a configuration of points in the world for which there is a finite number of solutions. The fourth and fifth are examples of numbers of correspondences between the camera and projector image planes for which there are a finite number of solutions.

I want to emphasize two things before undertaking this discussion. First, all of these problems can be overcome by simply avoiding these configurations or adding further constraints. Evidence for why a specific problem can be avoided or overcome will be presented in each discussion. Second, these problems are unlikely to occur or affect the accuracy in implementations of the system. These are selected as specific examples of the kind of problems of which an implementation must be vigilant.

### 4.2.1 Epipolar Line Degenerates to a Point

The epipolar line is the projection onto the camera's image plane of a ray emanating from the projector. It is theoretically possible for this line to degenerate to a point. Algebraically, the equation $Ax + By + C = 0$ of the epipolar line will degenerate to $C = 0$. Given the measurement equation described in Section 3.3.3.4, this implies that the normalization factor applied to the epipolar line will vanish. Thus the measurement and its derivatives are undefined. Recall that the normalization factor is applied to ensure that the distance from the imaged landmark is in a consistent unit (e.g. pixels). The geometric condition in which this will occur is if the camera's center of projection and the camera's image-plane normal are collinear with the ray emanating from the projector. In a physical realization of the proposed system, however, this event is extremely unlikely, if not impossible.

When this configuration occurs, the camera is unlikely to image anything from the projector. Either the camera would be behind the projector and thus the camera's view of the surface patch containing the landmark would be blocked by the projector, or the camera would be in front of the projector and thus the landmark would fall on the back of the camera housing.

However, it is possible with small equipment to be near this condition and thus be in an unstable configuration. This would imply that the noise in the measurements modeled by the $R$ matrix would be incorrect for a measurement taken in this “near-degenerate” configuration. Clearly, this would not be a measurement that one would want to use in the EKF without at least modifying the $R$ matrix in order to weight the measurement less. Fortunately, this condition is easily detectable by measuring the 2D distance between the epipole (computed with an equation given in Section 3.3.3.4 as part of determining the epipolar line) and the imaged landmark. If this distance is below a threshold, then the projection of the epipolar line can be considered too short for a reliable measurement. The parameter of the $R$ matrix can be adjusted to reflect the increased uncertainty, or the measurement can simply be ignored. We implemented this in the simulator, although the problem does not occur frequently enough to reliably measure the improvement offered by increasing the uncertainty.

### 4.2.2 Two Projectors and Camera Are Collinear

We have already seen that the algorithm requires at least two projectors to provide six DOF tracking. Let us suppose that the system is implemented with exactly two projectors. If the camera’s center of projection lies on the line defined by the centers of projection of the two projectors, then the camera pose can only be estimated to lie on that line, but its position along that line cannot be estimated.

Here is one intuitive argument supporting this claim. Refer back to the visualization of the algorithm presented in Section 3.4. This configuration would correspond to the two lines specified in that visualization (one line between the camera’s center of projection and each respective center of projection of the two projectors) becoming the same line. The point of that visualization was that the estimate of the camera’s pose would converge to the intersection of the two lines. However, in this case, the two lines are the same, and thus the intersection is the entire line.

A more rigorous argument uses the fact (noted in Section 2.3.2) that the fundamental matrix is defined only to within a scale factor. In one sense, this scale factor is a parameter of the state that is not reflected in the measurements. We can think of the state with the scale factor $\alpha$ explicitly indicated:

$$\mathbf{s} = \begin{bmatrix} \alpha x_1 & \alpha y_1 & \alpha z_1 & \theta_1 & \phi_1 & \rho_1 & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho \end{bmatrix}^t$$

It is also true that the matrices $U_c$ and $U_p$ that describe the intrinsic parameters of the camera and projector (respectively) are defined only to within a scale factor [McMillan97]. Let us denote these two scale factors by $\gamma_1$ and $\gamma_2$. The combined effect of these three scaling factors will result in a scale factor $\alpha\gamma_1\gamma_2$ being applied to the matrix $F$ as compared to some normalized form of $F$ (e.g. normalized such that the entry in the third column of the third row is 1.0). This scale factor will also multiply the coefficients of the epipolar line.

$$Fm' = \begin{bmatrix} \alpha\gamma_1\gamma_2 A \\ \alpha\gamma_1\gamma_2 B \\ \alpha\gamma_1\gamma_2 C \end{bmatrix}$$

The normalizing factor for the epipolar line will then be

$$\sqrt{(\alpha\gamma_1\gamma_2 A)^2 + (\alpha\gamma_1\gamma_2 B)^2} = \alpha\gamma_1\gamma_2\sqrt{A^2 + B^2}$$

and thus the scaling factor will have no effect on the final, computed line equation. This argument also implies that the scale factors associated with the matrices $U_c$ and $U_p$ do not affect the line equation. This latter result is good; these scale factors are arbitrary, as noted above. The former result, however, implies that so long as these scale factors are the only ones applied to the state, they cannot be estimated, since they do not affect the measurements. That is, if only the projector that gives rise to this scale factor is used, the scale factor cannot be estimated, and thus the estimate of the camera can be constrained only to within a line in space, which is the line defined by the projector's center of projection (a point) and the computed offset to the projector $[\ \alpha x_1\ \alpha y_1\ \alpha z_1\ ]^t$ (a vector, indicating a direction of translation but not a magnitude).

With a second projector at a known pose, however, there is a second way that we think of the state every time that second projector is used to add a constraint. That is a state with a different offset in both the position and orientation.

$$\mathbf{s} = \begin{bmatrix} \beta x_2 & \beta y_2 & \beta z_2 & \theta & \phi & \rho & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho \end{bmatrix}^t$$

If the camera is collinear with both projectors, then we can write

$$\beta \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \zeta\alpha \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

and by an argument analogous to the previous one, the multiplier $\zeta$ between $\alpha$ and $\beta$ will not affect measurements. Thus we are again only measuring $[\ x_1\ y_1\ z_1\ ]^t$, which defines the direction to the projector. We still have no information with which to measure $\alpha$, the magnitude of translation.

This gives an argument for the necessity of two projectors and to the condition that the system is unobservable if there are exactly two projectors and the true camera pose lies on the line defined by the two centers of projection of those projectors. This argument should be tempered by noting that unless the camera stays on or near this line (as opposed to merely passing through it), the unobservable condition is not maintained, and the tracking will not be disturbed. We can define "staying on the line" more precisely by using the $P$ matrix which contains the uncertainty of the current estimates of the state parameters. Only if the values in the $P$ matrix that give the uncertainty of the position parameters grow continuously will the tracking be affected.

A final comment is necessary on this configuration. Given that the configuration is unobservable, it follows that configurations "near" this one, such as the camera being near the line containing the two projector centers of projection, are necessarily unstable. If the camera is a small distance away from this line, there will be very little information about where the camera lies along the direction of the line. This could lead to problems in tracking the pose of the camera along the line, since there will be little information in that direction. This will be evidenced by greater uncertainty in that direction in the $P$ matrix). This may lead the EKF to attribute the error to the other parameters and make corrections in a suboptimal direction at a given time step. This problem can be avoided by adding more projectors in such a way that no three are collinear, or by placing the projectors in poses that the user cannot reasonably reach, such as on the ceiling.

### 4.2.3 World-space Points on a Critical Surface

It can be shown that three distinct camera offsets from the projector can be computed from the correspondence of a set of points that lie on certain quadric surfaces [Maybank92a]. A summary of that proof (which discusses reconstruction of motion of a single, calibrated camera) is given here and adapted for the case of determining pose of the camera with respect to a projector.

Recall the epipolar constraint $m^T F m' = 0$. Suppose that we are given a set of points in correspondence $m \leftrightarrow m'$. Let us suppose that we have two candidate solutions for the pose of the camera, described by the fundamental matrices $F$ and $G$. Note that the only difference between $F$ and $G$ is in that portion of the matrix determined by the essential matrix—i.e. not to a difference in the intrinsic parameter matrices for the camera or for the projectors.

Let us assume, without loss of generality, that the projector is located at the origin. It follows from the epipolar geometry that

$$(Fm \wedge Gm) \parallel m'.$$

That is, since $m'$ is perpendicular to both $Fm$ and $Gm$ (by the epipolar constraint), it must necessarily be parallel to another vector that is itself perpendicular to both $Fm$ and $Gm$. If we write the two candidate camera poses as (1) rotation $R$ and translation $a$ and (2) rotation $S$ and translation $b$, then the projective expansion of this parallel condition is

$$m' = R[((Sb \wedge Sm) \cdot Ra)m - ((Sb \wedge Sm) \cdot Rm)a]$$

We want to place conditions on the world space point $M$ that generates the correspondence $m \leftrightarrow m'$. Let us use the scale factor $x$ to denote the distance from the projector pose to the point $M$, $xm = M$ and the scale factor $y$ to denote the distance from the first candidate camera pose to $M$, $ym' = M$. Another way of stating the epipolar constraint is to express that the point $M$ is fixed in space:

$$ym' = R(xm - a)$$

If we solve this equation for $x$ and substitute $M = xm$, we get a quadric surface.

$$(Sb \wedge Sm) \cdot Rm - (Sb \wedge Sm) \cdot Ra = 0$$

This surface is known as the *critical surface*. It expresses an arrangement of points in the world that, when used as the points of correspondence between two image planes (the camera and projector), yields two different poses for the camera. A second equation is obtained by interchanging the roles of the displacements $(R, a)$ and $(S, b)$.

Note that this surface contains the origin (the projector position), and both candidate camera positions $a$ and $b$. It also contains the lines from the origin to $a$ and $b$. Of the quadrics that are non-singular under affine collineations, only hyperboloids of one sheet can contain lines[5]. Certain degenerate forms of hyperboloids of one sheet, such as elliptic cones and hyperbolic paraboloids, can also be critical surfaces.

The question then arises of how many candidate poses for the camera there can be. The answer is that there can be three such poses. This is proven by showing that two candidate poses uniquely determine a third pose. This in turn is done by expressing the quadric surface with the three displacements and proving equality of the surfaces [Maybank92a]. An alternate method of proof is to use the space of essential matrices [Faugeras93]. This is a manifold embedded in an eight-dimensional projective space (i.e. the projective space described by a nine-tuple of numbers, which is only defined up to a single scale factor). The shape of the projective space comes from the structure of the essential matrix: a $3 \times 3$ matrix defined up to a scale factor. The manifold is de-

[5] Virtual quadrics contain no real points. Ellipsoids, elliptic paraboloids, and hyperboloids of two sheets cannot contain lines.

fined by properties of essential matrices, notably that the determinant of an essential matrix must be zero. This implies that the equations defining the manifold are of degree three, there can be at most three intersections of the manifold with a line in the eight-dimensional projective space.

This condition is of course unlikely to occur in practical operation of the proposed tracking system. However, by definition of the projection operator that defines the projector (and the camera) geometrically, if the points (or $n$ most recent points) selected in the projector image plane form an ellipse in that image plane, then the world-space points must lie on an elliptic cone. In addition, it is known [Maybank92a] that if the world-space points are "near" a critical surface, then the computation of the camera motion is unstable. This could create difficulties for an EKF, which will need to disambiguate among the possible solutions. It may be that the EKF will already be close enough to the correct solution that it will converge to it, but the multiple solutions could be close enough to lead the EKF to an incorrect solution.

Also note that for this condition to occur over some finite length of time, the camera must remain on the critical surface. Also note unless the projectors are aligned in restricted geometric configurations (e.g. sharing the optical axis), this condition cannot occur simultaneously for multiple projectors. It may be that the constraints from a second projector eliminate one or both of the incorrect candidate solutions.

Even if this reduces the potential solutions, the fact remains that at a given time instant, it may be impossible to tell which of as many as three solutions is correct. This ambiguity could cause the EKF to move the estimate of the camera parameters towards an incorrect solution, affecting the rate of convergence and stability of the algorithm.

### 4.2.4 Five Image-plane Correspondences

Another configuration that can demonstrate the types of difficulties that an algorithm using the epipolar constraint must overcome is to look at a restricted set of projected landmarks. This is not a practical case in the sense that an implementation would never purposefully limit itself in this manner, but this discussion does lead to greater insight of the issues presented in the discussion of theoretical limitations.

Suppose we are restricted to using only five points in the image plane of each projector. Then there are ten solutions available for the direction between the camera and each projector. Note that I say "direction" to indicate that the scale factor that is unobservable between a camera and a single projector is neither observable in this situation nor accounting for the multiple solutions in this case. This fact was first proven in 1913, although an error in that proof was not corrected until later [Faugeras93]. The proof involves the construction two sixth-degree curves in the homogeneous plane. Two curves of degree $m$ and $n$ will have in general $mn$ intersections. Thus we will have 36 potential solutions, and we will eliminate some of those as impossible.

Let us express the five pairs of corresponding points as follows [Faugeras93].

$$a \leftrightarrow a', b \leftrightarrow b', c \leftrightarrow c', d \leftrightarrow d', e \leftrightarrow e'$$

We also have the (unknown) epipoles $x$ and $x'$ in the two images. There is in each plane an image of the *absolute conic*, the conic that is invariant under affine transformations [Faugeras93]. To this conic from the epipole there are two tangents, say $u$ and $v$ in the first image plane and $u'$ and $v'$ in the second. We know that these tangents are in correspondence, although whether $u \leftrightarrow u'$ and $v \leftrightarrow v'$, or $u \leftrightarrow v'$ and $v \leftrightarrow u'$ is unknown. Finally, we have the epipolar lines associated with each of the ten points, five in each image plane.

We want to consider the intersections of the epipolar lines and the tangents with the line defined by $a$ and $b$ in the first image plane and the line defined by $a'$ and $b'$ in the second image plane. These intersections must be related by a homography $H$. This homography has the following form for corresponding points $y$ and $y'$.

$$\rho y' = Hy = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} y$$

We apply this homography to the intersection points of the three epipolar lines defined by $c$, $d$, and $e$ (and the corresponding points in the other image plane), and to the intersection points of the tangents to the absolute conic. Taking ratios of these five equations gives us four quartic equations that express ratios between the coordinates of the points. Some simplification and substitution yields two sextic equations [Faugeras93].

The last step is then to eliminate spurious solutions. In the simplification process, there were ratios taken. We eliminate seven solutions that make the divisors vanish. We have a collineation between two homogeneous planes. We eliminate twelve solutions that fall in the nullspace of this collineation. There are denominators in the two sextic equations which we cannot allow to vanish. This yields six more solutions to eliminate (plus two solutions that made the divisors vanish and were already eliminated). That leaves $36 - 7 - 12 - 6 = 11$ solutions. It turns out that one of these solutions is of multiplicity two, leaving us with ten solutions. A complete proof of this number of solutions includes an algorithm to compute the solutions [Faugeras93].

Again, it should be noted that this condition would not occur in a practical implementation of the proposed system, since it would use many more than five points. Even configurations “near” this one of only five points in each image—say five clusters of points—are unlikely to occur in any practical implementation. But note that we take measurements one at a time. This implies that the five most recent measurements (landmarks) projected from a single projector and imaged by the camera will not yield a unique solution to the offset between that projector. If we assume that measurements are taken from the set of projectors in an alternating (or, for more than two projectors, round-robin) manner, then it will require six full rounds with one measurement from each projector to fully constrain the camera position. (Of course, an implementation need only find the camera pose with two projectors, but a round-robin implementation may prove to be more stable. It may also eliminate the configurations discussed earlier in this section, in the unlikely event that they occur.) This gives us insight into the rate of convergence of the tracking algorithm. There is a limit to the amount of information that exists in each constraint. This limit causes the algorithm to be slow to converge.

### 4.2.5 Six Image-plane Correspondences

This is in a sense the limiting case between the case in which there as many as ten solutions given five point correspondences and the case of ambiguous surfaces, in which there may be as many as three solutions. Using geometric invariants, it can be shown [Quan94] that with six correspondences, there are three invariants in the image plane which can be computed. By writing the sixth point as a projective linear combination of the other five points (which without loss of generality form the canonical basis of the three-dimensional projective space[6]) we can see that the three invariants are the ratios of the four coordinates of the sixth (projective) point.

With an image of these six points, one can write three quadratic equations in these four coordinates that use the six image-plane coordinates as coefficients, but none of these quadratic terms are squared terms [Quan94]. These three quadratic surfaces must intersect in $2^3 = 8$ points. Four of these points are vertices of the tetrahedron of reference (which by construction are the first four points in the canonical basis). The fifth point is determined by the fact that for each quadratic equation, there is a constraint (repetitive between each quadratic) that the $i^{\text{th}}$ coordinates of each point must sum to zero. This implies that the fifth point in the canonical basis is also an intersection point. This leaves us with $8 - 5 = 3$ solutions remaining.

Given the methods for computing the invariants, this implies that with six pairs of image-plane points in correspondence, we can compute the offset between the projector and camera up to, in some cases, a three-way ambiguity. This establishes the minimum number of pairs of points in correspondence to achieve the minimum number of solutions for any configuration of points. (Note that some configurations of points will yield a unique solution or only two solutions.)

This three-way ambiguity is maintained as long as the camera remains on the critical surface (the hyperboloid of one sheet or its degenerate form) [Maybank98]. It can also be determined by image metrics whether a new image of the same six points will break the ambiguity or not.

[6] The canonical basis is $[\,1\;0\;0\;0\,]^T, [\,0\;1\;0\;0\,]^T, [\,0\;0\;1\;0\,]^T, [\,0\;0\;0\;1\,]^T, [\,1\;1\;1\;1\,]^T$.

| Shutter | Location | | Variance | |
|---|---|---|---|---|
| time | $u$ | $v$ | $\sigma_u$ | $\sigma_v$ |
| 16 ms | 481.2 | 43.8 | 0.5 | 0.6 |
| | 479.6 | 45.1 | 0.2 | 0.3 |
| | 457.1 | 6.3 | 1.1 | 0.8 |
| | 425.7 | 44.9 | 0.6 | 0.3 |
| | 314.3 | 115.5 | 0.9 | 0.4 |
| | 251.6 | 121.8 | 1.3 | 0.6 |
| | 573.7 | 19.3 | 0.4 | 0.4 |
| | 432.8 | 23.0 | 0.6 | 0.4 |
| | 590.4 | 37.9 | 0.6 | 0.5 |
| | 243.1 | 31.2 | 0.9 | 0.9 |
| | 416.5 | 39.2 | 0.6 | 0.3 |
| | 535.7 | 25.0 | 0.7 | 0.4 |
| | 479.6 | 45.2 | 0.3 | 0.3 |
| 4 ms | 472.7 | 52.5 | 0.6 | 0.5 |
| | 582.2 | 45.3 | 0.5 | 0.6 |
| | 418.5 | 52.8 | 0.4 | 0.5 |
| | 409.6 | 46.9 | 0.5 | 0.6 |
| | 437.9 | 8.9 | 0.5 | 0.2 |
| | 448.9 | 15.2 | 0.6 | 0.6 |
| | 308.0 | 123.5 | 0.4 | 0.3 |
| | 246.7 | 130.3 | 0.5 | 0.4 |
| | 565.1 | 27.0 | 0.6 | 0.4 |
| | 425.7 | 30.9 | 0.4 | 0.6 |
| | 582.3 | 45.2 | 0.5 | 0.3 |
| | 236.8 | 40.4 | 0.5 | 0.4 |
| | 527.4 | 33.4 | 0.4 | 0.3 |
| 4 ms | 482.6 | 52.5 | 0.5 | 0.5 |
| | 592.0 | 45.5 | 0.7 | 0.5 |
| | 420.0 | 46.5 | 0.6 | 0.4 |
| | 428.6 | 52.4 | 0.5 | 0.6 |
| | 447.8 | 8.7 | 0.5 | 0.2 |
| | 458.7 | 14.8 | 0.7 | 0.5 |
| | 317.9 | 123.1 | 0.5 | 0.3 |
| | 256.5 | 129.7 | 0.5 | 0.5 |
| | 574.6 | 26.6 | 0.5 | 0.3 |
| | 435.8 | 30.5 | 0.5 | 0.5 |
| | 591.7 | 45.2 | 0.4 | 0.3 |
| | 246.8 | 39.6 | 0.6 | 0.3 |
| | 537.2 | 33.2 | 0.5 | 0.3 |

Table 4.2: Variance of the landmark locations in the detection experiment using both normal and fast shutter times. This demonstrates that our assumption that we can locate the pixel to within 1.0 pixels of noise should hold with the strategy for imperceptible structured light presented in Section 1.3.1.

| Location | | Variance | | Percentage |
|---|---|---|---|---|
| $u$ | $v$ | $\sigma_u$ | $\sigma_v$ | detected |
| 419.3 | 114.5 | 0.32 | 0.18 | 100% |
| 232.1 | 189.3 | 0.16 | 0.11 | 100% |
| 297.4 | 183.1 | 0.17 | 0.10 | 100% |
| 400.2 | 112.8 | 0.21 | 0.09 | 96% |
| 410.2 | 118.8 | 0.29 | 0.19 | 100% |
| 418.9 | 95.5 | 0.23 | 0.05 | 100% |
| 430.0 | 75.5 | 0.26 | 0.03 | 96% |
| 441.9 | 81.0 | 0.22 | 0.05 | 100% |
| 530.4 | 88.5 | 0.19 | 0.12 | 96% |
| 571.8 | 78.7 | 0.29 | 0.19 | 100% |

Table 4.3: Landmark noise on the non-planar, non-white surface shown in Figure 4.6. This environment was constructed to be similar to the primary intended application, that of ultrasound-guided needle biopsy. Landmarks 2, 4, 5, 8, and 10 imaged inside the medical model; the other landmarks imaged on the curtain.

| User | Camera Speed<br>Hertz | Registration Error<br>pixels | Position Error<br>mm (avg) | Orientation Error<br>mr (avg) |
|---|---|---|---|---|
| 1 | 200 | 0.84 | 1.18 | 1.41 |
| | 120 | 1.96 | 2.52 | 2.95 |
| | 60 | 5.56 | 6.94 | 8.15 |
| | 30 | 15.66 | 17.55 | 24.23 |
| 2 | 200 | 0.69 | 1.46 | 1.72 |
| | 120 | 1.39 | 2.98 | 3.60 |
| | 60 | 4.60 | 8.64 | 10.63 |
| | 30 | 10.83 | 22.66 | 28.38 |
| 3 | 200 | 0.84 | 1.38 | 1.72 |
| | 120 | 1.98 | 2.95 | 3.72 |
| | 60 | 5.43 | 8.02 | 10.22 |
| | 30 | 14.23 | 18.20 | 23.15 |
| 4 | 200 | 0.54 | 0.92 | 1.11 |
| | 120 | 1.15 | 1.85 | 2.34 |
| | 60 | 3.12 | 4.94 | 6.42 |
| | 30 | 7.78 | 12.24 | 16.60 |
| 5 | 200 | 0.92 | 1.40 | 1.58 |
| | 120 | 1.87 | 2.85 | 3.24 |
| | 60 | 6.02 | 8.58 | 9.78 |
| | 30 | 12.18 | 16.69 | 20.62 |
| 6 | 200 | 0.64 | 0.90 | 1.15 |
| | 120 | 1.37 | 1.93 | 2.50 |
| | 60 | 3.61 | 5.30 | 6.87 |
| | 30 | 8.05 | 11.73 | 15.78 |
| 7 | 200 | 0.48 | 0.89 | 1.03 |
| | 120 | 1.00 | 1.86 | 2.21 |
| | 60 | 2.63 | 4.95 | 5.78 |
| | 30 | 6.76 | 12.39 | 14.95 |

Table 4.4: Performance metrics for the user study. Each metric was measured at four different camera speeds, using the optimal noise parameters determined for that speed given in Table 4.1. The registration error is the RMS error over all frames, which is higher than the per-frame average. One could argue that this is unnecessarily pessimistic since the user will see error in units of the per-frame average, not in RMS units. The position error is given in millimeters and is the per-frame average of the position error. Similarly, the orientation error listed is the per-frame average of the orientation error, in milliradians.

| User | Camera | Average | | | Maximum | | |
|---|---|---|---|---|---|---|---|
| | Speed | $s_{0,p} - s^{-}_{0,p}$ | $v_p - v^{-}_p$ | $a_p$ | $s_{0,p} - s^{-}_{0,p}$ | $v_p - v^{-}_p$ | $a_p$ |
| | Hz | mm | $\frac{\mathrm{m}}{\mathrm{s}}$ | $\frac{\mathrm{m}}{\mathrm{s}^2}$ | mm | $\frac{\mathrm{m}}{\mathrm{s}}$ | $\frac{\mathrm{m}}{\mathrm{s}^2}$ |
| 1 | 200 | 1.18 | 0.0711 | 0.514 | 14.35 | 0.517 | 8.21 |
| | 120 | 2.52 | 0.0707 | 0.497 | 23.48 | 0.490 | 7.54 |
| | 60 | 6.94 | 0.0679 | 0.463 | 65.88 | 0.511 | 6.78 |
| | 30 | 17.55 | 0.0591 | 0.406 | 103.64 | 0.395 | 4.92 |
| 2 | 200 | 1.46 | 0.0653 | 1.066 | 9.47 | 0.280 | 15.90 |
| | 120 | 2.98 | 0.0660 | 0.985 | 21.50 | 0.290 | 14.00 |
| | 60 | 8.64 | 0.0639 | 0.806 | 55.77 | 0.275 | 11.89 |
| | 30 | 22.66 | 0.0624 | 0.593 | 109.79 | 0.209 | 7.52 |
| 3 | 200 | 1.38 | 0.0663 | 1.062 | 16.10 | 0.426 | 36.30 |
| | 120 | 2.95 | 0.0667 | 0.998 | 27.64 | 0.437 | 30.92 |
| | 60 | 8.02 | 0.0671 | 0.871 | 77.03 | 0.421 | 21.05 |
| | 30 | 18.20 | 0.0617 | 0.677 | 138.17 | 0.275 | 5.77 |
| 4 | 200 | 0.92 | 0.0357 | 1.090 | 10.29 | 0.391 | 14.30 |
| | 120 | 1.85 | 0.0362 | 0.993 | 16.36 | 0.267 | 12.61 |
| | 60 | 4.94 | 0.0357 | 0.829 | 30.74 | 0.249 | 9.71 |
| | 30 | 12.24 | 0.0329 | 0.567 | 81.30 | 0.150 | 5.54 |
| 5 | 200 | 1.40 | 0.0590 | 0.892 | 13.48 | 0.595 | 7.01 |
| | 120 | 2.85 | 0.0589 | 0.836 | 39.94 | 0.559 | 5.77 |
| | 60 | 8.58 | 0.0586 | 0.731 | 104.76 | 0.520 | 5.27 |
| | 30 | 16.69 | 0.0447 | 0.562 | 110.69 | 0.311 | 4.49 |
| 6 | 200 | 0.90 | 0.0300 | 1.497 | 5.31 | 0.259 | 25.66 |
| | 120 | 1.93 | 0.0315 | 1.351 | 12.23 | 0.252 | 19.93 |
| | 60 | 5.30 | 0.0331 | 1.080 | 31.88 | 0.243 | 13.49 |
| | 30 | 11.73 | 0.0316 | 0.656 | 66.33 | 0.226 | 6.23 |
| 7 | 200 | 0.89 | 0.0280 | 1.043 | 9.56 | 0.274 | 29.87 |
| | 120 | 1.86 | 0.0284 | 0.955 | 16.87 | 0.276 | 18.57 |
| | 60 | 4.95 | 0.0296 | 0.794 | 52.53 | 0.240 | 14.90 |
| | 30 | 12.39 | 0.0283 | 0.556 | 113.41 | 0.207 | 10.91 |

Table 4.5: Average and maximum values of the parameters of position estimation error from the user experiments. Note that the units for the position error are millimeters, to avoid printing unnecessary decimal places, while the units for the velocity and acceleration terms are meters per second and meters per second squared, respectively.

| User | Camera | Average | | | Maximum | | |
|---|---|---|---|---|---|---|---|
| | Speed | $s_{0,o} - s_{0,o}^{-}$ | $v_o - v_o^{-}$ | $a_o$ | $s_{0,o} - s_{0,o}^{-}$ | $v_o - v_o^{-}$ | $a_o$ |
| | Hz | mr | $\frac{\text{rad}}{\text{s}}$ | $\frac{\text{rad}}{\text{s}^2}$ | mr | $\frac{\text{rad}}{\text{s}}$ | $\frac{\text{rad}}{\text{s}^2}$ |
| 1 | 200 | 1.412 | 0.0788 | 0.274 | 15.41 | 0.5195 | 3.44 |
| | 120 | 2.945 | 0.0783 | 0.270 | 33.29 | 0.5151 | 3.19 |
| | 60 | 8.150 | 0.0785 | 0.261 | 61.45 | 0.5067 | 2.26 |
| | 30 | 24.226 | 0.0797 | 0.244 | 123.62 | 0.4715 | 1.98 |
| 2 | 200 | 1.720 | 0.0833 | 0.467 | 13.43 | 0.6627 | 6.23 |
| | 120 | 3.595 | 0.0821 | 0.441 | 25.19 | 0.6602 | 5.38 |
| | 60 | 10.626 | 0.0812 | 0.375 | 82.40 | 0.6708 | 3.93 |
| | 30 | 28.384 | 0.0808 | 0.321 | 169.77 | 0.5378 | 3.68 |
| 3 | 200 | 1.723 | 0.0983 | 0.512 | 19.84 | 0.9309 | 11.04 |
| | 120 | 3.715 | 0.0965 | 0.490 | 41.42 | 0.9520 | 9.35 |
| | 60 | 10.219 | 0.0954 | 0.448 | 87.76 | 0.8369 | 6.10 |
| | 30 | 23.146 | 0.0892 | 0.383 | 190.11 | 0.7170 | 4.21 |
| 4 | 200 | 1.106 | 0.0447 | 0.451 | 14.47 | 0.7500 | 8.89 |
| | 120 | 2.338 | 0.0431 | 0.417 | 21.52 | 0.6694 | 5.47 |
| | 60 | 6.424 | 0.0430 | 0.361 | 53.92 | 0.5102 | 4.15 |
| | 30 | 16.604 | 0.0431 | 0.270 | 103.36 | 0.2084 | 2.22 |
| 5 | 200 | 1.580 | 0.0585 | 0.456 | 14.12 | 0.4910 | 4.43 |
| | 120 | 3.239 | 0.0569 | 0.435 | 44.30 | 0.4550 | 3.39 |
| | 60 | 9.782 | 0.0555 | 0.394 | 112.30 | 0.4265 | 3.18 |
| | 30 | 20.620 | 0.0479 | 0.329 | 136.22 | 0.2716 | 2.91 |
| 6 | 200 | 1.148 | 0.0537 | 0.643 | 7.73 | 0.4557 | 12.84 |
| | 120 | 2.503 | 0.0535 | 0.584 | 18.49 | 0.4465 | 12.31 |
| | 60 | 6.866 | 0.0550 | 0.470 | 56.48 | 0.4209 | 5.91 |
| | 30 | 15.780 | 0.0541 | 0.307 | 106.01 | 0.3926 | 2.69 |
| 7 | 200 | 1.026 | 0.0319 | 0.450 | 11.24 | 0.4868 | 8.35 |
| | 120 | 2.212 | 0.0313 | 0.414 | 20.74 | 0.4759 | 7.07 |
| | 60 | 5.782 | 0.0316 | 0.352 | 50.24 | 0.3685 | 6.79 |
| | 30 | 14.947 | 0.0331 | 0.257 | 86.62 | 0.2817 | 4.84 |

Table 4.6: Average and maximum values of the parameters of orientation estimation error from the user experiments. Again, take note of the units of orientation error, milliradians, and the units for angular velocity and angular acceleration, radians per second and radians per second squared, respectively.

| User | Camera | Position | Orientation | Registration |
|---|---|---|---|---|
| | Speed | Correction | Correction | Correction |
| 1 | 200 | 0.903 | 0.908 | 0.875 |
| | 120 | 0.899 | 0.898 | 0.872 |
| | 60 | 0.920 | 0.922 | 0.872 |
| | 30 | 0.939 | 0.918 | 0.878 |
| 2 | 200 | 0.910 | 0.911 | 0.855 |
| | 120 | 0.923 | 0.929 | 0.858 |
| | 60 | 0.953 | 0.948 | 0.863 |
| | 30 | 0.958 | 0.955 | 0.872 |
| 3 | 200 | 0.905 | 0.902 | 0.853 |
| | 120 | 0.912 | 0.915 | 0.856 |
| | 60 | 0.938 | 0.938 | 0.861 |
| | 30 | 0.950 | 0.946 | 0.854 |
| 4 | 200 | 0.896 | 0.889 | 0.840 |
| | 120 | 0.914 | 0.920 | 0.841 |
| | 60 | 0.957 | 0.960 | 0.850 |
| | 30 | 0.971 | 0.963 | 0.856 |
| 5 | 200 | 0.904 | 0.906 | 0.859 |
| | 120 | 0.907 | 0.912 | 0.861 |
| | 60 | 0.937 | 0.946 | 0.872 |
| | 30 | 0.952 | 0.954 | 0.879 |
| 6 | 200 | 0.898 | 0.903 | 0.834 |
| | 120 | 0.924 | 0.936 | 0.838 |
| | 60 | 0.967 | 0.969 | 0.856 |
| | 30 | 0.976 | 0.974 | 0.869 |
| 7 | 200 | 0.898 | 0.892 | 0.832 |
| | 120 | 0.914 | 0.918 | 0.839 |
| | 60 | 0.960 | 0.965 | 0.845 |
| | 30 | 0.965 | 0.969 | 0.860 |

Table 4.7: Percentage of error after a predictor phase that remained after the subsequent corrector phase measured in the user experiments for each of the three types of error.

| Camera Speed | 200 | 120 | 60 | 30 |
|---|---|---|---|---|
| User | | | | |
| 1 | 8.1 | 13.0 | 20.1 | 28.4 |
| 2 | 8.0 | 12.6 | 20.1 | 28.5 |
| 3 | 7.8 | 12.6 | 19.5 | 23.5 |
| 4 | 4.4 | 8.4 | 14.9 | 21.3 |
| 5 | 7.6 | 12.3 | 21.2 | 27.8 |
| 6 | 4.3 | 8.5 | 16.0 | 23.4 |
| 7 | 3.7 | 8.0 | 14.0 | 21.7 |

Table 4.8: Number of frames with landmarks successfully detected in order to achieve convergence to under 2.0 pixels of registration error. These numbers are according to the performance model for the user experiments. With the camera speed, these numbers can be converted to a bound on the number of milliseconds of latency.

# Chapter 5

# Extensions

In this chapter, I discuss extensions to the proposed system. This chapter is separated from the previous two since none of the work described here has been implemented, even in simulation. However, the specification is complete enough to separate it from a mere listing of future directions for research, as appears in Chapter 6. There are two qualitative types of extensions. One extension, suggested by the analysis in Section 4.1.6, analyzes the performance improvement we could expect by adding other sensors, such as accelerometers or rate gyroscopes, to an implementation using normal-speed cameras. The other extensions allow the algorithm to estimate quantities, such as the camera intrinsic parameters, projector parameters, and surface geometry, in real time. All these parameters will affect the measurements. In theory, any or all of these parameters can be added to the state and estimated simultaneously. I will discuss each of these in turn and give new state vectors and state transition matrices for each of these quantities.

## 5.1 Adding Inertial Sensors to Decrease Dynamic Error

The user study in Section 4.1.5 showed that the algorithm performed well in simulation with 200 Hz cameras. It may not be practical to require this specialized equipment in order to achieve the low registration error we require in our intended application. Thus the first extension we consider is the addition of inertial sensors. Such devices have been demonstrated to reduce dynamic registration error in AR systems [Azuma95, Foxlin98].

With the analysis in Section 4.1.6, we see that there can be significant error in the predictor phase of the EKF-based algorithm. While most of this error on a given iteration can be attributed to pose error that remained after the previous corrector phase, this does explain how the error arose after having reached a converged (correct) solution. That pose error resulted from being unable to correct the error incurred in previous predictor phases. We incur such error even on the first iteration in which the camera moves after the pose has been correctly estimated. The error in pose prediction results from not having accurate estimates of the velocity and acceleration of the components of the camera pose. This can occur in two ways: applying the "best" model of dynamic motion with poor estimates of the velocities and/or accelerations, or in applying a higher-order model than is necessary. That is, if the user is nearly still, then a model that incorporates velocity is prone to introduce error until it can properly estimate the velocity to be nearly zero. Thus one appealing direction for further research is to improve the predictor phase of the algorithm by incorporating inertial sensors into the system.

Suppose we add translational accelerometers to directly measure components of translational acceleration or angular rate gyroscope (ARG) to directly measure angular velocity. That should reduce the error in the pose that is predicted. Work in our lab [Weber98] has shown that the devices can be read at rates of 60 to 100 Hz without introducing a phase shift. At higher rates, the data is out of phase, although this error can be corrected with filtering. Even the moderate rates for which the device can be read without introducing the phase shift allow the acceleration to be integrated to accurately compute relative position.

By reducing the error of the predictor phase, we should be able to reduce the error that remains after the corrector phase. This would reduce the final registration error in a given iteration that results from error in the predicted pose. It is also possible to infer a change in orientation by measuring multiple independent components of translational acceleration, since the accelerometers will not measure acceleration around the same point. We do not have experimental evidence of the accuracy we could expect in measuring orientation with this strategy, however, and thus we will leave analysis of such a system for future work.

By incorporating the accelerometers, we should also be able to reduce the error in the estimated translational velocity; however, we are not yet sure by what amount. We will thus assume that the error in translational velocity is no worse than when we estimate it only from the camera's view of the structured light. Finally, we of course would have a better estimate of the acceleration than the numerical differentiation of position that would occur in the EKF. The acceleration error would be due only to error in the accelerometers (around $0.06$ meters per second squared for many commercial accelerometers [Titterton97, Verplaetse95]) and to change in the acceleration since the previous accelerometer measurement. We measure this last quantity in the simulator, at 0.0336 meters per second squared per second.

If we were to add ARGs to the user's head, we could expect to reduce the orientation component of the error that we incur in the predictor phase. Commercial ARGs achieve errors in angular velocity of as low as $0.5°$ per second [Titterton97, Crossbow98]. Additionally, the change in velocity during the time interval will cause error in our prediction. This of course, is the acceleration, so we have already measured this quantity. Similar to the case of estimating translational velocity from translational accelerometers, we would expect the use of ARGs to improve the estimation of angular acceleration, but we have no evidence of what the improvement or final accuracy might be.

Let us return to the example of User 1 with 30 Hz cameras we examined in Section 4.1.6. Previous work in our lab has applied translational accelerometers to tracking of the user's head position [Azuma95]. That system combined the original version of the UNC optical ceiling tracker (with the hardware described in Section 2.2.2.6 and the algorithm described in Section 2.2.3.2) with inertial sensors in two ways simultaneously:

- with accelerometers in order to predict head position
- with rate gyroscopes in order to predict head orientation

That system reduced the error in position prediction to $1.1$ mm and the error in orientation prediction to $0.18°$ on average for a 60 ms prediction interval. Another simulation tracked a missile with an EKF-based inertial navigation system aided by one of a laser range finder, a multi-function radar detector, or an infrared tracker, fixed on the ground [Titterton97]. That simulation found that the standard deviation of the error in pitch and yaw would each be under $0.14°$ and the standard deviation of the error in position transverse to the direction of motion would rise to eleven meters after ten seconds, which is ten percent above the noise of ten meters assumed in the auxiliary sensor measurements. (Ten percent above our noise of one pixel with $\frac{273.1}{640}$ millimeters per pixel is about 0.5 millimeters.)

Thus if we repeat the calculation of the average errors in position and in orientation after the predictor phase for the case in which we have accelerometers and ARGs on the user's head, we get the following equations.

$$\begin{aligned}
s_p - s_p^- &\approx 0.0005\,\text{m} + 0.0591\,\frac{\text{m}}{\text{s}} \cdot 0.0333\,\text{s} + 0.5 \cdot (0.06\,\frac{\text{m}}{\text{s}^2} + 0.0336\,\frac{\text{m}}{\text{s}^3} \cdot 0.0333\,\text{s}) \cdot (0.0333\text{s})^2 \\
&\approx 2.5\,\text{mm} \\
s_o - s_o^- &\approx 0.14° + (\frac{0.5°}{\text{s}} + \frac{13.98°}{\text{s}} \cdot 0.0333\text{s}) \cdot 0.0333\text{s} + 0.5 \cdot \frac{13.98°}{\text{s}} \cdot (0.0333\text{s})^2 \\
&\approx 0.18°
\end{aligned}$$

We then convert this to registration error and to latency as before. We assume that we maintain the same reduction of registration error during the corrector phase as before (to 0.878 times the error before the corrector phase, according to Table 4.7). We get a result of the number

$$\begin{aligned}
\text{Registration error} &\leq 2.5\,\text{mm} * \frac{640\text{ pix}}{273.1\text{ mm}} + 0.18° \cdot \frac{640\text{ pix}}{28.64°} \approx 9.9\text{ pixels} \\
9.9 \cdot 0.878^n &\leq 2.0 \\
\Longrightarrow n &\geq \frac{\ln\left(\frac{2.0}{9.9}\right)}{\ln 0.878} \Longrightarrow n \geq 12.3\text{ frames}
\end{aligned}$$

This is far less registration error than we previously computed as the expected bound, and we have cut the expected dynamic registration error (i.e. the latency) to less than half the previous amount. Note that according to Figures 4.9 and 4.10, we rarely experience the amount of registration error this analysis predicts, since it assumes that the position error and orientation error yield registration error in the same direction.

If we were to add accelerometers to the system, we would need to change the state vector and state transition matrix, as well as write a new measurement function. The state vector and state transition matrix for adding estimated acceleration would be as follows.

$$\mathbf{s} = \left[ \begin{array}{cccccccccccccccccc} x & y & z & \theta & \phi & \rho & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho & a_x & a_y & a_z & a_\theta & a_\phi & a_\rho \end{array} \right]^t$$

$$A_{\text{acc}} = \left[ \begin{array}{cc|c} \multicolumn{2}{c|}{A} & \frac{1}{2}\Delta t \cdot I_6 \\ \cline{3-3} & & \Delta t \cdot I_6 \\ \hline 0_6 & \multicolumn{1}{|c|}{0_6} & I_6 \end{array} \right]$$

In order to improve the readability, we give the state transition matrix in block form, using the matrix $A$ given in Section 3.3.3.1 for the state transition in the basic algorithm as a submatrix. $I_n$ represents the $n \times n$ identity matrix and $0_n$ the $n \times n$ zero matrix.

Let us assume that we have three translational accelerometers mounted in a mutually orthogonal arrangement. The new measurements are simply the accelerations given by the three accelerometers. These accelerations are given in a coordinate system that is relative to the camera coordinate system. Thus far, we have tacitly assumed that we are storing in the state the position and velocity relative to the world coordinate system. Thus our prediction for the measurements would be the current estimated accelerations rotated into the world coordinate system.

$$a_{\text{pred}} = R_{WC}^{-1} \left[ \begin{array}{c} a_x \\ a_y \\ a_z \end{array} \right]$$

If the accelerometers are not mounted orthogonally to each other, then the above equation would need to be modified to account for this distortion. This difference between orthogonality and the true directions could be reflected by applying the same difference to the rotation matrix.

Similarly, if we add ARGs to the system, then we need to predict their velocity measurements. Again, if we assume that the ARGs are mounted orthogonally, then we use the velocity components stored in the state, add in the acceleration, and rotate it into the world coordinate system.

$$v_{\text{pred}} = R_{WC}^{-1} \begin{bmatrix} v_\theta + \Delta t \; a_\theta \\ v_\phi + \Delta t \; a_\phi \\ v_\rho + \Delta t \; a_\rho \end{bmatrix}$$

The measurements given above are vectors. Note that this still fits into the EKF framework, even though in Chapter 3, the measurement was in fact a scalar. The addition of accelerometers and ARG could also be specified with a series of scalar measurements. This may more accurately reflect the relative times at which the devices can be read in a particular implementation.

Angular accelerometers can directly measure one of the quantities missing in the above example. Current technology can achieve errors in the range of 0.07 radians per second squared [Titterton97]. However, if we use this error in angular acceleration to recompute the expected worst-case registration error, it drops only to 9.75 pixels, and the latency drops only to 12.2 frames. Thus, one would not expect angular accelerometers to improve performance greatly over translational accelerometers and ARG. If the ARG were removed from the system and use of both translational and angular accelerometers were considered, one would expect improved performance over using just the translational accelerometers. However, the accuracy of estimating the angular velocity from the optical measurements and the angular accelerometer measurements is an open question.

This analysis is meant merely to provide insight to future research. The analysis shows that with a standard rate camera, the system would be greatly improved by using at least translational accelerometers and one of ARG or angular accelerometers.

## 5.2 Estimating Surface Geometry

The most interesting quantity that can be estimated with this technique is the geometry of the surface patches that contain landmarks. I have said repeatedly throughout this dissertation that we do not want to rely on a priori knowledge of the surface shape. However, if we accurately compute the camera pose, then we can use that data with the correspondence determined to triangulate the position of the surface point. The uncertainty estimates of the state variables in the EKF can tell us whether we can expect to accurately compute the surface geometry.

This would imply that we maintain a surface model and update it with points. Of course, if the surface is dynamic, then we would need to account for this in the surface model. However, we have already seen a good tool for estimating this type of variable—the extended Kalman filter. We could incorporate variables into the state specified in Chapter 3 or use a separate EKF to estimate the surface geometry. A sinusoidal dynamic model that simulates the motion of the skin surface of a breathing patient could be measured by a conventional tracking system and a set of LED affixed to a human subject.

The most interesting aspect of this extension is the implication it has for an AR system. With this information, the AR system would have exactly the data it needs to properly depict occlusion relationships between real and synthetic elements of the scene. This is because with the surface model it becomes trivial to compute depth to the real surface at any pixel of the camera image. A particularly clever algorithm for selecting projector pixels might predict where in the camera image a landmark might be detected and provide depth data at exactly those pixels at which the system needs to paint synthetic imagery. This would avoid the expense of computing a complete depth map that some AR systems have computed, as discussed in Section 2.1. With even a few points, a 2D convex hull algorithm and a rasterization algorithm with an intelligent interpolation scheme might provide a reasonable image of the real surfaces in the depth buffer.

It should be stated very clearly that this is an **extremely** complex task. While similar tasks are attempted in computer vision under the name of motion-and-structure algorithms, it is not clear that the error propagation from the “motion” portion of the computation to the “structure” portion of the computation can be managed, or that a simultaneous computation in this framework would be stable. These algorithms have not to my knowledge used multiple sensors to determine the camera pose, so it is likely that an implementation of this system that incorporates accelerometers or rate gyroscopes would be able to perform these computations. Because of the vital importance of the data to creating a convincing AR environment, I believe this is a promising direction for future research, as discussed in Section 6.2.

## 5.3 Estimating Intrinsic Camera Parameters

The intrinsic camera parameters are those parameters that are invariant to the pose of the camera in the environment, but determine the projection operator that maps the world onto the camera image plane, as defined in Section 2.3. Camera calibration, or determining these parameters, is a difficult problem with even the best off-line methods [Faugeras93]. However, at least one recursive algorithm to estimate the focal length in real-time has been presented [Azarbayejani95]. Simultaneously estimating the intrinsic parameters and the pose (extrinsic parameters) is known in the computer vision literature as *self-calibration*.

In order to incorporate these parameters into the current framework for estimating the pose, we must extend the state vector and correspondingly add rows and columns to the state transition matrix, the process covariance matrix, and the process noise covariance matrix. Of course, the size of the Kalman gain matrix will also change, but since this matrix is defined by the equations of the filter, I will not consider it here. Also, the measurement function $h(s^-)$ remains the same, since the parameters added to the state here were already properly accounted for in the function given in Section 3.3.3.4. In that discussion, however, these parameters were regarded as known, fixed constants. We now are regarding them as constants to be estimated, with some initial estimate given. Thus the system must now compute derivatives with respect to these parameters and extend the Jacobian to include them.

Camera models vary widely [McMillan97]. At a minimum, the field of view, aspect ratio[1], and focal length must be included. I will include image-plane offsets between the image-plane origin and the central optical ray. As noted in Section 2.3, other parameters may be included. Including those parameters in the following equations would be little extra effort conceptually, but would significantly add to the density of the equations. The following is an example state vector and dynamic model to include the camera intrinsic parameters.

- the image-plane width $w_C$
- the image-plane height $g_C$
- the horizontal field of view $\xi_C$

$$\mathbf{s} = \begin{bmatrix} x & y & z & \theta & \phi & \rho & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho & w_C & g_C & \xi_C \end{bmatrix}^t$$

$$A_{\text{intr}} = \left[\begin{array}{c|c} A & 0_3 \\ \hline 0_3 & I_3 \end{array}\right]$$

The intrinsic parameters for an AR system are quite likely to be constants or change infrequently, in which case the dynamic model can treat them as constants.

Camera calibration is a requirement for VST AR. Although the camera parameters likely do not change during operation of most systems today, focus can be a powerful depth cue and the ability to change it during a performance of a task may well prove beneficial. It would be a worthwhile effort to allow the camera's focal length to be adjusted in real-time in order to determine whether there is benefit in allowing this. Currently, this would be a time-consuming procedure given the difficulty of camera calibration. The algorithm proposed in this dissertation, however, provides an excellent framework for real-time estimation of focal length and other intrinsic parameters. However, this algorithm has not been tested, and self-calibration is sometimes regarded as an unstable process. Also note that this change has introduced (further) non-linearity to $h(s)$. What affect,

[1] The aspect ratio is the ratio of the horizontal to vertical image size. Specifying the horizontal and vertical fields of view is equivalent to specifying a single field of view and the aspect ratio. The single field of view can be the horizontal, vertical, or diagonal.

if any, this has on the convergence or stability of the algorithm is a direction for further research. As with the surface computation, it is unclear how (or whether) multiple sensors that give information regarding camera pose would improve the stability of self-calibration.

## 5.4 Estimating Intrinsic and Extrinsic Projector Parameters

*Autocalibration* refers to the measurement of parameters that are internal to the tracking system (i.e. that the user need not know), but affect the operation of the system. In this case, these parameters include the following.

- the matrices $U_i$ that describe the intrinsic parameters of the projectors
- the matrices $R_{WP,i}$ that rotate the world to $P_i$
- the vectors $T_{WP,i}$ that translate the world to $P_i$

Like the intrinsic parameters of the camera, these parameters are also already properly incorporated into the measurement function, so once again all that is necessary is to extend the state vector and the matrices of the EKF with the appropriate symbols and expressions. These parameters can be considered constants.

$$\mathbf{s} = \begin{bmatrix} x & y & z & \theta & \phi & \rho & v_x & v_y & v_z & v_\theta & v_\phi & v_\rho & w_i & g_i & \xi_i & x_i & y_i & z_i & \theta_i & \phi_i & \rho_i \end{bmatrix}^t$$

$$A_{\text{proj}} = \left[ \begin{array}{c|c} A & 0_9 \\ \hline 0_9 & I_9 \end{array} \right]$$

Once again, the extra rows of the state transition matrix have 1.0 on the diagonal and 0.0 elsewhere, reflecting the dynamic model which assumes these extra values to be constant.

Autocalibration has shown to work for both the old [Gottschalk93] and new [Welch97] versions of the UNC optical ceiling tracker, although in the initial version, it was an off-line process.

# Chapter 6

# Summary

This chapter summarizes the results of the dissertation, directions for further research, and its contribution to AR.

## 6.1 Summary

AR, as noted in Section 1.1, has yet to receive acceptance for many applications, principally because of the problems achieving registration and designing a suitable HMD. The goal of this dissertation was to find a better way to solve the problem of achieving registration than existing solutions. Two significant drawbacks of our current vision-based tracking system are that it restricts scene movement and that it restricts landmark placement.

While no optical system can eliminate these restrictions, we have significantly reduced them with an approach that does not require the landmarks to be at known 3D locations. By requiring knowledge of only the 2D positions of the landmarks, we have significantly reduced restrictions on motion within the environment. Since the landmarks are no longer physical objects but projected objects, we can move them around the environment dynamically. We have removed the landmarks from the set of physical obstacles that might impede the user's progress in the application. For example, certain placements of the colored landmarks for our vision-based system required the physician to reach around a target in order to perform a biopsy. Other placements were susceptible to having the landmarks occluded during the course of the procedure, degrading the quality of the registration achieved. These were clearly problems we needed to overcome.

This has been done without sacrificing the accuracy of the system. We are still relying only on our ability to accurately detect landmarks on the image plane of the camera, which is exactly the measurement on which we currently rely. The algorithm uses the current estimated pose to predict a line on which the detected landmark should lie, and makes a correction to that estimate depending on the distance of the detected landmark from the line. If the distance is zero—i.e. the point is indeed on the line—then we learn no new information about the camera pose from that measurement. This predictor-corrector framework comes from our choice to use the EKF. The correction mechanism within the EKF uses the Jacobian of the point-to-line distance function along with estimates of the covariance of the parameters to determine the direction in which to move the estimate in the multi-dimensional parameter space.

We believe this work has built a solid foundation on which we will be able to extend our work with tracking systems. We now address future research.

## 6.2 Future work

First and foremost, we plan to implement the system. We currently have projectors and cameras that we believe are suitable. Accounting for the delay between the time we instruct the projector to display a pattern and the time we receive the corresponding image from the camera, however, has proven to be a somewhat more difficult engineering problem than we had anticipated. This will enable us to implement the strategy for imperceptible structured light described in Section 1.3.1. We plan to test the tracking system with the AR ultrasound visualization system described in Section 2.1.2.4.

Using multiple point landmarks per frame or small 2D landmarks might improve the stability of the algorithm. Intuitively, one might think that multiple landmarks or 2D landmarks would offer more information content per frame than a single point landmark. Whether this information could be distributed in such a way as to truly add useful data to the computations remains an open question. Recall that one intuitive argument for two projectors was that the epipolar lines associated with multiple landmarks generated by one projector were close to being parallel. Thus more landmarks per frame may not add much more useful information. The distribution of epipolar lines varies significantly with the relative pose of the camera with respect to the projector. Thus this strategy may be useful only when the camera is in certain poses with respect to the projector. Also note that this idea would require the algorithm to be able to solve the correspondence problem. This is not trivial, but it may be that certain geometric relationships between the camera and projector make the task easier. If not, color or shape may be of use.

The analysis of the algorithm presented in Section 4.1.6 did not attempt to use semantic information about the task or analysis of the collected motion data to analyze the quality of the motion prediction. A more detailed theoretical analysis of motion prediction [Azuma95] might improve the accuracy of the analysis. A theoretical analysis of the corrector phase might tell us whether there is more information to be had in the camera images we get, something that we have felt intuitively but cannot see how to do—e.g. can we extract the angle towards the “correct” 2D location for the landmark on the camera image plane?

Chapter 5 gave four extensions to the system. The first was the addition of inertial sensors to improve the accuracy of the system, which appears to be necessary according to the analysis in Section 4.1.6. Given the specialized nature of the cameras the system would require without inertial sensors, this extension offers the potential for a cost-effective implementation.

The method allows the incorporation *self-calibration*. This term is defined in computer vision literature as the simultaneous calibration of the intrinsic and extrinsic parameters of the camera. These terms were defined in Section 2.3. This task has been performed successfully with similar algorithms [Azarbayejani95, Maybank92b].

The framework also allows for auto-calibration [Welch96] of the pose of the second (and successive) projector(s) and the intrinsic parameters of the projector(s). These values are fixed, but difficult to calibrate accurately. They are internal to the system, and should not be of concern to the user. With the proposed framework, however, we can refine the estimates of these parameters. This can reduce the calibration effort required to set up the system.

The proposed system already possesses exactly the information required to determine point-wise scene structure dynamically: relative pose and correspondence between image plane points. This enables the system to compute depth relationships for the AR system to properly determine occlusion between real and synthetic objects at every camera pixel which has (recently) seen a projected landmark. Given the success of similar filter-based algorithms for scene reconstruction, one could imagine the inclusion of scene structure in the state to be estimated by this algorithm. However, for an AR system, it is only necessary to know the depth for pixels at which synthetic objects are painted. The algorithm could select projector pixels which, based on computed surface information, are likely to image at the camera pixels that synthetic objects occupy. This would give an efficient depth map that, while not necessarily complete, would cover (in the best case), exactly those (camera) pixels at which the system needs the depth. A hole-filling algorithm similar to a flood fill painting algorithm might be a useful tool in the case of a few scattered pixels of depth data.

The ability to dynamically move landmarks within the scene opens up the possibility to try to place landmarks at scene locations or at camera image locations that are most likely to give new information (in directions that have not recently been sampled) or to improve the stability. Automatic selection of landmarks can lead to a high frequency of landmark sightings. Infrequent landmark sightings have limited previous vision-based systems.

## 6.3 Contribution

As a by-product of our design, we require no information about the geometry of the surface at which the user is looking, and we allow the surface to change shape arbitrarily. The only requirement is that the surface must reflect enough light from the projector into the camera in order to accurately detect the landmark in the camera image. According to our plans to use imperceptible structured light and the preliminary experiment to determine the expected noise (described in Section 4.1.5), this should reduce to finding a single bright point in a binary image. This is a much simpler task than the search for landmarks of a specific color or pair of colors that we now use [State96a]. The search in the proposed system is aided by the epipolar geometry and the prediction mechanism in the algorithm.

We have also increased the range of viewpoints for which we can acquire tracking data. The landmarks must be in view to acquire tracking information, but with the strategy for dynamic placement of landmarks, we increase the area over which landmarks can be distributed without recalibration of the system. Also, the system is completely scalable in the number of projectors, so one could easily add another projector to increase the tracking volume. This would require another instance of the projector-to-projector calibration described in Section 3.3.2.2.

Compared to our previous vision-based tracking system, landmark occlusion is less disruptive. With the recursive formulation based on the EKF and acquiring a single measurement at a time, failure to detect a landmark is not a catastrophic error. It merely fails to reduce the error covariance that results after the time-step update to the estimated state.

An AR system must calibrate the tracking sensor to the video cameras that provide the VST view. By adding the tracking role to the cameras, we remove this difficult procedure from the requirements of an AR system. This comes at the cost of increasing the complexity of the tracking system, but this seems a reasonable sacrifice given the lack of success we have had achieving registration in AR with current tracking systems.

With the strategy of dynamic structured light patterns, we create a way to avoid the expensive task of identifying correspondence between landmarks that limits the performance of many computer vision algorithms that use point-to-point correspondence data to compute the camera pose. This allows our system to run in real time, which is beyond the speed of most computer vision algorithms that use this constraint. The simplicity of the computation at each time step in the EKF is a strength of the proposed algorithm.

The update rate of the system is limited only by the acquisition rate of the camera and the refresh rate of the projectors. Both types of devices have witnessed great performance improvements recently. We believe there will be numerous inexpensive devices suitable for this system in the near future.

We are hopeful of building this system soon, given the impressive potential of the algorithm that we have seen in simulation. We believe that once the system is implemented, it will open many doors to future research, not only the ideas given above, but also by enabling new applications of augmented reality.

# Bibliography

[AOA98] Adaptive Optics Associates, Incorporated (1998). `http://www.aoainc.com/`.

[Aliaga94] Aliaga, D. G. (1994). *Virtual and Real Object Collisions in a Merged Environment*. In *Proceedings of Virtual Reality Software and Technology (VRST) '94*, Pages 287–298.

[Ascension98] Ascension Technology Corporation (1998). `http://www.ascension-tech.com/`.

[Azarbayejani95] Azarbayejani, A. and Pentland, A. P. (1995). *Recursive Estimation of Motion, Structure, and Focal Length*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):545–561.

[Azuma95] Azuma, R. T. (1995). *Predictive Tracking for Augmented Reality*. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

[Azuma97] Azuma, R. T. (1997). *A Survey of Augmented Reality*. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385.

[Azuma91] Azuma, R. T. and Ward, M. (1991). *Space Resection by Collinearity: Mathematics Behind the Optical Ceiling Head-Tracker*. Technical Report TR91-048, Department of Computer Science, University of North Carolina at Chapel Hill.

[Bajura92] Bajura, M., Fuchs, H., and Ohbuchi, R. (1992). *Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient*. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, Volume 26, Pages 203–210.

[Bajura95] Bajura, M. and Neumann, U. (1995). *Dynamic Registration Correction in Video-Based Augmented Reality Systems*. *IEEE Computer Graphics and Applications*, 15(5):52–60.

[Barfield95] Barfield, W., Hendrix, C., Bjorneseth, O., Kaczmarek, K. A., and Lotens, W. (1995). *Comparison of Human Sensory Capabilities with Technical Specifications of Virtual Environment Equipment*. *Presence: Teleoperators and Virtual Environments*, 4(4):329–356.

[Berger97] Berger, M.-O. (1997). *Resolving Occlusion in Augmented Reality: a Contour Based Approach without 3D Reconstruction*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, Pages 91–96. IEEE, IEEE Computer Society Press.

[Besl89] Besl, P. J. (1989). *Active Optical Range Imaging Sensors*, In *Advances in Machine Vision*, Chapter 1, Pages 1–63. Springer-Verlag.

[Bhatnagar91] Bhatnagar, D., Pujari, A. K., and Seetharamulu, P. (1991). *Static Scene Analysis using Structured Light*. *Image and Vision Computing*, 9(2):82–87.

[Breen96] Breen, D. E., Whitaker, R. T., Rose, E., and Tuceryan, M. (1996). *Interactive Occlusion and Automatic Object Placement for Augmented Reality*. In *Proceedings of Eurographics '96*, Pages C–11–C–22. The Eurographics Association.

[Brown92] Brown, R. G. and Hwang, P. Y. (1992). *Introduction to Random Signals and Applied Kalman Filtering*. Wiley and Sons, 2nd edition.

[Caprile90] Caprile, B. and Torre, V. (1990). *Using Vanishing Points for Camera Calibration. International Journal of Computer Vision*, 4:127–140.

[Caudell92] Caudell, T. P. and Mizell, D. W. (1992). *Augmented Reality: An Application of Heads Up Display Technology to Manual Manufacturing Processes*. In *Proceedings of Hawaii International Conference on System Sciences*, Volume II, Pages 659–669. IEEE Computer Society Press.

[Chen90] Chen, C., Hung, Y., Chiang, C., and Wu, J. (1997). *Range Data Acquistion using Color Structured Lighting and Stereo Vision*. *Image and Vision Computing*, 15:445–456.

[Cho98] Cho, Y. and Neumann, U. (1998). *Multi-ring Color Fiducial Systems for Scalable Fiducial Tracking Augmented Reality*. In *Proceedings of IEEE 1998 Virtual Reality Annual International Symposium (VRAIS '98)*, Page 212.

[CIWG97] Covariance Intersection Working Group (1997). A culminating advance in the theory and practice of data fusion, filtering, and decentralized estimation. Available at `http://www.ait.nrl.navy.mil/people/uhlmann/CovInt.html`.

[Crossbow98] Crossbow Technology, Inc. (1998). `http://www.xbow.com/`.

[Daley95] Daley, R. C., Hassebrook, L. G., Stanley C. Tungate, J., Jones, J. M., Reisig, H. T., Reed, T. A., Williams, B. K., Daugherty, J. S., and Bond, M. (1995). *Topographical Analysis with Time Modulated Structured Light*. *SPIE Proceedings*, 2488(5):396–407.

[Davies97] Davies, E. (1997). *Machine Vision: Theory, Algorithms, Practicalities*. Academic Press.

[Debevec98] Debevec, P. (1998). *Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography*. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, Pages 189–198. ACM SIGGRAPH, Addison Wesley.

[DePiero96] DePiero, F. W. and Trivedi, M. M. (1996). *3-D Computer Vision using Structured Light: Design, Calibration, and Implementation Issues*. *Advances in Computers*, 43:243–278.

[Deyst68] Deyst, Jr., J. J. and Price, C. F. (1968). *Conditions for Asymptotic Stability of the Dicrete Minimum-Variance Linear Estimator*. *IEEE Transactions on Automatic Control*, Pages 702–705.

[Drascic93] Drascic, D., Grodski, J. J., Milgram, P., Ruffo, K., Wong, P., and Zhai, S. (1993). *ARGOS: A Display System for Augmenting Reality*. In *Video Proceedings of INTERCHI '93: ACM Conference on Human Factors in Computing Systems*.

[Durlach95 (Chap. 5)] Durlach, N. I. and (ed.), A. S. M. (1995). *Virtual Reality: Scientific and Technological Challenges*. Report of the Committee on Virtual Reality Research and Development to the National Research Council.

[Edwards93] Edwards, E. K., Rolland, J. P., and Keller, K. P. (1993). *Video See-through Design for Merging of Real and Virtual Environments*. In *IEEE Virtual Reality Annual International Symposium*, Pages 223–233. IEEE.

[Emura94] Emura, S. and Tachi, S. (1994). *Compensation of Time Lag between Actual and Virtual Spaces by Multi-Sensor Integration*. In *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI '94)*, Pages 463–469.

[Fakespace98] Fakespace, Incorporated (1998). `http://www.fakespace.com`.

[Faro98] FARO Technologies, Incorporated (1998). `http://www.faro.com/`.

[Faugeras93] Faugeras, O. (1993). *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press.

[Faugeras92] Faugeras, O. D., Luong, Q.-T., and Maybank, S. J. (1992). *Camera Self-Calibration: Theory and Experiments*. In *Proceedings of the Second European Conference on Computer Vision*, Pages 321–334.

[Faugeras87] Faugeras, O. D., Lustman, F., and Toscani, G. (1987). *Motion and Structure from Point and Line Matches*. In *International Conference on Computer Vision*, Pages 25–33.

[Faul98] Faul, I. (1998). *A Theoretical Comparison of 2-Camera and 3-Camera Optical Localizers with Active or Passive Instrumentation*. In *Medicine Meets Virtual Reality 6*, Pages 284–290. IOS Press.

[Feiner93] Feiner, S., MacIntyre, B., and Seligmann, D. (1993). *Knowledge-based Augmented Reality*. *Communications of the ACM*, 36(7):52–62.

[Feiner95] Feiner, S. K., Webster, A. C., III, T. E. K., MacIntyre, B., and Keller, E. J. (1995). *Architectural Anatomy*. *Presence: Teleoperators and Virtual Environments*, 4(3):318–325.

[Ferrin91] Ferrin, F. J. (1991). *Survey of Helmet Tracking Technologies*. In *SPIE Proceedings Vol. 1456 Large-Screen Projection, Avionic, and Helmet-Mounted Displays*, Pages 86–94.

[Fitzmaurice93] Fitzmaurice, G. W. (1993). *Situated Information Spaces and Spatially Aware Palmtop Computers*. *Communications of the ACM*, 36(7):38–49.

[Foley90] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley.

[Fournier95] Fournier, A. (1995). *Illumination Problems in Computer Augmented Reality*. Technical report, University of British Columbia, Vancouver, British Columbia, Canada.

[Foxlin96] Foxlin, E. (1996). *Inertial Head-Tracker Sensor Fusion by a Complimentary Separate-Bias Kalman Filter*. In *Proceedings of IEEE 1996 Virtual Reality Annual International Symposium (VRAIS '96)*, Pages 185–194.

[Foxlin98] Foxlin, E., Harrington, M., and Pfeifer, G. (1998). *Constellation™: A Wide-range Wireless Motion-tracking System for Augmented Reality and Virtual Set Applications*. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, Pages 371–378. ACM SIGGRAPH, Addison Wesley.

[Fuchs77] Fuchs, H., Duran, J., and Johnson, B. (1977). *A System for Automatic Acquistion of Three-dimensional Data*. In *Proceedings of the 1977 National Computer Conference*, Volume 46, Pages 49–53.

[Fuchs98] Fuchs, H., Livingston, M. A., Raskar, R., Colucci, D., Keller, K., State, A., Crawford, J. R., Rademacher, P., Drake, S. H., and Anthony A. Meyer, M. (1998). *Augmented Reality Visualization for Laparoscopic Surgery*. In *Proceedings of First International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer-Verlag.

[Furness86] Furness, T. (1986). *The Super Cockpit and its Human Factors Challenges*. In *Proceedings of the Human Factors Society*, Pages 48–52.

[Gottschalk93] Gottschalk, S. and Hughes, J. F. (1993). *Autocalibration for Virtual Environments Tracking Hardware*. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, Volume 27, Pages 65–72.

[Grimson95] Grimson, W., Ettinger, G., White, S., Gleason, P., Lozano-Pérez, T., III, W. W., and Kikinis, R. (1995). *Evaluating and Validating an Automated Registration System for Enhanced Reality Visualization in Surgery*. In *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95)*,.

[Hartley95] Hartley, R. I. (1995). *In Defence of the 8-point Algorithm*. In *Fifth International Conference on Computer Vision*, Pages 1064–1070. IEEE.

[Hawkes97] Hawkes, D. (1997). *Virtual Reality and Augmented Reality in Medicine*, In *The Perception of Visual Information, 2nd edition*, Chapter 12, Pages 361–390. Springer.

[Herring96] Herring, T. A. (1996). *The Global Positioning System*. *Scientific American*, Pages 44–50.

[Holloway95] Holloway, R. L. (1995). *Registration Errors in Augmented Reality Systems*. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

[Hornbeck95] Hornbeck, L. J. (1995). *Digital Light Processing and MEMS: Timely Convergence for a Bright Future*. In *Micromachining and Microfabrication '95*. Plenary Session Paper.

[Hu93] Hu, X. and Ahuja, N. (1993). *Sufficient Conditions for Double or Unique Solution of Motion and Structure*. *CVGIP: Image Understanding*, 58(2):161–176.

[IGT98] Image Guided Technologies, Incorporated (1998). `http://www.imageguided.com/`.

[Immersion98] Immersion Corporation (1998). `http://www.immerse.com/`.

[Intersense98] InterSense Incorporated (1998). `http://www.isense.com/`.

[Iu96] Iu, S.-L. and Rogovin, K. W. (1996). *Registering Perspective Contours with 3-D Objects without Correspondence, using Orthogonal Polynomials*. In *Proceedings of the 3rd IEEE Virtual Reality Annual International Symposium (VRAIS '96)*, Pages 37–44. IEEE Computer Society Press.

[Jacobs97] Jacobs, M. C., Livingston, M. A., and State, A. (1997). *Managing Latency in Complex Augmented Reality Systems*. In *1997 Symposium on Interactive 3D Graphics*, Pages 49–54. ACM SIGGRAPH.

[Jancène95] Jancène, P., Neyret, F., Provot, X., Tarel, J.-P., Vézien, J.-M., Meilhac, C., and Verroust, A. (1995). *RES: Computing the Interactions Between Real and Virtual Objects in Video Sequences*. In *Proceedings of Networked Realities Conference*. IEEE.

[Janin93] Janin, A. L., Mizell, D. W., and Caudell, T. P. (1993). *Calibration of Head-Mounted Displays for Augmented Reality Applications*. In *IEEE Virtual Reality Annual International Symposium*, Pages 246–255. IEEE.

[Jezouin90] Jezouin, J. and Ayache, N. (1990). *3D Structure from a Monocular Sequence of Images*. In *International Conference on Computer Vision*, Pages 441–445.

[Kaiser98] Kaiser Electro-Optics, Inc. (1998). `http://www.keo.com/`.

[Kakez97] Kakez, S., Conan, V., and Bisson, P. (1997). *Virtual Documented Environments: A New Interface Paradigm for Task-oriented Access to Information*. In *Proceedings of Eurographics '97*, Fellner, D. and Szirmay-Kalos, L., editors, Pages C–319–C–327. The Eurographics Association, Blackwell Publishers.

[Kalman60] Kalman, R. (1960). *A New Approach to Linear Filtering and Prediction Problems*. *Transactions of the ASME–Journal of Basic Engineering*, Pages 35–45.

[Kelly86] Kelly M.D., P. J., Kall, B., and Goerss, S. (1986). *Computer-assisted Stereotaxic Resection of Intra-Axial Brain Neoplasms*. *Journal of Neurosurgery*, 64:427–439.

[Kim97] Kim, D., Richards, S. W., and Caudell, T. P. (1997). *An Optical Tracker for Augmented Reality and Wearable Computers*. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS '97)*, Pages 146–150.

[Klinker97] Klinker, G. J., Ahlers, K. H., Breen, D. E., Chevalier, P.-Y., Crampton, C., Greer, D. S., Koller, D., Kramer, A., Rose, E., Tuceryan, M., and Whitaker, R. T. (1997). *Confluence of Computer Vision and Interactive Graphics for Augmented Reality*. *Presence: Teleoperators and Virtual Environments*, 6(4):433–451.

[Koller97] Koller, D., Klinker, G., Rose, E., Breen, D., Whitaker, R., and Tuceryan, M. (1997). *Real-time Vision-based Camera Tracking for Augmented Reality Applications*. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST–97)*, Pages 87–94.

[KSC-TR] KSC-TR. *Digital Low-Pass Filter without Phase Shift*. Technical Report KSC-11471, John F. Kennedy Space Center. NASA Tech Briefs.

[Kuipers80] Kuipers, J. B. (1980). *SPASYN–An Electromagnetic Relative Position and Orientation Tracking System*. *IEEE Transactions on Instrumentation and Measurement*, IM-29(4):462–466.

[Kumar94] Kumar, R. and Hanson, A. R. (1994). *Robust Methods for Estimating Pose and a Sensitivity Analysis*. *CVGIP: Image Understanding*, 60(3):313–342.

[Kutulakos96] Kutulakos, K. N. and Vallino, J. (1996). *Affine Object Representations for Calibration-Free Augmented Reality*. In *Proceedings of the 3rd IEEE Virtual Reality Annual International Symposium (VRAIS '96)*, Pages 25–36. IEEE Computer Society Press.

[Lavoie96] Lavoie, P., Ionescu, D., and Petriu, E. M. (1996). *3-D Object Model Recovery from 2-D Images using Structured Light*. In *IEEE Instrument Measurement Technology Conference*, Pages 377–382.

[McMillan97] Leonard McMillan, J. (1997). *An Image-Based Approach to Three-Dimensional Computer Graphics*. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

[Livingston97] Livingston, M. A. and State, A. (1997). *Magnetic Tracker Calibration for Improved Augmented Reality Registration*. *Presence: Teleoperators and Virtual Environments*, 6(5):532–546.

[Logitech98] Logitech Incorporated (1998). `http://www.logitech.com/`.

[Longuet-Higgins81] Longuet-Higgins, H. C. (1981). *A Computer Algorithm for Reconstructing a Scene from Two Projections*. *Nature*, 293:133–135.

[Longuet-Higgins84] Longuet-Higgins, H. C. (1984). *The Reconstruction of a Scene from Two Projections – Configurations that Defeat the 8-Point Algorithm*. In *Proceedings of the First Conference on Artificial Intelligence Applications*. IEEE.

[Lorensen93] Lorensen, W., Cline, H., Nafis, C., Kikinis, R., Altobelli, D., and Gleason, L. (1993). *Enhancing Reality in the Operating Room*. In *Proceedings of IEEE Visualization '93*.

[Loscos98] Loscos, C., Drettakis, G., and Robert, L. (1998). *Interactive Modification of Real and Virtual Lights for Augmented Reality*. In *SIGGRAPH 98 Conference Abstracts and Applications*, Page 265. ACM SIGGRAPH.

[Luong95] Luong, Q.-T. and Faugeras, O. (1996). *The Fundamental Matrix: Theory, Algorithms, and Stability Analysis*. *The International Journal of Computer Vision*, 17(1):43–75.

[Madritsch96] Madritsch, F. and Gervautz, M. (1996). *CCD-Camera Based Optical Beacon Tracking for Virtual and Augmented Reality*. In *Proceedings of Eurographics '96*, Pages C–207–C–216. The Eurographics Association, Blackwell Publishers.

[Manhart93] Manhart, P. K., Malcolm, R. J., and Frazee M.D., J. G. (1993). *Augeye: A Compact, Solid, Schmidt Optical Relay for Helmet Mounted Displays*. In *IEEE Virtual Reality Annual International Symposium*, Pages 234–245. IEEE.

[Maybank92a] Maybank, S. J. (1992a). *Theory of Reconstruction from Image Motion*. Springer-Verlag.

[Maybank92b] Maybank, S. J. and Faugeras, O. D. (1992b). *A Theory of Self-Calibration of a Moving Camera*. *The International Journal of Computer Vision*, 8(2):123–151.

[Maybank98] Maybank, S. J. and Shashua, A. (1998). *Ambiguity in Reconstruction from Images of Six Points*. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[Maybeck79] Maybeck, P. S. (1979). *Stochastic Models, Estimation, and Control*. Academic Press.

[Mellor95] Mellor, J. (1995). *Realtime Camera Calibration for Enhanced Reality Visualization*. In *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95),*, Pages 471–475.

[Melzer97] Melzer, J. E. and Moffitt, K., editors (1997). *Head-mounted Displays: Designing for the User*. McGraw-Hill.

[Meyer92] Meyer, K., Applewhite, H. L., and Biocca, F. A. (1992). *A Survey of Position-trackers*. *Presence: Teleoperators and Virtual Environments*, 1(2):172–200.

[Mizell98] Mizell, D. (1998). Private communication.

[Mulder94] Mulder, A. (1994). *Human Movement Tracking Technology*. Technical Report 94-1, School of Kinesiology, Simon Fraser University.

[Nandhakumar92] Nandhakumar, N. and Aggarwal, J. K. (1992). *Multisensory Computer Vision*. *Advances in Computers*, 34:59–111.

[Neumann96] Neumann, U. and Cho, Y. (1996). *A Self-Tracking Augmented Reality System*. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, Pages 109–115.

[Neumann98] Neumann, U. and Park, J. (1998). *Extendible Object-Centric Tracking for Augmented Reality*. In *Proceedings of IEEE 1998 Virtual Reality Annual International Symposium (VRAIS '98)*, Pages 148–155.

[NDI98] Northern Digital Incorporated (1998). `http://www.ndigital.com/`.

[Ohshima98] Ohshima, T., Satoh, K., Yamamoto, H., and Tamura, H. (1998). *AR$^2$Hockey: A Case Study of Collaborative Augmented Reality*. In *Proceedings of IEEE 1998 Virtual Reality Annual International Symposium (VRAIS '98)*.

[Parkinson96] Parkinson, B. W. and Spilker, J. J., editors (1996). *Global Positioning System: Theory and Practice, Volumes I and II*. American Institute of Aeronautics and Astronautics, Inc.

[Polhemus98] Polhemus Incorporated (1998). `http://www.polhemus.com/`.

[Press88] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition.

[Quan94] Quan, L. (1994). *Invariants of Six Points from Three Uncalibrated Images*. In *Proceedings of the 3$^{rd}$ European Conference on Computer Vision*, Pages 459–470.

[Raab79] Raab, F., Blood, E., Steiner, T., and Jones, H. (1979). *Magnetic Position and Orientation Tracking System*. *IEEE Transactions on Aerospace and Electronic Systems*, AES-15:709–718.

[Raskar98] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. (1998). *The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays*. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, Pages 179–188. ACM SIGGRAPH, Addison Wesley.

[Rheingold91] Rheingold, H. (1991). *Virtual Reality*. Summit Books.

[Rolland97] Rolland, J., Wright, D., and Kancherla, A. (1997). *Towards a Novel Augmented Reality Tool to Visualize Dynamic 3D Anatomy*. In *Proceedings of Medicine Meets Virtual Reality 5*.

[Rolland94] Rolland, J. P., Holloway, R. L., and Fuchs, H. (1994). *A Comparison of Optical and Video See-Through Head-Mounted Displays*. In *SPIE Telemanipulator and Telepresence Technologies*, Volume 2351. SPIE.

[Rose95] Rose, E., Breen, D., Ahlers, K. H., Crampton, C., Tuceryan, M., Whitaker, R., and Greer, D. (1995). *Annotating Real-World Objects using Augmented Reality*. In *Proceedings of Computer Graphics International '95*, Pages 357–370.

[Schmaltz32] Schmaltz, G. (1932). *A Method for Presenting the Profile Curves of Rough Surfaces*. *Naturwiss*, 18:315–316.

[Sherman98] Sherman, W. (1998). *Virtual Reality*. Morgan Kaufmann.

[Sims94] Sims, D. (1994). *New Realities in Aircraft Design and Manufacture*. *IEEE Computer Graphics and Applications*, 14(2):91.

[Soatto94a] Soatto, S., Frezza, R., and Perona, P. (1994a). *Recursive Motion Estimation on the Essential Manifold*. In *Proceedings of the* $3^{rd}$ *European Conference on Computer Vision*, Pages 61–72.

[Soatto94b] Soatto, S. and Perona, P. (1994b). *Recursive Estimation of Camera Motion from Uncalibrated Image Sequences*. In *Proceedings of the* $1^{st}$ *IEEE International Conference on Image Processing*, Pages III–58–62.

[Soatto93] Soatto, S., Perona, P., Frezza, R., and Picci, G. (1993). *Recursive Motion and Structure Estimation with Complete Error Characterization*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '93)*, Pages 428–433.

[Sowizral93] Sowrizal, H. A. and Barnes, J. C. (1993). *Tracking Position and Orientation in a Large Volume*. In *IEEE Virtual Reality Annual International Symposium*, Pages 132–139. IEEE.

[State94] State, A., Chen, D. T., Tector, C., Brandt, A., Chen, H., Ohbuchi, R., Bajura, M., and Fuchs, H. (1994). *Case Study: Observing a Volume-Rendered Fetus within a Pregnant Patient*. In *Proceedings of IEEE Visualization '94*, Pages 364–368.

[State96a] State, A., Hirota, G., Chen, D. T., Garrett, W. F., and Livingston, M. A. (1996a). *Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking*. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, Pages 429–438. ACM SIGGRAPH, Addison Wesley.

[State96b] State, A., Livingston, M. A., Hirota, G., Garrett, W. F., Whitton, M. C., and Fuchs, H. (1996b). *Technologies for Augmented-Reality Systems: Realizing Ultrasound-Guided Needle Biopsies*. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, Pages 439–446. ACM SIGGRAPH, Addison Wesley.

[Sutherland65] Sutherland, I. E. (1965). *The Ultimate Display*. In *Information Processing 1965, Proceedings of IFIP Congress 65*, Pages 506–508.

[Sutherland68] Sutherland, I. E. (1968). *A Head-mounted Three-dimensional Display*. In *Proceedings of the 1968 Fall Joint Computer Conference*, Volume 33, Pages 757–764. Thompson Book Co.

[Titterton97] Titterton, D. and Weston, J. (1997). *Strapdown Inertial Navigation Technology*. Peter Peregrinus, Ltd., London, United Kingdom.

[Tuceryan95] Tuceryan, M., Greer, D. S., Whitaker, R. T., Breen, D. E., Crampton, C., Rose, E., and Ahlers, K. H. (1995). *Calibration Requirements and Procedures for a Monitor-Based Augmented Reality System*. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273.

[Selspot98] Univ. of Iowa (1998). `http://www.uiowa.edu/˜vpr/research/units/selspot.htm`.

[Verplaetse95] Verplaetse, C. (1995). Can a pen remember what it has written using inertial navigation?: An evaluation of current accelerometer technology. Available from `http://verp.www.media.mit.edu/people/verp/projects/smartpen/`.

[Viéville90] Viéville, T. and Faugeras, O. (1990). *Feed-Forward Recovery of Motion and Structure from a Sequence of 2D-Line Matches*. In *International Conference on Computer Vision*, Pages 517–520.

[Viirre98] Viirre, E., Pryor, H., Nagata, S., and Furness III, T. A. (1998). *The Virtual Retinal Display: A New Technology for Virtual Reality and Augmented Vision in Medicine*. In *Medicine Meets Virtual Reality 6*, Pages 252–257. IOS Press.

[Wang90] Wang, L.-L. and Tsai, W.-H. (1990). *Computing Camera Parameters using Vanishing-Line Information from a Rectangular Parallelepiped*. *Machine Vision and Applications*, 3:129–141.

[Ward92] Ward, M., Azuma, R., Bennett, R., Gottschalk, S., and Fuchs, H. (1992). *A Demonstrated Optical Tracker with Scalable Work Area for Head-mounted Display Systems*. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, Volume 25, Pages 43–52.

[Weber98] Weber, H. (1998). Personal communication.

[Webster96] Webster, A., Feiner, S., MacIntyre, B., Massie, W., and Krueger, T. (1996). *Augmented Reality in Architectural Construction, Inspection, and Renovation*. In *Proceedings of the Third ASCE Congress for Computing in Civil Engineering*.

[Weintraub92] Weintraub, D. and Ensing, M. (1992). *Human Factors Issues in Head-up Display Design: the Book of HUD*.

[Welch95] Welch, G. and Bishop, G. (1995). *An Introduction to the Kalman Filter*. Technical Report TR95-041, Department of Computer Science, University of North Carolina at Chapel Hill.

[Welch97] Welch, G. and Bishop, G. (1997). *SCAAT: Incremental Tracking with Incomplete Information*. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, Pages 333–344. ACM SIGGRAPH, Addison Wesley.

[Welch96] Welch, G. F. (1996). *Single-constraint-at-a-time Tracking*. Ph.D. Dissertation, University of North Carolina at Chapel Hill.

[Whitaker95] Whitaker, R. T., Crampton, C., Breen, D. E., Tuceryan, M., and Rose, E. (1995). *Object Calibration for Augmented Reality*. In *Proceedings of Eurographics '95*, Pages C–15–C–27.

[Wloka95] Wloka, M. M. and Anderson, B. G. (1995). *Resolving Occlusion in Augmented Reality*. In *1995 Symposium on Interactive 3D Graphics*, Pages 5–12. ACM SIGGRAPH.