Merging Real and Virtual Environments with Video See-Through Head-Mounted Displays

by
Michael A. Bajura

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
1997

Approved by:

Advisor: Henry Fuchs

Reader: Gary Bishop

Reader: James Coggins

Turner Whitted

Jonathan Marshall

Abstract

Michael A. Bajura:

Merging Real and Virtual Environments

with

Video See-Through Head-Mounted Displays

(Under the direction of Henry Fuchs)

This dissertation explores the problem of inserting virtual (computer-generated) objects into natural scenes in video-based augmented-reality (AR) systems. The augmented-reality system problems addressed are the shorter-term goal of making synthetic objects appear to be more stable and registered and the longer-term goal of presenting proper occlusion and interaction cues between real and synthetic objects. These results can be achieved in video-based AR systems by directly processing video images of the natural scene in real-time. No additional hardware or additional tracking mechanisms are needed above those currently used in augmented-reality systems.

The implications of this work suggest a different approach to the design of AR tracking systems in which the user's world view is used as part of the tracking process. This work shows that tracking system errors which affect the user's world view can be corrected by measuring these errors dynamically and correcting for them in a way which is not readily apparent or objectionable to the user. This means that either cheaper tracking systems could be used or possibly even no separate tracking system in certain situations. Furthermore, the user's world view can be used to construct a 3-D model of his environment which can be used to model object interaction and occlusion relations. This is the first step toward further work of properly shading and lighting synthetic objects inserted into natural scenes.

A real-time (30 Hz) AR system is demonstrated which uses simple image feature tracking to improve registration and tracking accuracy. This is extended to a more general system which illustrates how reconstruction of natural scene geometry, correct synthetic and real object occlusion, and collision detection could be achieved in real-time. 3-D error analysis is included. A camera calibration appendix contains accuracy measurements.

Acknowledgements

Support for this research was provided by The Link Foundation, the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (NSF Cooperative Agreement ASC8920219), and DARPA contract DABT 63-93-C-0048 "Enabling Technologies and Application Demonstrations for Synthetic Environments."

Thanks to: Andrei State for Figure 3.1, Henry Fuchs for writing suggestions.

Special thanks to Gary Bishop for assistance and support above and beyond the call of duty.

And also my family, a blessing whose value I am only starting to appreciate.

Contents

List of Tables

List of Figures

List of Abbreviations

AR - Augmented Reality
CPU - Central Processing Unit
HMD - Head Mounted Display
LCD - Liquid Crystal Display
SE - Synthetic Environment
VE - Virtual Environment
VR - Virtual Reality

Chapter 1

Introduction

1.1 The Problem

One of the barriers to the acceptance and utility of *synthetic-environment* (SE) [1] systems has been their inability to function in a user's natural surroundings. Currently most SE systems are *virtual-environment* (VE) systems which immerse a user with images of entirely synthetic surroundings which can be unfamiliar, abstract, and lacking the detail of everyday experience. This often requires a user to learn and adapt to a new environment which may be unnecessary for many applications [2]. Another type of SE is an *augmented-reality* (AR) system where a user is presented with images of his natural surroundings which are *augmented* with computer-generated images. For example, these could be 3-D medical data sets of a patient which are anatomically registered [Bajura et al., 1992] [State et al., 1996b] or 3-D assembly directions for complex machinery [Feiner et al., 1993] [Feiner et al., 1992]. AR systems share the same basic technology of VE systems except that the user is able to "see through" his *head-mounted display* (HMD) into his natural environment with the use of either half-silvered mirrors (optical see-through), or video cameras (video see-through). It is interesting to note that Ivan Sutherland's [3] original synthetic-environment system was actually an AR, or see-through, system with which the user could view computer-generated images superimposed on views of his natural environment [Sutherland, 1968].

So why have SE system applications evolved toward a VE, or non see-through, design denying the user access to his natural environment? Aside from the actual desire to immerse a user in a

[1] The terms synthetic environment (SE), virtual environment (VE), and augmented reality (AR) are taken from *Virtual Reality: Scientific and Technological Challenges*. VE and AR systems are logically grouped into the broader classification of SE systems which project a user into an "interactive environment mediated by artificial electronic and electromechanical devices."[Durlach and Mavor, 1994, page 21]

[2] A example application is molecular docking taking place in an abstract molecular docking world versus a user's natural environment. It would be possible to immerse a user in a synthetic environment modeled after his own environment but this modeling task is not always practical.

[3] Ivan Sutherland is often credited with the original concept of a synthetic-environment system in his visionary paper *The Ultimate Display* [Sutherland, 1965].

completely synthetic experience, VE systems become dominant for three main reasons:

1. Given the same underlying technology, a AR system exposes errors in tracking, registration accuracy, and system delay which are not as apparent in a similar VE system,

2. AR systems require an environment model to simulate interactions between real and synthetic objects which is not needed for strictly VE systems, and

3. Immersive, wide-angle, head-mounted displays are easier to make without see-through capability.

Because of the difficulties in creating usable AR systems many of the SE applications developed to date have been designed as VE systems. Many of these applications could be improved by an AR system, for example one which allows interactions in a more natural setting, perhaps with a colleague [4] .

This dissertation addresses the first two problems listed above in the case of video see-through AR systems. In this type of AR system, video cameras mounted on a user's HMD view the user's natural surroundings. Images from these cameras are mixed electronically with synthetically generated images and displayed on the user's HMD to provide see-through capability. An alternative see-through strategy is to use half-silvered mirrors to optically superimpose(reflect) a synthetic image onto a user's natural view. The strength of the video see-through approach is the capability to completely control a user's images, both in time and content, in a way which is not possible with a more direct optical see-through system.

[4] An alternate approach is to retain the VE model and synthetically model a user's natural environment and include colleagues as synthetic actors. This approach lacks the richness and realism of a see-through system because of the modeling required.

1.2 Thesis Statement and Contribution

This dissertation explores techniques for improving video see-through AR systems by applying computer vision algorithms to the real-world video stream which is already present in a video see-through AR system. Processing this video stream makes it possible to:

1. Correct image registration errors caused by tracking system errors and system delays, and to
2. Sense the natural environment to perform synthetic and real object occlusions and interactions, a first step toward more complete interactions such as shadowing and physically-based modeling.

These two contributions support the thesis:

> A video see-through HMD user's visual comprehension can be enhanced by applying real-time computer vision techniques to the HMD user's see-through video stream to detect 3-D surfaces and objects in the user's physical surroundings. Such a system is feasible and practical to build and will provide improved perception for see-through virtual environment applications.

The contribution of this work is the combination of existing computer vision techniques with existing SE systems. Systems based on *both* of these disciplines (computer vision and SE systems) yield results which have not been achieved with existing methods. Two results are presented in this dissertation:

1. Apparent tracking accuracy can be improved by identifying precalibrated artificial fiducial points in the natural environment and using those points to correct for camera parameters such as position and orientation. This correction is necessary because errors exist in any camera and tracking system which cause misregistration of synthetic objects inserted into a real scene. This result is presented in Chapter 3. In Chapter 4 this result is extended to more natural and uncalibrated fiducial points.
2. Information about natural scenes can be gathered from the video see-through image stream and used for virtual and real object interaction to enhance visualization by providing 3-D interaction cues. This result is presented in Chapter 4 with an analysis of 3-D reconstruction error. Camera calibration error analysis is presented in appendix A.

This dissertation does not purport to create or identify the best or most novel computer vision techniques for these tasks. The goal is merely to show the existence of techniques which can improve video see-through AR system design. The appeal of this general approach is that it doesn't require adding hardware to existing video-based AR systems, merely processing video data which are already available.

1.3 Organization of the Dissertation

- Chapter 2 presents current models for SE (VR and AR) systems and proposes an improved AR system model. Weaknesses of current system designs are explained with motivation for system designs based on both image generation and image analysis. Other results in AR systems are covered, showing how each relates to these models.

  The second part of chapter 2 reviews results which are applicable to recovering natural scene information from a user's head-mounted display. Although these results come mostly from the areas of computer vision, image processing, and robotics, several researchers have also been pursuing similar results from computer graphics and VR backgrounds.

- Chapter 3 explains a method which can correct image registration, calibration, and tracking errors by using precalibrated fiducial points in the user's natural environment. The locations of these points in the user's field of view are used to correct various errors in an AR system [Bajura and Neumann, 1995a] [Bajura and Neumann, 1995b]. Applications of this technique have yielded encouraging results in the 3-D ultrasound visualization research at the University of North Carolina [State et al., 1996a] [State et al., 1996b].

- Chapter 4 describes a method for detecting surface geometry in the user's natural environment to generate proper occlusion cues and to perform collision detection. This more difficult than correcting image registration (as in Chapter 3) because it requires constructing a model of scene geometry. This model is used to simulate real and virtual object interaction and to correct for tracking and system calibration errors. This correction method is similar to that presented in chapter 3 except that the fiducial points are dynamically selected and located without precalibration.

- Chapter 5 contains conclusions and directions for future work in this area.

- Appendix A covers camera calibration which is used in Chapter 4.

Chapter 2

Background

The idea of AR as a dynamic immersive mode of interaction is usually attributed to Ivan Sutherland from his visionary paper *The Ultimate Display* [Sutherland, 1965]:

> The ultimate display would, of course, be a room within which the computer can control the existence of matter.

Since then much progress has been made, in both small and large steps, toward this goal. The ability to synthesize extremely complex and detailed images at interactive rates is now commonplace. Head-mounted displays and systems for tracking head and body positions are commercial products. Force feedback systems are an active area of research [Adachi et al., 1995] [Buttolo and Hannaford, 1995] and are already viable commercial products [PHANToM, 1997]. It seems that the ultimate display problem is well on its way to being solved... Or is it?

Many of the original problems present in Sutherland's early system [Sutherland, 1968] remain. The delay between a user's head motion and corresponding image display remains largely unchanged. Accurate registration between computer-generated imagery and real imagery is still elusive.

Current VR systems mask the shortcomings of tracking and image-generation systems by presenting the user with a completely synthetic environment instead of a hybrid of real and virtual environments. In a completely synthetic environment, tracking errors cause simple distortions, and image-generation delays cause image update delays, both of which users have been able to accommodate with some patience. AR systems based on this same technology do not function well because these same shortcomings cause visible registration errors between real and computer-generated objects — often destroying the usefulness of an application such as image-guided surgery or the illusion of a fused reality.

This chapter starts with a taxonomy of SE systems, showing how a VE system is typically extended to become a "see-through" AR system. The weaknesses of this design are pointed out followed by a brief review of how each system component contributes to these problems. An improved

AR system model is presented which incorporates computer vision techniques. A summary of related AR research shows how other systems relate to the current and improved system model.

The second part of this chapter reviews related work in environment sensing and range-finding. Most of these results are from the computer vision, image processing, and robotics communities although some more recent work is also from a computer graphics and VR perspective.

2.1 Current VE and AR System Models

Current VE System Model

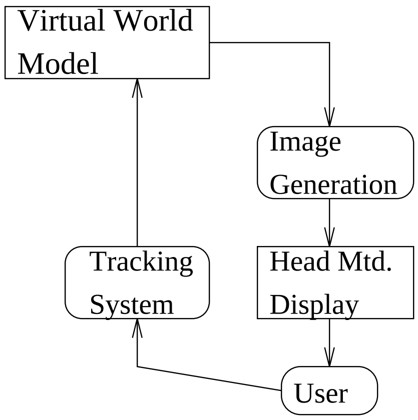

Figure 2.1: Current VE System Model

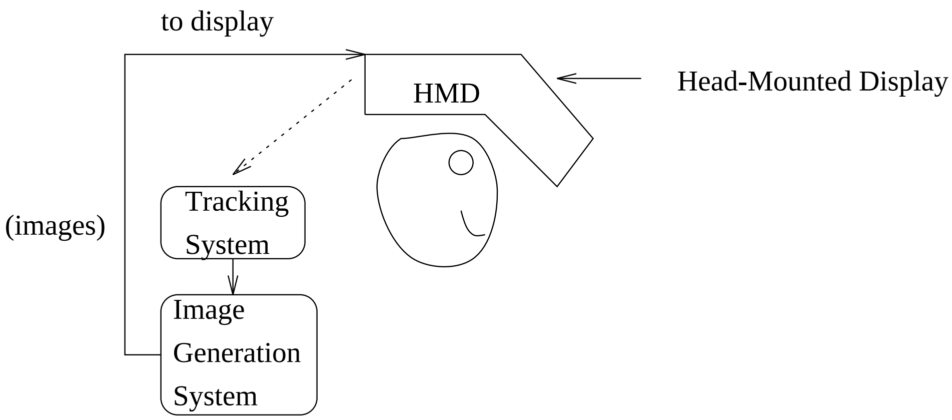

Figure 2.2: Example of Current VE System

Figure 2.1 is a model of a typical VE system and figure 2.2 is an example setup. The user views images of a virtual-world model which are rendered by an image-generation system. The user wears a *head-mounted display* (HMD) whose position is sensed by a tracking system. The tracking system's reports are used to position a virtual camera in a virtual-world model for image generation. The user is immersed in the virtual-world model by viewing images of it which are updated in response to his head movement. The tracking and image-generation systems form a user-controlled feedback loop.

Current AR System Model

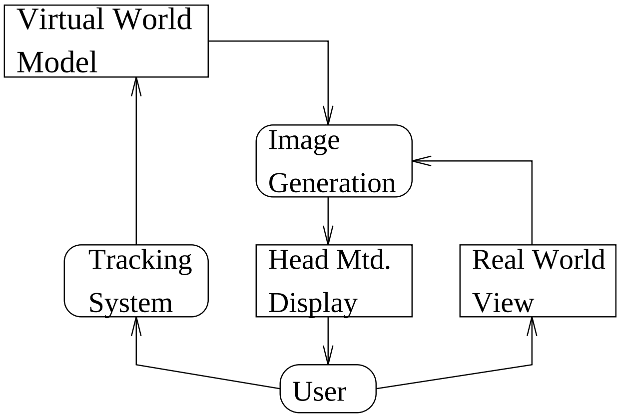

Figure 2.3: Current AR System Model

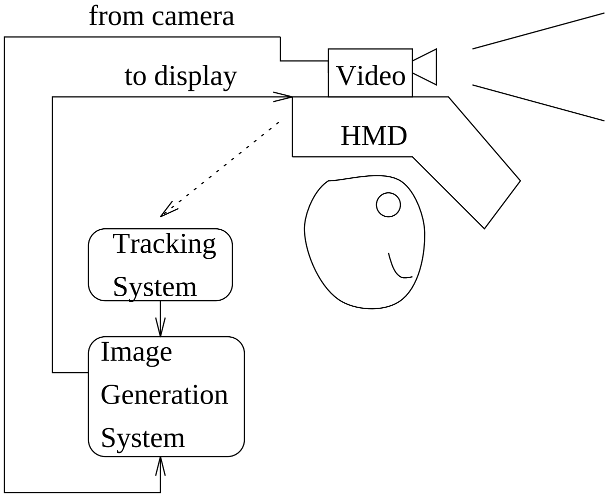

Figure 2.4: Example of Current Video-Based AR System

Figure 2.3 outlines how the VE system model is currently converted into an AR system and figure 2.4 shows an example of a video-based setup. An additional feedback loop from the user directly into the image rendering process adds see-through capability to the user's HMD. In a video-based system, the user supplies a real-world view from a video camera mounted on his head. Images from this camera are composited with images generated by a virtual camera which models the real camera's position, orientation, and distortion characteristics. In an optical system, a half-silvered mirror mounted in the user's line of sight superimposes a reflected synthetic image on the user's direct view of his surroundings. In a stereo system, separate cameras and feedback paths or mirrors are used for each eye.

2.1.1 Weaknesses of Current System Models

The weaknesses in these models are due to both delay *and* inaccuracies in the feedback paths and are more pronounced in AR (see-through) systems than in VE (non see-through) systems.

1. Feedback path delay:

    - In VE systems: The user's perception of the virtual environment is delayed because of update delays in response to his head movement (system lag). The inconsistent sensory cues created by this time delay can cause disorientation or even nausea if they are excessive and can limit the effectiveness of an application [Pausch et al., 1992]. Pausch [Pausch et al., 1992] cites unpublished work by Lilienthal claiming that update delays should be limited to less than 35 msec because "the motion cues may give the impression of motion in one direction while the delayed visual cues give the impression of motion in another direction."

    - In AR systems: In addition to system lag, computer-generated objects can appear to be misregistered with respect to the natural environment during user movement. This occurs when the delay in the tracking/image-generation loop is different from the delay in the real-world view/image-generation loop [Azuma and Bishop, 1994] (note that there are *two* paths back to the user(figure 2.3)). For example, a user with a 45 degree field-of-view rotating his head at 90 degrees/second with a delay of 100 msec sees an error of 9 degrees or 20% of his field-of-view. This problem is not present in VE systems which have only one feedback loop. Video-based AR systems, like VE systems, also suffer from adding basic delay to the user's perception of the natural environment.

2. Feedback path inaccuracies:

    - In VE systems: Tracking inaccuracies and distortions in image generation and display can give the user a distorted perception of the virtual environment. In the author's experience, this often is not a problem because the tasks being performed are designed to be relative instead of absolute. The user naturally exaggerates or diminishes his motions to compensate for errors. Conversely, tasks and interaction methods requiring absolute precision have been largely avoided because of the increased accuracy requirements.

- In AR systems: Computer-generated objects appear to be statically misregistered with respect to the natural environment because the real-world view isn't modeled correctly by the tracking system and/or virtual camera model. This means the simulated camera for rendering synthetic objects may not be in the same position and orientation as the real camera or may not match the real camera's distortion parameters. When images from the real camera and virtual camera are combined, the camera registration error becomes apparent. This is the chief problem with designing and building AR systems with the current AR system model. As demonstrated by Holloway [Holloway, 1995], any errors in tracking, camera, or system calibration are amplified through an AR system creating image registration errors between the synthetic world model and actual world view. No matter how accurate the synthetic world model is, registration errors will be caused by a failure to align the virtual camera view with the real camera view.

In the current AR system model there is no way to handle virtual object and real object interactions such as collision detection and important visual cues such as obscuration and shading. This can be done for static environments by digitizing the natural environment and building a virtual-world model of the real-world environment before starting an AR application. However, the model of the real world still suffers from the same registration problems as do other computer-generated objects. In addition, attempting to copy the real world into the virtual world is not a practical or long-term solution for dynamic environments.

What is needed is a framework for AR systems which can dynamically model a natural scene while detecting and correcting for errors in calibration, tracking, and feedback path delay. A model which addresses these needs is presented in section 2.2. This dissertation addresses the correction of registration errors in Chapter 3 and demonstrates a method for sensing surfaces to present occlusion cues and to simulate object interactions in Chapter 4.

2.1.2 Tracking Systems

A tracking system is used to monitor the position and viewing direction of the user's head [1] so that an image-generation system can render synthetic images appropriate for the user's current pose. A few technologies which have been employed for this are: mechanical linkages [Faro, 1997], magnetic sensors [Polhemus, 1997] [Ascension, 1997], optical sensors [Ward et al., 1992], and acoustic methods [SAC, 1994]. Bhatnagar [Bhatnagar, 1993] provides a survey of these methods.

Magnetic tracking systems are attractive because no line-of-sight or mechanical tether is needed. Although the theoretical tracking accuracy of these systems is quite good, in practice their accuracy

[1] Usually the user's Head-Mounted Display is tracked, not the actual pose of the user's head

has been disappointingly low in magnetically distorted lab environments [Livingston and State,]. Mechanical arms and optical sensors have achieved better accuracy than magnetic systems at the expense of mechanical or line-of-sight restrictions.

The important operating characteristics of a tracking system are accuracy, resolution, update rate, latency (or lag), and working volume. For purposes of providing an AR environment, accuracy and resolution affect the computed position of the user's real-world view of his natural environment. Errors in the computed position cause misregistered images in the real and virtual views during static viewing. This happens because the model of the virtual camera doesn't match the position and viewing direction of the real camera exactly. Update rate and latency come into play during movement. A slow update rate or significant latency can cause virtual objects inserted into a natural scene to appear to swim about during user movement. Some success at effectively reducing the latency of tracking systems has been achieved with predictive tracking schemes using optimal filters and accelerometers [Azuma and Bishop, 1994] [Azuma and Bishop, 1995]. The drawback with this approach is that tracking error grows rapidly with larger prediction intervals and input signal frequencies. This means that predictions about the very near future can be made reliably but that longer-term predictions (which are of more interest) still have large amounts of error.

2.1.3 Image-Generation Systems

Image-generation systems transform models of objects or models of complete environments into images. Recently progress has been made toward rendering more complex images at increasing rates [Molnar et al., 1992] [Akeley, 1993]. Unfortunately the emphasis in these systems has been on total system throughput, often at the expense of image-generation latency and flexibility of camera models.

Because of the tradeoff of latency for image complexity and because of the delay incurred by rendering images as discrete frames [Mine and Bishop, 1993], the delay in image generation for synthetic-reality systems has remained largely unchanged or has even grown worse since Sutherland's pioneering work [Sutherland, 1968] [2] . This fact has been widely tolerated because the improvements in image complexity and rendering models have been so dramatic and possibly also because there has not been sufficient (economic) incentive to focus on low-latency systems. However, we are near the point where greater improvements in VR and AR experiences will be made by reducing image-generation latency rather than by increasing image complexity. Regan and Pose [Regan and Pose, 1994] describe a system for reprojecting images immediately prior to display on a HMD which significantly shortens head rotation latency. This falls under the research area of image-based rendering which directly generates reprojections of a scene from existing projections

[2]Sutherland's system could transform and display up to 3000 line segments at 30 frames per second while the Pixel-Planes 5 system at UNC [Fuchs et al., 1989] requires 54 ms to render a single triangle [Mine and Bishop, 1993]

[Chen, 1995] [McMillan and Bishop, 1995] [Levoy and Hanrahan, 1996] [Gortler et al., 1996].

Most high-performance image-generation systems have been optimized to render distortion-free images with an ideal pinhole camera model. This can be a problem for AR applications when trying to match rendered images with the distortion properties of a real lens system. To match image distortion in both the real and computer-generated image feedback paths(figure 2.3), either the image-generation system must generate distorted images or the real-world view must be distorted to match the pinhole camera model of the image-generation system. Real-time distortion correction has been demonstrated on commercial hardware [Watson and Hodges, 1995] by texture-mapping rendered images onto a warped polygon mesh and rendering a second image of that mesh. Unfortunately this solution adds a cycle of delay to the rendering process. The key in the texture-mapping solution is selecting a high enough order interpolation scheme to avoid obvious image artifacts, particularly in areas with significant image stretching.

2.1.4 Image Mixing

Two techniques have been used to mix a real-world view with a synthetic view to present an augmented reality. One is to use a half-silvered mirror (optical see-through), and the other is to capture the real-world view with a video camera and to generate composite images electronically (video see-through). There are advantages and disadvantages to each technique.

The half-silvered mirror technique has the advantage of letting the user see his natural surroundings directly without delay or loss of image quality, which could be important in certain applications. There can be problems in combining the dynamic range of natural scenes with the dynamic range of typical image displays, but theoretically this is not a significant drawback with the appropriate selection of optical filters and/or light shutters. An important issue is showing occlusion between dark virtual objects and bright real objects. A half-silvered mirror cannot mask a bright object although some form of addressable light shutter (Liquid Crystal Display) might help. Another drawback in optical scene merging is keeping up with reality — the user sees images of his natural surroundings at the speed of light while synthetic images are delayed by total system delay. Some success has been achieved in effectively speeding up synthetic images by using predictive tracking schemes [Azuma and Bishop, 1994] [Azuma and Bishop, 1995], but again the tracking error increases as position estimates are projected further into the future.

The advantage of the video see-through method is complete control over the composite images seen by the user. The real-world images can be delayed if needed and synthetic images can be rendered after spatial corrections to match the real-world view. The price of this complete control is the delay in processing, generating, and mixing the real-world and synthetic images. It should be noted that the delay in a video see-through system is not fundamentally different from the delay currently tolerated in a fully synthetic VR system. There is a degradation of the user's view of his natural scene because of technological limitations of video cameras and HMD display systems which may be unacceptable for certain applications. At present, the sampling resolution is artificially limited by the current video standards (NTSC, RS170) used by most real-time video cameras. Video see-through with electronic image mixing is the image mixing method studied in this dissertation.

2.1.5 Head-Mounted Displays for Augmented-Reality

The parts of a head-mounted display for augmented-reality are:

1. The image displays (one for each eye):

   Several different image display technologies provide varying resolutions, with Liquid Crystal Displays (LCDs) and field-sequential color displays being the most popular. Field-sequential displays achieve significantly greater resolution over LCDs by time-multiplexing color signals instead of simultaneously displaying separate color elements. Recent effort has been made to produce tessellated displays which combine several smaller displays into a larger one creating a larger field-of-view with increased resolution [Kaiser, 1997].

2. The optical systems for viewing the displays:

   Optical systems are usually independent for each image display for each eye. Each optical system places the virtual image of its display at a comfortable accommodation distance from the user's eye, allowing the user to focus on a point a few feet in front of him instead of a few inches. An important perceptual problem is that objects in the field-of-view are all seen at the same accommodation distance from the user's eye (visible in the display), which is not true of natural vision [Robinett and Rolland, 1991]. Important optical design issues include field-of-view, angular resolution, and distortion. Robinett [Robinett and Rolland, 1991] and Edwards [Edwards et al., 1993] demonstrate it is also important to match a user's convergence and inter-pupillary distance.

   In an optical see-through system, half-silvered mirrors are integrated with the optics to allow the user to directly view his natural surroundings while viewing the reflection of an image display.

   In a video see-through system, video cameras are used to obtain images of the user's surroundings. The total optical distortion observed by the user is a composition of the optical distortion in the video camera and the optical distortion in the image display. This dissertation addresses the problem of correcting for optical distortion in the video camera only and generating correctly registered images which account for this distortion. Designing the appropriate optical systems for distortion-free viewing with correct stereopsis on a HMD and/or finding the appropriate pre-distortion functions to achieve this are left as separate problems [Robinett and Rolland, 1991][Edwards et al., 1993].

3. A mounting fixture for the head:

   Helmets are frequently used as mounting fixtures. There must be space for image displays and associated optics, optional video cameras, and a tracking system if needed. Some HMDs are compact and light enough to be mounted on eyeglass-style frames [GeneralReality, 1997]. Unfortunately these frames often lack the rigidity needed for precise tracking and modeling.

2.1.6 Modeling the Camera

The two requirements for modeling a real camera attached to a tracking system are:

1. Characterization of the camera:

   Characterizing a camera means defining a camera model which provides a mapping from the images seen by the camera to ones seen by an idealized pinhole camera in the same position as the real camera. Equivalently this means mapping points in the camera's image plane to unique rays in space. Ideally this relationship could be represented by a perspective projection, an approximation which is sometimes adequate for certain cameras in certain applications. Generally non-perspective distortion must be characterized because all lens systems exhibit some amount of it. This distortion is not necessarily symmetric or centered on a camera's image plane although this simplification is often adequate.

   Parameters which characterize a camera can be classified as either external or internal camera parameters:

   - External camera parameters are the position and orientation of the camera relative to a coordinate system origin or tracking system origin.
   - Internal camera parameters describe aspects of image formation which are independent of a camera's external parameters. Field-of-view and focal length are examples of internal camera parameters for an idealized camera. However their meanings are somewhat ill-defined in a general setting.

2. Characterization of the tracking system itself:

   Characterization and calibration of tracking systems is beyond the scope of this work. This dissertation assumes the tracking systems used are independently characterized and calibrated.

The composition of a camera's external and internal parameters and readings from a tracking system make it possible to match points in the camera's image plane with specific rays in 3D world-space as the camera moves.

An effective interactive method for camera modeling is presented in chapter 3. Appendix A proposes an alternative scheme which directly and separately measures internal and external camera parameters. Discerning the interrelation between internal and external camera parameters is a frequent problem in other calibration methods.

2.2 New AR System Model

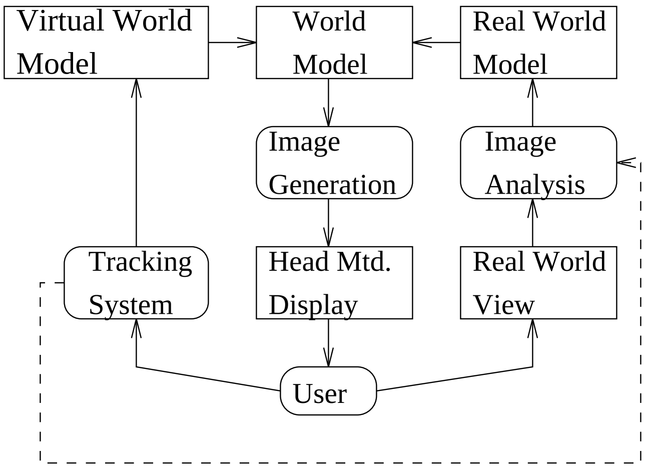

Figure 2.5: New AR System Model

Figure 2.5 illustrates a new proposed model for video see-through AR systems in which computer vision techniques are used to build a model of the real world. This makes it possible to correct for errors in tracking, calibration, registration, and delay on a frame-by-frame basis and to more accurately insert virtual objects into a real scene. The tracking system is an input into the process of building the real-world model as well as an input into the virtual-world model.

The central idea in the proposed system model is to combine the original virtual-world model and a detected real-world model into a combined world model. During the merging process, differences in the real-world and virtual-world models can be rectified to reduce image registration artifacts caused by errors in tracking, calibration, registration and delay. In contrast to the current AR system model where tracking, calibration, and delay errors directly affect image registration, these errors affect the quality of the real-world model. How these errors affect the final AR images depends on a) the sophistication of the real-world model, b) what approximations are made when combining the real world and virtual-world models, and c) the resolving power of the image analysis algorithm. The goal when merging the models is to minimize the registration error which is visible in the resulting 2-D images, not the 3-D registration error between the two models — in this way effectively hiding many of the underlying 3-D registration errors.

Because images are produced from a combined world model, the image-generation processes is relatively straightforward, requiring only slight variations on standard computer graphics rendering techniques. Simulation of other interactions besides viewing between real and virtual objects can also be handled by standard simulation techniques.

The problem of unequal feedback path delay between the real-world view and virtual-world model is addressed by delaying the unified world model to equalize the feedback path timings. This does not address the fundamental problem of delay in response to user movement; however this resulting delay need not be any longer than delays which are currently present in conventional VE systems. It is assumed and experience suggests that presenting correctly registered and rendered composite images with some delay presents a more convincing reality than one achievable with less delay in the real-world scene and with incorrect registration, as in optical see-through systems.

2.3 Other AR Systems

Augmented-reality systems based on the current model (figure 2.3) have been used to visualize ultrasound medical data within a patient [3] [Bajura et al., 1992], to provide 3D graphical annotations for machinery [4] [Feiner et al., 1993] [Feiner et al., 1992], and to assist manual manufacturing [5] [Caudell and Mizell, 1992]. These systems suffer from the weaknesses outlined in section 2.1.1.

Mellor [Mellor, 1995a] [Mellor, 1995b] reports using known fiducial marks in an augmented-reality scene to register anatomical data relative to those marks. The effective position of the camera is computed dynamically to be consistent with the observed positions of fiducial marks. This method works well for displaying registered data when the fiducial marks are visible and positioned nearby the registered data. Uenohara [Uenohara and Kanade, 1995] reports similar results using a template matching scheme to locate features on known objects. Ravela [Ravela et al., 1996] tracks known features on a known object through different poses of that object, effectively repositioning the camera at every step. These approaches are similar to the new AR model in figure 2.5 but without a tracking system.

Kanade [Kanade, 1996b] and Wloka [Wloka and Anderson, 1995] report using real-time and near real-time depth-from-stereo, respectively, to generate depth maps relative to a viewing position. These depth maps are then used to combine synthetic images with real ones from video cameras. Kanade [Kanade, 1996b] calls this operation z-keying, after the z-buffering computer graphics rendering algorithm and the chroma-keying method used in video production. In Kanade's system the camera position is not tracked, which means the camera cannot move in the real scene because synthetic 3-D geometry remains fixed in the camera's local coordinate system. Wolka uses a mechanical tether to track camera position. In both of these systems the real-world model (depth map)

[3] Application: A physician can see live scanned output from an ultrasound transducer apparently emanating from the transducer itself instead of from a conventional side display.

[4] Application: 3D annotations and internal structure are projected onto a user's view of a laser printer to assist maintenance tasks.

[5] Application: 3D annotation of aircraft assemblies for manual manufacturing tasks.

relative to camera position is recomputed for every combined image. No world model is maintained between image updates to either reposition the camera (Wolka) or to construct a more accurate or larger world model. The accuracy of both of these approaches is limited because correspondence is computed only to single pixel accuracy. This imposes a measurement discretization equal to the number of pixels disparity throughout the working volume.

The MIT Alive project [MediaLab, 1996] generates images of computer-generated artificial life interacting with people in a natural setting. A ground-plane constraint tracks the position of people moving in the environment and serves to position them relative to computer-generated life forms on a virtual plane.

An emerging technique for rendering images from recorded images of 3-D environments called *image-based rendering*, pioneered by Chen [Chen, 1995], McMillan [McMillan and Bishop, 1995], Levoy [Levoy and Hanrahan, 1996], and Gortler [Gortler et al., 1996] holds some promise for augmented-reality applications. However the weaknesses of not collecting images interactively and not generating models of reprojected scenes (so that computer-generated objects may be inserted properly) remain to be addressed.

2.4 Sensing the Environment

Many methods for sensing depth in the environment are presented in literature. The purpose here is to identify methods which are applicable to video-see through head-mounted displays and not to provide a complete summary of all computer vision and range-finding research.

Broadly speaking, methods for sensing depth remotely are either active or passive. Active strategies use some form of structured light to illuminate a scene which is scanned by a sensor. Examples of active methods are light stripe (triangulation) [Cyberware, 1997] [Gruss et al., 1992], moire (interference), and radar (time-of-flight). Besl [Besl, 1989] provides a good overview of active sensing strategies. These methods are unattractive to AR systems because they would involve adding light to the natural scene which may be objectionable and adding hardware which may not be needed.

Examples of passive range-finding methods include stereo, depth-from-focus/defocus, and depth-from-image sequences (structure-from-motion). Because these methods do not use structured light to illuminate an area, the area being scanned must already contain sufficient visual texture, or variation in brightness, for analysis. There is a grey area between active and passive methods where a visual texture is projected onto a scene simply to increase its visual texture content but not to measure correspondence.

Finding depth from image sequences appears to be a promising depth recovery approach for HMD applications because it is incremental, tractable, and can be logically extended to build larger world models. This approach is studied in this work, although other methods have been demonstrated. The following subsections detail the arguments against depth-from-stereo and depth-from-focus/defocus and argure for finding depth from image sequences. Depth-from-stereo and depth-from-focus could still be integrated into a larger depth-from-image sequence scheme.

2.4.1 Depth-from-Stereo

Even though a head-mounted display could easily provide two views of the real world from which depth could be computed by stereo correspondence, this method is surprisingly unattractive for two reasons:

1. Stereo requires finding a correspondence of features in at least two separate images:

   In arbitrarily complex scenes or in cases where the images are greatly different (cameras far apart) this is known to be a difficult and potentially intractable problem. When the cameras are close together, finding correspondence becomes tractable except that the baseline for triangulation is relatively small and may not yield accurate results.

2. Integration of the stereo measurements into a larger world model would still be needed as the user pans around the environment:

   Even with a perfect stereo system, a method would be needed to optimally combine each stereo pair into a larger real-world model to improve model accuracy, facilitate real and virtual-object simulation, and to adjust camera pose. A stereo-only algorithm is insufficient.

2.4.2 Depth-from-Focus/Defocus

Depth-from-focus or defocus is measured by estimating the amount of blur in different parts of an image. Because of depth-of-field effects, features further from the focal plane of an imaging system are blurrier than features closer to the focal plane. By relating the amount of blur(defocus) of a particular feature to its depth, a range map can be computed [Ens and Lawrence, 1993][Nayar et al., 1995]. This can be done in parallel for each point in an image. Nayar [Nayar et al., 1995] reports generating depth estimates at 512×480 resolution at 30 Hz with an accuracy to 0.3%. Another "pixel-by-pixel" approach is to adjust the imaging system sequentially for each point in an image until it is in focus. The setting of the lens system needed to focus each point is used to estimate that point's distance from the camera [Siegel and Leary, 1992]. Unfortunately, this pixel-by-pixel approach is impractical because of its sequential nature.

The best characteristic of the depth-from-focus/defocus approach is that only one camera or optical axis is used, avoiding both correspondence and occlusion problems. The chief drawback of the depth-from-focus/defocus approach is that the effective baseline of the imaging system is no larger than the aperture of the camera being used, effectively limiting the accuracy and working volume of this method. In a demonstration system, Ens [Ens and Lawrence, 1993] reports 10% RMS accuracy over a range of 80 to 95 cm from the camera, or 1.7% RMS accuracy relative to the camera. Nayar reports 0.3% RMS accuracy over an approximately 30 cm working range. This is not enough working volume for a usable AR application. Like the stereo only approach, this approach would also require additional measurement integration to synthesize a complete world model.

2.4.3 Depth-from-Image Sequences

In this approach recognizable features are tracked in an image sequence to incrementally construct a scene model. Finding the correspondence of features in image sequences is tractable because changes between images are incremental and bounded, making a parallel and/or real-time correspondence search possible.

This method is attractive for use in AR systems because a user naturally pans around his environment over time. Views from different positions can be used to build an effectively larger baseline of correspondences than would be possible with a simple stereo system, yielding a theoretically better solution than is possible with stereo. The drawback is that the user *must* move around for this method to converge.

Setting aside for a moment the definition of what a feature is, there are three cases for finding the depth of features from an image sequence depending on what is known and unknown. Assuming a calibrated camera model and that feature correspondences have been made through a sequence of images(described in section 2.4.4), the three cases are:

1. The motion of the camera is known and feature positions are unknown:

   Feature positions relative to the camera can be found directly by least squares methods. Knowing the position of the camera at different times constrains the correspondences of features in different images to lie along lines of known geometry, or epipolar lines, in each image. This can speed up the search for correspondences. Stereo can be thought of as a special case of this method with only two images.

   Two examples of finding depth from several images are described by Kanade [Kanade et al., 1992] and Bolles [Bolles et al., 1987]. By constraining camera motion to be linear, perpendicular to viewing direction, and parallel to image scan lines, image correspondences are more easily located in successive images on identical image scan lines. Feature redundancy in multiple images increases accuracy of the solution. These formulations are not suitable to head-mounted display applications because of the requirement for linear camera motion perpendicular to camera viewing direction.

   Shekhar [Shekhar and Chellappa, 1992] uses a partially-tracked camera and an extended Kalman filter to find feature positions and improve camera motion parameters. This method is similar to a tracked head-mounted display except that the camera position is not as well known and is estimated by a filtering process.

Skifstad [Skifstad, 1991] demonstrates a mobile robot which estimates depth in its environment by using its own measured motion to position successive camera images and to restrict the search for object features to specific image areas.

2. The feature positions are known and camera motion is unknown:

   The motion of the camera in each frame can be found directly by least squares methods, assuming there are a sufficient number of points visible and that they do not lie on a degenerate surface. This case describes systems by Mellor [Mellor, 1995b], Uenohara [Uenohara and Kanade, 1995], Ravela [Ravela et al., 1996], and State [State et al., 1996a] which identify known fiducial marks to reposition a virtual camera.

3. Both feature positions and camera positions are unknown:

   This defines the classic non-linear structure-from-motion problem. Because both camera and feature positions are unknown, shape and motion recovery are possible only up to a scale factor. Three approaches to solving this problem are:

   (i) 2 frame methods:

   A linear 3×3 (9 element) matrix matrix relating the relative positions of two frames can be defined by at least 8 features in each image which do not lie on a degenerate surface, or Maybank Quadric. Longuet-Higgins [Longuet-Higgins, 1981] [Longuet-Higgins, 1984] and Tsai [Tsai and Huang, 1984] describe non-linear methods for recovering the relative orientation between two frames from this "essential matrix." Once relative orientation is known, the structure of the features can be computed. The chief drawback to this method is noise sensitivity to input data. This two frame formulation has been used with some success as a starting point for other batch and incremental methods.

   (ii) Batch methods:

   Non-linear minimization is used on all data from all frames of a sequence to find an optimal solution. Because all data are considered simultaneously, this method is much less sensitive to noise than the two frame method. This method achieves the best answer in situations where all data are available simultaneously [Weng et al., 1993]. Getting a good initial guess can be important to speed up minimization or ensure that the correct solution is found. Examples of this method are by Szeliski [Szeliski and Kang, 1993a] [Szeliski and Kang, 1993b], and Weng [Weng et al., 1993].

   Tomasi [Tomasi and Kanade, 1992] [Tomasi and Kanade, 1991a] [Tomasi and Kanade, 1991b] describes a linear solution for factorizing shape and motion under the assumption of orthographic projection. Shape and motion are decomposed into a product of shape and rotation matrices which is found by singular value decomposition analysis. This method achieves impressive results and is extended to a perspective approximation which preserves this factorization property [Poelman and Kanade, 1992]. This approximation is good when the range of depth estimation remains small compared to the distance between the camera and range of estimation even if the range of estimation undergoes significant changes in depth over an image sequence. These shape and motion solutions are not well suited to AR applications which typically require estimating distance at arm's length over a similar range, violating both the assumption of orthographic projection and the range of good approximation for the perspective approximation to orthographic projection.

Hildebrand [Hildebrand and Köthe, 1993] describes a method which combines multiple two-frame solutions from a group of frames for an initial estimate of camera position and object structure which is improved by iteration. Unfortunately, the selection of frames and correspondence of regions in those frames is not automatic.

(iii) Incremental methods:

These methods converge on a structure (and motion) model by processing feature position data frame by frame. An initial guess for structure is updated by each successive frame to yield a more accurate model and motion estimate. Like the batch method, the initial guess and parameterization selection are important for converging on the correct solution. Optimal (Kalman) filtering is typically used for this process. The advantage of an incremental approach is that a partial solution is available at intermediate steps.

Several authors [Matthies et al., 1989] [Weng et al., 1993] [Broida and Chellappa, 1986] [Thomas and Oliensis, 1992] [Azarbayejani and Pentland, 1994] [Bouguet and Perona, 1995] [Morita and Kanade, 1994] report varying results using Kalman filters. Azarbayejani [Azarbayejani and Pentland, 1994] claims a unique parameterization including camera focal length improves solution accuracy and speed. Morita [Morita and Kanade, 1994] presents a sequential method based on the batch method reported by Tomasi [Tomasi and Kanade, 1992] [Tomasi and Kanade, 1991a] [Tomasi and Kanade, 1991b] which achieves nearly the same results as Tomasi but retains the same limitations. Bouguet [Bouguet and Perona, 1995] achieves impressive results for recovering the structure of a hallway and relative camera position for an 8000 frame round-trip image sequence.

2.4.4 Locating and Tracking Features in Image Sequences

To recover structure from motion in an image sequence, Tomasi and Shi [Tomasi and Kanade, 1991b] [Shi and Tomasi, 1994] define a feature simply as a point which can be tracked reliably from one image to another. Tomasi [Tomasi and Kanade, 1991b] demonstrates that trackable features can be recognized by local statistics based on squared image gradient values by implementing a tracking algorithm based on Lucas' [Lucas and Kanade, 1981] method of registering windows by shifting them relative to one another to maximize a correlation metric. Lucas suggests that parameters for rotation, scale, and shearing could also be used to register feature windows. Shi [Shi and Tomasi, 1994] addresses this idea by estimating an *affine motion field* for each feature. When a feature's correlation with its original view (after correction for motion and distortion) falls below a given threshold, Shi and Tomasi consider correspondence to be lost.

Lee [Lee and Cooper, 1993] describes a structure from motion algorithm in which larger windows in an image sequence are registered by affine transformations between a window's original viewing and the current image. The transformations of the individual windows instead of points are then used to recover structure. The claim is that computing the affine transformation of a larger region is more stable than trying to locate individual features within that region. The drawback with this approach is that it is only possible to compute the affine registration in windows with sufficient visual texture — probably the equivalent of locating enough individual features by Tomasi's method to define a local affine transformation from individual features.

Forstner [Forstner, 1994] presents a scale-space approach to feature extraction based on local statistics of the image function. The criteria used are based on the same squared gradient information used by Tomasi [Tomasi and Kanade, 1991b] to select features. In addition to the trackable features as defined by Tomasi, Forstner identifies lines and edges in static figures.

Real-time tracking of simple features in video images has been demonstrated in several applications. A few recent examples are listed here. Bajura [Bajura and Neumann, 1995a] demonstrates real-time tracking of LEDs by threshold detection. Blake [Blake and Isard, 1994] demonstrates incremental real-time tracking of curved silhouettes suitable for recognizing hand and lip movement. The ultrasound visualization project at the University of North Carolina has achieved good results in incrementally tracking color dot targets [State et al., 1996a] under a variety of lighting conditions. It appears that many of the various non real-time incremental tracking algorithms are limited only by processing speed from being real-time algorithms. The simple real-time tracking algorithms demonstrated to date have become possible only because of increases in processing power. Further advances in processing power will soon make more general real-time tracking algorithms possible.

Chapter 3

Correcting Registration Error

This chapter is a reformatted copy of a paper which appeared in the proceedings of the Virtual Reality Annual International Symposium (VRAIS) 1995 [Bajura and Neumann, 1995a]. It describes an experiment in which known features are tracked in real-time in an augmented-reality environment. The features are used to correct the apparent positions of virtual objects appearing in the natural scene. The resulting system is significantly more stable and accurate than the same system without real-time video-based registration correction. Section 3.5.2 describes an interactive method for calibrating the video camera.

Dynamic Registration Correction in Augmented-Reality Systems

Michael Bajura
Department of Computer Science
UNC Chapel Hill
Chapel Hill, NC 27599-3175
bajura@cs.unc.edu

Ulrich Neumann
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
uneumann@cs.usc.edu

3.1 Abstract

This paper addresses the problem of correcting visual registration errors in video-based augmented-reality systems. Accurate visual registration between real and computer-generated objects in combined images is critically important for conveying the perception that both types of object occupy the same 3-dimensional (3D) space. To date, augmented-reality systems have concentrated on simply improving 3D coordinate system registration in order to improve apparent (image) registration error. This paper introduces the the idea of dynamically measuring registration error in combined images (2D error) and using that information to correct 3D coordinate system registration error which in turn improves registration in the combined images. Registration can be made exact in every combined image if a small video delay can be tolerated.

Our experimental augmented-reality system achieves improved image registration, stability, and error tolerance from tracking system drift and jitter over current augmented-reality systems. No additional tracking hardware or other devices are needed on the user's head-mounted display. Computer-generated objects can be "nailed" to real-world reference points in every image the user sees with an easily-implemented algorithm. Dynamic error correction as demonstrated here will likely be a key component of future augmented-reality systems.

KEYWORDS: Augmented Reality, Virtual Reality, Registration.

3.2 Introduction

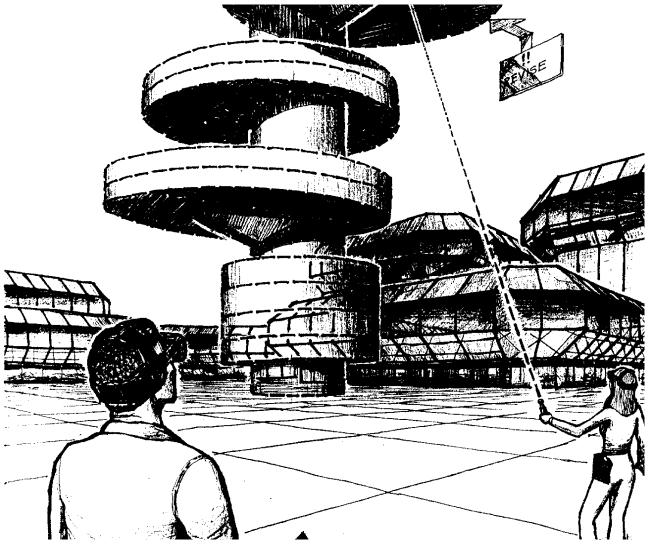

Figure 3.1: Example AR application: Visualization of a proposed building design

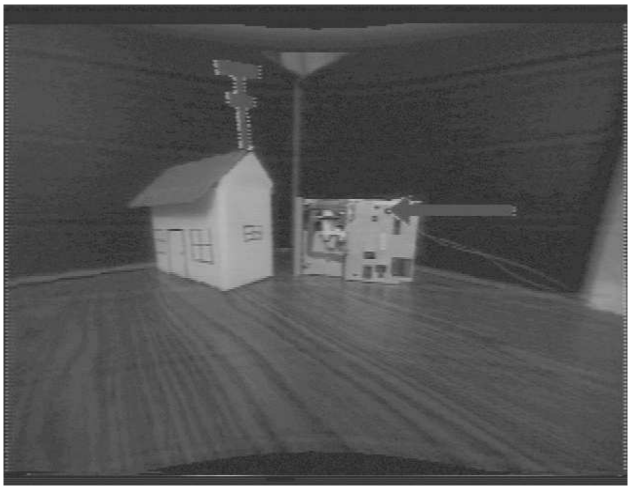

Figure 3.2: Experimental AR system showing dynamic registration of a virtual antenna and an annotation arrow which appear to be "nailed" in place

Augmented-reality (AR) systems allow users to interact with real and computer-generated objects by displaying 3D virtual objects registered in a user's natural environment. Figure 3.1 illustrates an application of this powerful visualization tool where a user can visualize an as-yet unbuilt building in its proposed natural setting. Other applications include interactive 3D illustrations for constructing and for maintaining complex machinery [Feiner et al., 1992] [Feiner et al., 1993] [Caudell and Mizell, 1992] and in-patient visualization of medical data, e.g., ultrasound [Bajura et al., 1992]. In all these applications it is vitally necessary for computer-generated objects and real-world objects to be visually registered with respect to each other in every image the user sees. If accurate registration is not maintained, the computer-generated objects appear to float around in the user's natural environment without having a specific 3D spatial position. Figure 3.2 is an image from our experimental AR system which dynamically corrects image registration on a frame-by-frame basis. It shows a computer-generated television antenna registered correctly on a toy house and a direction arrow registered correctly on a disk drive. The antenna and the arrow maintain correct registration in every image the user sees, including when the user is moving. The rest of this paper explains how this result is achieved and suggests future directions for dynamic registration correction. Section 3.3 describes the current model for augmented-reality systems and the sources of error in them. Section 3.4 explains a method for dynamically correcting registration error. Section 3.5 describes the implementation and results of our experimental AR system which dynamically corrects registration error. Conclusions and future directions follow in section 3.6.

3.3 Current Model for AR Systems

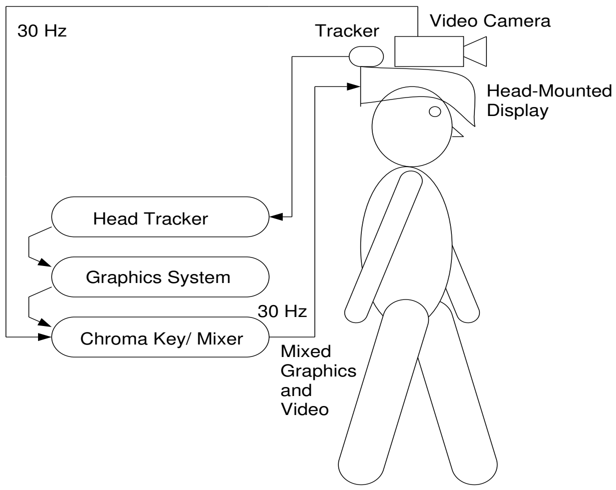

Figure 3.3: Typical Video-based AR System Components

Video-based augmented-reality systems are currently based on the model shown in figure 3.3. The user wears a head-mounted display (HMD) which presents combined images of both real and virtual (computer-generated) objects. Images of real objects are obtained from a video camera mounted on the user's display helmet. Images of virtual objects are generated by a graphics system. A tracking system reports the user's head position to the graphics system so it can render images with a virtual camera model of the video (real) camera's view of the world. The real and virtual images are typically merged with a chroma-key or video mixer for display in the HMD. For clarity in this paper, we discuss only the monocular case with one video camera mounted on the HMD. A stereo system would add a second video camera which would be treated as a second independent monocular system. Constructing a binocular HMD which presents correct stereopsis is problem addressed in [Edwards et al., 1993].

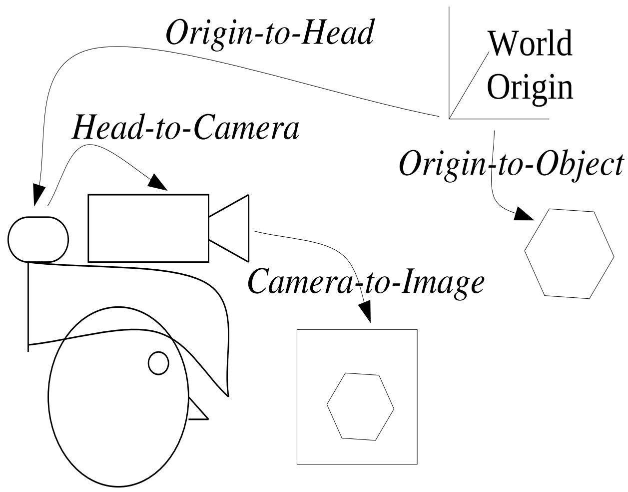

Figure 3.4: Transformations from Object to Image

The apparent registration between real and virtual objects depends on how accurately the virtual camera models the real one. Figure 3.4 shows a detailed transformation model for the virtual camera. Virtual objects are positioned by an Origin-to-Object transformation which specifies the position and orientation of a virtual object relative to a coordinate system origin. The virtual camera is positioned relative to the coordinate system origin by the composition of two transformations: Origin-to-Head, and Head-to-Camera. The Origin-to-Head transformation is reported by the tracking system and specifies the location of a tracking element's position on the user's HMD. The fixed transformation from this tracking element's position to the effective viewpoint of the real camera is the Head-to-Camera transformation. Virtual camera images are produced by a perspective projection onto a virtual image plane. A non-linear Camera-to-Image mapping is then applied which matches the field of view and lens distortion of the real camera. Typically the Head-to-Camera transformation and the Camera-to-Image mapping are determined by a calibration procedure such as the one described in section 3.5.2. It should be noted that this model does not address the problem of correcting for distortion in the HMD optics which is a separate problem from generating correctly registered images [Edwards et al., 1993].

Image registration error in combined real and virtual images is caused by the following types of errors:

1. The tracking system's origin is not aligned with the world coordinate system origin. This error causes all virtual objects to appear to be displaced from their proper positions.

2. The virtual Origin-to-Object transformation is not the same as the real Origin-to-Object transformation for a particular object. This error causes individual objects to appear out of position.

3. The virtual camera position is not the same as the real camera position. This can be caused by errors in either the static Head-to-Camera transformation or the dynamic Origin-to-Head transformation reported by the tracking system. The tracking system exhibits two types of error: temporal error, and position and orientation error. Position and orientation errors cause misregistration in all cases, while temporal errors cause misregistration only during user movement. Temporal errors are caused by a delay in sensing and reporting tracking information to the computer graphics system and the computer graphics system's delay in generating the appropriate virtual images [Adelstein et al., 1992].

4. The virtual Camera-to-Image mapping doesn't accurately model the real camera. The Camera-to-Image mapping abstraction is that any real camera can be modelled by an idealized pinhole camera with a particular center of projection, viewing direction, field of view, and distortion function. The distortion function is a 2D warp which accounts for the non-linearities found in lens-based projection systems. Errors in the Camera-to-Image mapping cause misregistration to vary with screen position.

3.4 Correcting Registration Error

The new idea presented here is to dynamically measure the registration error, or misregistration, in each combined image and use that information to correct the system errors that caused the misregistration. In the above model (figure 3.3), absolute correctness of all the transformations shown in figure 3.4 is needed for absolute image registration. This situation is much like the design of an "open loop" system from systems theory. An input generates an output which has errors. The only way to improve the system is to make each system component more accurate. Another idea from systems theory is a "closed loop" system where a system's error output is used again at the input to improve the system's output. The model advanced here resembles the "closed loop" design where the image registration error, or misregistration, is used to correct the transformation parameters which caused it. The type of correction which can be performed depends on two main factors:

1. the method used for detecting and measuring image misregistration, and

2. the uncertainty and image-space sensitivity of the different parameters to be adjusted.

Both are described below.

Different methods for measuring image misregistration dictate what kinds of correction can be performed. One way to measure image misregistration is to identify a recognizable point on each object to be registered. The image coordinates of each point are located in both the real and uncorrected virtual images. The difference between each point's position in each real image and corresponding uncorrected virtual image is the registration error, or misregistration, for the object corresponding with that point. This measure of misregistration can correct for errors such as camera orientation and sometimes camera position. A drawback with this measure is that neither the distance between an object and the camera nor an object's orientation can be estimated. Another way to measure image misregistration is to attempt to recognize an object's position, size, and orientation in each real image. This information can correct camera to object distance as well as relative object orientation.

Even if a particular misregistration measure does not allow estimation of a particular transformation parameter, misregistration can still be reduced. In many cases parameters which cannot be estimated can be assumed to be correct and the image registration error can be reduced by adjusting the remaining parameters. In other cases there is no way to separate the error contributions from different parameters and one or more must be adjusted depending on their relative uncertainty. The important point is that image registration error can be reduced even if some approximations are made.

The selection of which parameters to adjust depends on both their uncertainty and how sensitive image-space errors are to that uncertainty. For example, if the positions of objects are well known but the camera position and orientation are relatively uncertain, the camera position and orientation should be adjusted instead of object positions. Another example is camera position versus camera orientation. When an object is relatively close to a camera, its projection in image coordinates is more sensitive to the camera's position and less so to its orientation. For objects relatively far from a camera, the camera's orientation most strongly influences where an object's image appears.

This approach to correcting registration error can also be used to correct temporal errors. If the tracking system delay is longer than the delay in measuring image misregistration, the misregistration can be used to improve the most recent tracking system estimate. In video-based AR systems it is possible to effectively reduce the delay in measuring image registration to zero by delaying the real video image stream by the time it takes to measure image registration and generate corrected virtual images. This makes it possible to correct temporal and spatial image registration exactly in every image the AR user sees. If there is registration error, it is only because the error compensation algorithm failed. For applications which can tolerate minimal delays, potentially perfect registration can be achieved. This trade-off is not possible with optically based AR systems which allow the user to see his surroundings directly.

Some success at improving registration error has been achieved with autocalibration approaches [Gottschalk and Hughes, 1993] and predictive tracking techniques [Azuma and Bishop, 1994] [List, 1984] which use a state estimate to help predict current measurements. However these approaches still suffer from the "open loop" requirement for perfect tracking and calibration.

3.5 The Experimental System

This section describes an experimental AR system which corrects image registration error on a frame-by-frame basis. Section 3.5.1 describes the functional components of the system and what hardware is used for them. System calibration is discussed in section 3.5.2. Section 3.5.3 describes how registration error is corrected in the experimental system. Section 3.5.4 shows the results of operating the system both with and without dynamic registration correction.

3.5.1 System Components

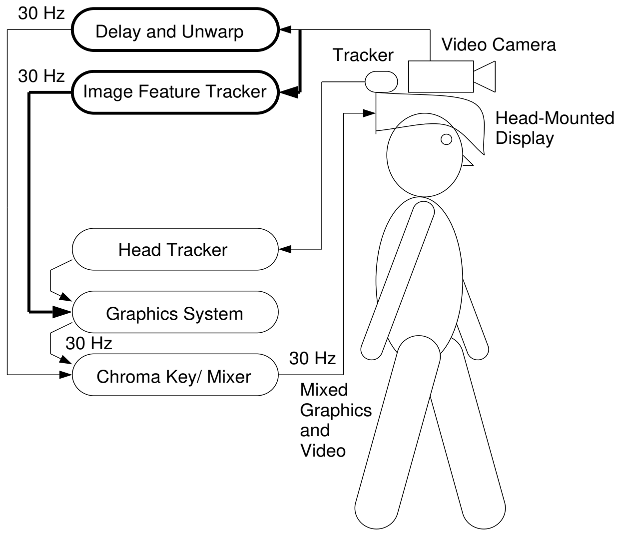

Figure 3.5: Video-based AR System Model with Registration Correction

Figure 3.5 is a schematic for the experimental AR system. It is similar to the system described in figure 3.3 except for the addition of a real-time video delay and unwarp pipeline and a real-time image feature tracker. The delay and unwarp pipeline delays video by a constant number of frames and optionally applies an inverse distortion function which converts the incoming signal into an equivalent pinhole camera image. With our hardware, it is more practical to undistort the real camera video images to match the undistorted virtual images instead of distorting the virtual images to match the real camera ones. The pipeline delay is adjustable but constant during operation. The pipeline delay is set to match the delay in generating the correct virtual image to mix with the corresponding real camera video frame.

The image feature tracker recognizes features in the real video stream and passes their image coordinates to the graphics system. The features to be detected are red LEDs driven by a 9V power supply. The LEDs are significantly brighter than other objects in the environment. The LEDs are detected by applying a brightness and image area threshold to each image. Correspondence between LEDs and the particular features they represent is established by matching detected LED positions with the nearest estimated feature positions in each corresponding uncorrected virtual image. To do this it is not necessary to render uncorrected virtual images. Only the feature positions must be computed. Once feature correspondence is established, the difference between each feature's position in each real image and it's estimated position in each corresponding virtual image can be used to render virtual images which are better registered with the real video images. The methods used for correcting registration are explained in section 3.5.3.

The camera used this experiment is a Panasonic GP-KS102 color CCD camera with a highly distorting 110 degree wide angle lens. The head tracking system is an Ascension A Flock of Birds magnetic tracking system. The delay and unwarp, image feature tracker, and graphics system are different software modules which utilize separate portions of the Pixel-Planes 5 graphics multicomputer at UNC [Fuchs et al., 1989]. Video is input to the Pixel-Planes 5 system via a real-time video digitizer and output via a standard double-buffered frame buffer. Although it is desirable to mix the real camera and virtual camera video signals digitally, the bandwidth requirements of 30 Hz operation require the use of an analog Sony CRK-2000 Universal Chroma Keyer video mixer.

The AR world of the experimental system consists of a virtual TV antenna positioned atop a real model house (where an LED is located) and a virtual arrow which indicates an adjustment screw on a real disk drive (also where an LED is located).

3.5.2 Calibration

Before the system can be operated, the Head-to-Camera transformation and the Camera-to-Image mapping must be estimated. This is done by operating the AR system and using manual feedback to converge on a solution. Optional compensation for non-linear lens distortion in the Camera-to-Image mapping is measured by examining a distorted camera image and finding a 2D warp function which converts that image into an undistorted one [Bajura, 1993]. If non-linear lens distortion is not considered, a best-fit calibration solution by matching field of view is possible even for distorting lenses.

Figure 3.6 shows how the Head-to-Camera transformation is initially estimated. A calibration fixture is used to represent a fixed position and orientation which are measured relative to the tracking system origin. When the camera is placed in a specific position and orientation relative to the calibration fixture, the position and orientation of the head tracking element is recorded. The Head-to-Camera transformation is the difference between the head tracking element's position and orientation and the camera's position and orientation. A calibration fixture is needed because the tracking system reports positions relative to a fixed but not precisely known origin. If the tracking

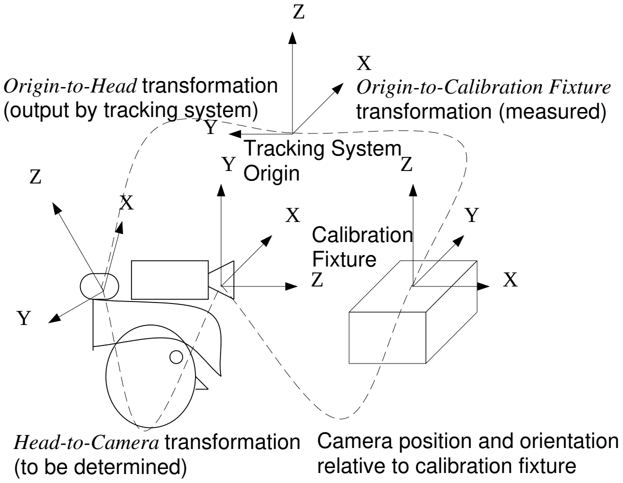

Figure 3.6: Calibration Transformations

system reported coordinates in a known coordinate system relative to itself a calibration fixture would be unnecessary.

The calibration fixture is located by using the head tracking element to perform rigid body rotations about each of the calibration fixture's coordinate axes. As rotations about each axis are performed, the tracking element's position and orientation are recorded and used to compute each axis of rotation. Because rigid body rotations are used it isn't necessary to know the offset of the tracking element from each axis of rotation beforehand. The position and orientation of the calibration fixture's coordinate system are determined once rotations about two axes are performed. By taking enough careful measurements it is possible to locate the calibration fixture to nearly the precision of the tracking system itself.

Ideally, only one measurement of the head tracking element is needed to estimate the Head-to-Camera transformation while the camera is simultaneously placed in both a specific position and a specific orientation relative to the calibration fixture. Because it is difficult to accurately position and orient the camera at the same time, separate measurements are made to estimate the position and orientation components of the Head-to-Camera transformation.

The Head-to-Camera estimation is refined by making further measurements while running the AR system. Virtual 3D coordinate axes are placed at the same position as the calibration fixture's coordinate axes determined above. If the Head-to-Camera transformation is correct, the virtual and real coordinate axes will appear in the composite images to be aligned in both position and orientation when viewed along the optical axis of the camera. If the coordinate axes are not aligned, the Head-to-Camera estimate can be improved with the following heuristics. Only rough estimates for the Camera-to-Image mapping are needed at this point because alignment along the optical axis isn't affected by either the field of view or the lens distortion model.

If the camera is positioned relatively far from the coordinate axes, misregistration is primarily due to orientation errors in the Head-to-Camera transformation. Rotations about the X and Y axes of the camera orientation should be made to align the axis positions. Rotations about the camera Z axis should be made to align the axis orientations.

If the camera is positioned relatively near to the coordinate axes, misregistration is primarily due to position errors in the Head-to-Camera transformation. Translations along the camera X and Y axes should be made to align the axis positions. Translations along the camera Z axis will move the camera viewpoint either in front of or behind a virtual point, for example the coordinate axis origin, when the camera is very close to that point.

Once the Head-to-Camera transformation has been adjusted so that the virtual and real coordinate axes appear to be aligned when viewed along the camera's optical axis, the field of view component of the Camera-to-Image mapping can be adjusted. This is done by viewing the coordinate axes at angles off the camera's optical axis and separately adjusting the camera's X and Y fields of view until the coordinate axes are realigned. Without lens distortion correction, alignment will not be possible for all off-axis viewing angles. However, misalignment may be minimal with lower distortion lenses.

Non-linear lens distortion in the Camera-to-Image mapping is calibrated by imaging a test pattern and finding a distortion function which undistorts the test pattern image. This is done by appealing to a basic rule of (linear) projective geometry: straight lines remain straight under projection. Scales may change and parallel lines may intersect, but the image of a straight line is always straight. If there is a mapping which converts images from a distorting camera into ones where all straight lines appear to be straight, then the distorting camera can be modelled by a composition of this mapping and a pinhole camera model.

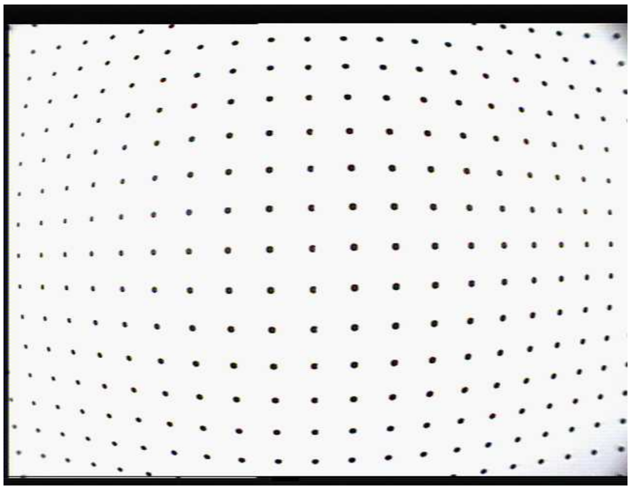

Figure 3.7: Distorted Image of Calibration Pattern

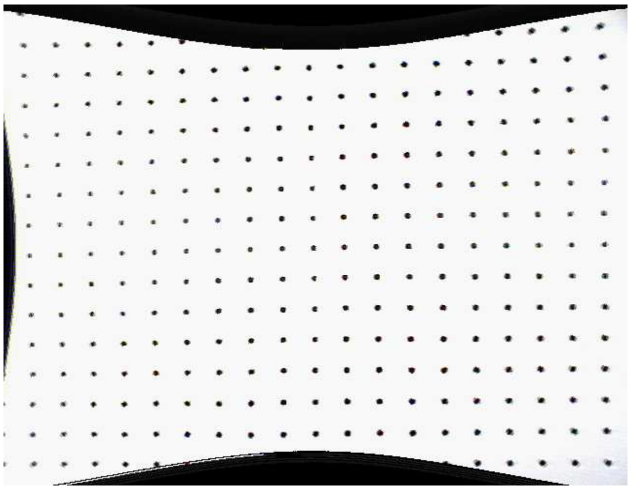

Figure 3.8: Corrected Image of Calibration Pattern

Figures 3.7 and 3.8 are images of a test pattern imaged with the 110 degree wide angle camera lens. Figure 3.7 is the distorted image output from the camera. Figure 3.8 is a corrected version of the same image. The correction is a radial distortion at the image center which accounts for most of the image distortion [Weng et al., 1992].

The important point about calibration is that it is difficult to do accurately, particularly when the tracking system used has noticeable tracking error throughout its working volume. Using a more accurate measuring device to measure the Head-to-Camera transformation would not eliminate errors in the AR system because camera position would still be a function of the tracking system which reports the Origin-to-Head transformation [Janin et al., 1993].

3.5.3 Correcting Registration Error

The image registration model of matching a point on each object makes it difficult to determine which particular errors are causing misregistration. One way to think about this is to consider the misregistration as a function of the camera position and orientation error (a composition of errors in the Origin-to-Head and Head-to-Camera transformations), Camera-to-Image mapping error, and Origin-to-Object transformation error:

$$\begin{aligned}\text{Misregistration} = f(&\text{Camera position and orientation error,}\\ &\text{Camera-to-Image mapping error,}\\ &\text{Origin-to-Object transformation error})\end{aligned}$$

Misregistration can be reduced by modifying one or more of the parameters which might be causing it. Two approaches to reducing registration error are studied in this experiment. One approach assumes that the camera position and orientation are absolutely correct and that misregistration is due to errors in the Camera-to-Image mapping and Origin-to-Object transformation. The second approach assumes that the Camera-to-Image mapping and Origin-to-Object transformations are correct and that the camera position and orientation are in error. Neither of these approaches is optimal in the sense of minimizing error by smoothly adjusting all the possible parameters according to parameter certainty and registration sensitivity, e.g. optimal filtering. Such an analysis is difficult to make and may not be any better than making a few reasonable assumptions. Both of the approaches tried here are relatively easy to implement and are sensible in certain situations.

In the first correction approach, if the reported position and orientation of the virtual camera are assumed to be correct, there is no way to tell whether registration errors were caused by incorrect Camera-to-Image mapping, incorrect Origin-to-Object transformations, or both. By making a further assumption that the Camera-to-Image mapping is also correct, object positions alone can be adjusted to account for any registration error. To render a corrected image, each misregistered object is temporarily displaced to a position where it will appear to be registered correctly. This correction produces combined images with no measured registration error. Since the registration metric gives no estimate of distance between each object and the camera, virtual objects are displaced on a constant radius (rotated) from the virtual camera viewpoint. This maintains the best estimate of distance between the camera and each object so that objects don't grow and shrink unnaturally.

In the second correction approach, the virtual camera viewpoint is corrected to reduce registration error while object position and camera distortion are assumed to be correct. If enough features are visible it is theoretically possible to compute both camera position and orientation from the 3D (X,Y,Z) feature positions and their corresponding (U,V) image locations. If the feature positions aren't degenerate, the camera position and orientation can be recovered by non-linear methods with a minimum of 4 points and by linear methods with a minimum of 6 points [Horaud et al., 1989] [Ganapathy, 1984b]. Trying to correct the camera position this way isn't practical for at least three reasons. First, there is no way to guarantee enough features will be visible in every image. Second, these solution methods are highly sensitive to noise and spatial feature distribution. Third, a good estimate of the virtual camera position is already available.

The easiest simplification to make is that the virtual camera position is correct as reported by the Origin-to-Head and Head-to-Camera transformations and that the registration error is entirely due to camera orientation error. This is a good assumption for three reasons. First, orientation corrections can be made when only one feature is visible. If more than one feature is visible a best-fit solution can be found. Second, under the assumption that most objects are relatively far from the camera, which is true in most AR applications, registration errors are much more sensitive to errors in camera orientation than camera position. This means that solving for camera position is unstable (sensitive to errors) and that solving for camera orientation (when camera position is fixed) is well-behaved (relatively insensitive to errors). Third, tracking system data has more error in rotation than in translation. This is because HMD wearers typically rotate their heads faster than they translate them and the head tracking system used incurs significant delays in reporting measurements (temporal error) [Liang et al., 1991]. In the experimental system, camera orientation error is adjusted by considering only one "reference" feature position and rotating the virtual camera to align that position. This is only an approximation which can correct the alignment of a particular point but not an orientation about that point.

3.5.4 Registration Results

The experimental system (figure 3.6) can be operated in nine different modes by different selections of the two parameters real-video-delay and registration-correction-method. Real-video-delay is one of:

1. no delay or distortion correction
2. delay without distortion correction
3. delay with distortion correction.

Registration-correction-method is one of:

A. none

B. correction by adjusting Camera-to-Image mapping and/or Origin-to-Object transformations (move the object)

C. correction by adjusting camera orientation (rotate the camera).

The results of different combinations of these parameters are described below:

(1,A): This "open loop" mode is equivalent to the "current model" shown in figure 3.3. Figure 3.9 shows the result: the virtual objects are not aligned with their proper positions and lag noticeably behind during user movement in spite of careful calibration and system tuning.

(1,B): This option has good registration at the object feature positions except during user motion when the registration still lags noticeably. It appears to be possible to shake the virtual objects from their proper positions, but they always return. This case shows the simple power of the "closed loop" system model over the "open loop" system model in figures 3.3 and 3.9. Despite the lack of lens distortion correction, noticeable lag, and various other errors, the virtual objects still appear to belong in specific spatial positions, a result not easily achieved without dynamic registration correction.

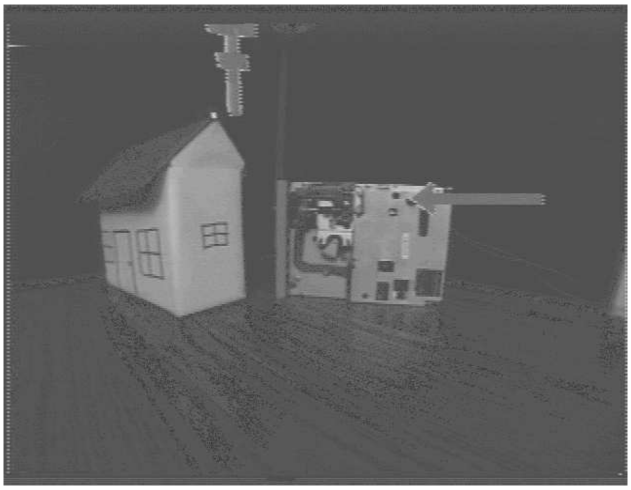

Figure 3.9: “Open loop” Mode without Dynamic Registration Correction or Distortion Correction: Virtual objects “swim” around and are poorly aligned.

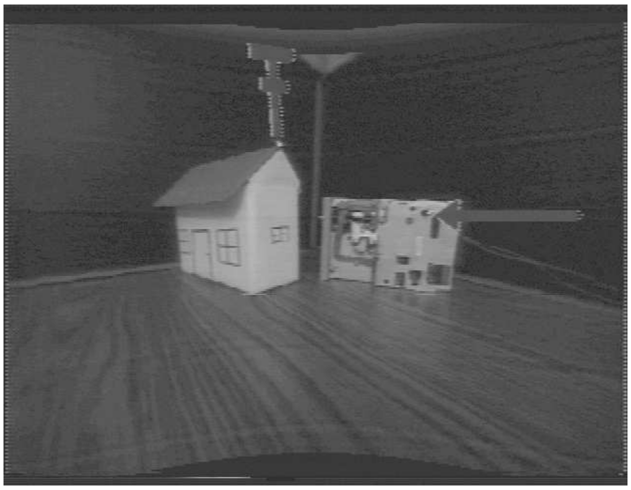

Figure 3.10: “Open loop” Mode with Video Delay and Optical Distortion Correction: Virtual objects do not swim as much and registration is somwehat improved.

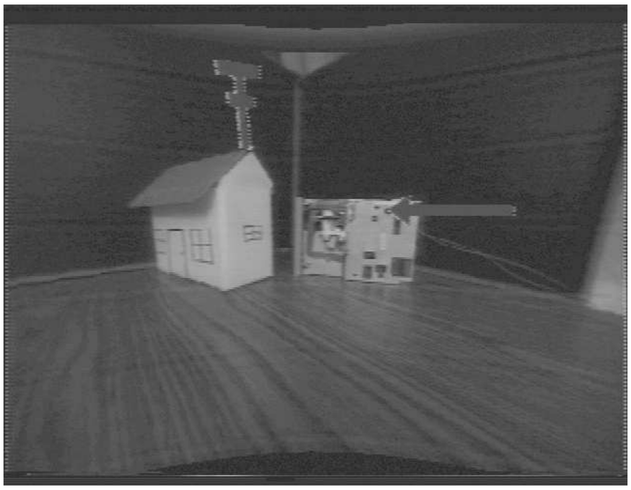

Figure 3.11: “Closed loop” Mode with Optical Distortion Correction, Video Delay, and Dynamic Correction of Object Position and Lens Distortion: Virtual objects appear to be “nailed” to their reference positions.

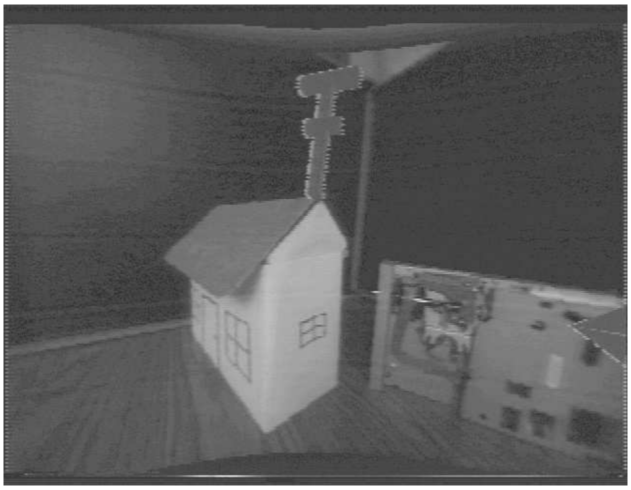

Figure 3.12: “Closed loop” Mode with Optical Distortion Correction, Video Delay, and Camera Orientation Correction to Align Base of Antenna: Adjustment screw is steady but slightly misregistered.

(2,A), (2,B): These combinations have the same static results as (1,A) and (1,B) above. However the registration error during motion (temporal registration error) is extremely small because the real video delay is the same as the tracking and image generation delay – the dynamic registration appears to be the same as the static registration. The reduction in the "swimming" of the virtual objects during motion makes them appear much more stationary and solid, even in the case of (2,A) where the registration is poor.

(3,A): The addition of lens distortion correction without registration correction produces the best "open loop" operation possible with the experimental system (figure 3.10). The lens distortion correction improves registration considerably but the virtual objects still wander slightly during movement and appear in different positions as the tracking system exhibits errors within its working volume.

(3,B): This combination of distortion correction, delay, and registration correction by displacing objects produces the best registration in the experimental system (figure 3.11). In all cases, during both static and dynamic viewing, the virtual objects appear to be registered correctly with respect to their reference positions. They appear to be "nailed" in place.

(3,C): Here only the reference position for the TV antenna is used to adjust the virtual camera orientation while the real video is corrected for distortion and delayed(figure 3.12). No registration correction is made for lens distortion or object position errors. This combination produces the 2nd best registration after combination (3,B). The base of the antenna appears to be registered correctly on the house, but the arrow on the disk drive adjustment screw consistently appears to be just a bit low. This misregistration could be caused by errors in the Origin-to-Object transformations for the TV antenna and disk drive screw or by errors in the initial camera orientation which aren't completely corrected with this method.

(1,C), (2,C): These combinations do not make sense. Without lens distortion correction it is not possible to modify the camera position to improve registration for more than one object.

3.6 Conclusions and Future Directions

- Building augmented-reality systems with accurate registration is difficult. The visual registration requirement between real objects and virtual objects exposes any measurement or calibration error in an AR system. The strongest argument in favor of dynamic compensation is that no matter how much measurement and calibration are performed, there may (will) still be errors in the composite images. What the real camera sees must be taken as the ground truth and the registration error between the real and virtual images must be corrected by compensating for errors in calibration, tracking, and/or distortion correction. It is more practical to measure and correct errors using a “closed loop” design than to avoid making them in the first place with an “open loop” design.

- The experiment described here demonstrates the importance and feasibility of dynamically measuring and correcting image space registration error. The experimental system is more stable and better aligned than systems without registration correction.

- The idea of measuring and correcting image registration error has implications for the design of future augmented-reality systems. Since feedback can compensate for tracking errors, in essence becoming part of the tracking system itself, less accurate and less expensive tracking systems may be feasible. Optical tracking systems [Azuma and Bishop, 1994] could be designed to use stationary cameras to track a user’s position while cameras on the user’s head could look outward to determine the user’s orientation. Feedback also reduces the accuracy requirements for lens distortion correction and system calibration.

- The success of registration correction depends on the ability to accurately measure registration in the first place. This is not a simple task in general. The experiment described here uses an oversimplified method for measuring registration which may not be practical in many environments. A large amount of work in this area has already been done by the computer vision community. Hopefully some of their results can be applied to AR systems.

- Correct occlusion cues are still needed for augmented-reality systems to be truly believable. This method of registration only works for virtual objects which are completely in front of real ones. What is really needed is a way to sense positions and depths in the environment from the real camera. With such information, the reference positions could be used to position virtual objects which could be hidden properly if they were obscured.

Chapter 4

Sensing the Environment and Generating Synthetic Geometry

This chapter demonstrates the feasibility of sensing surface geometry in an AR user's natural environment and using the geometry a) to present proper occlusion and detect collisions between real and virtual objects and b) to correct camera pose. An example medical application is presented showing a virtual medical ultrasound transducer scanning a (mannequin) patient's abdomen. Point features are tracked in the monocular image sequence seen by the HMD. The 3-D positions of those points are recovered by optimal filtering to generate a polygonal surface. The virtual ultrasound transducer scans this surface and views of the transducer are reprojected onto the original images. The purpose of this work is to demonstrate a sequential (incremental) algorithm which could accomplish this task in real-time and to estimate its computational requirements.

4.1 Overview

Figures 4.1 (also figure 2.5) and 4.2 outline the experimental system described here. The Real-World-View is captured by a Panasonic PG-KS102 color CCD video camera with a Cosmicar 12.4 mm $f/1.8$ C-Mount lens. The camera is mounted on a FARO [Faro, 1997] mechanical tracking arm which is the Tracking System. As the experimental system operates, points visible in the Real-World-View are tracked over time. Their 2-D image positions are combined with arm 3-D tracking information to locate their 3-D coordinates. A Real-World-Model is constructed by converting these point coordinates into a 3-D triangular mesh. The position of the camera is corrected based on the visibility of already-located 3-D points. A virtual ultrasound transducer simulates a sweep across the patient's abdomen showing scanned data visible inside a synthetic hole inside the patient. Images are generated to show visual occlusion from a World-Model encompassing both real and synthetic

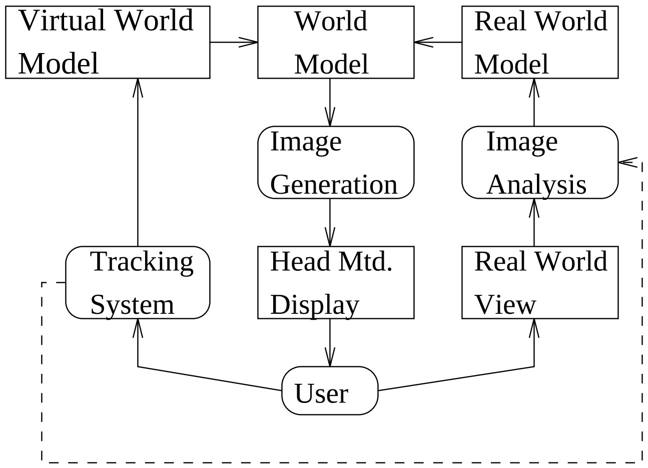

Figure 4.1: AR System Model

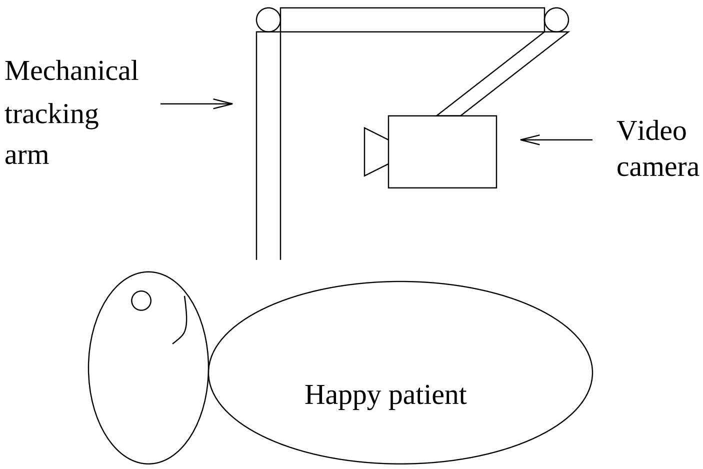

Figure 4.2: Experimental Setup

geometry.

The following sections describe parts of the experimental system in more detail. Section 4.2 describes how feature points are located and tracked through a sequence of images. Section 4.3 describes how the 2-D image positions of these feature points are incrementally combined with tracking information to locate their 3-D positions. Section 4.4 examines the 3-D positional accuracy achieved with this method on a test fixture. Section 4.5 describes a method for generating a 3-D surface mesh from feature position data. Section 4.6 explains how camera pose can be estimated from feature positions. Section 4.7 discusses rendering. Section 4.8 estimates the computational requirements of this method. Section 4.9 presents results of the example application.

4.2 Tracking Points in Images

Points are tracked through a monocular image sequence because the computation is local, incremental, and amenable to image-space parallelization making real-time computation more tractable. Depth-from-Stereo (or more than one image), although the intuitive approach for a HMD system, is unattractive because of the difficulty in accurately and quickly matching points in different images. In addition, even a working stereo approach would not solve all problems because optimal estimation and world-model reconstruction would still be needed to improve accuracy and perform occlusion. An alternative approach which uses a stereo image sequence is proposed in section 4.3.

Shi and Tomasi [Shi and Tomasi, 1994] [Tomasi and Kanade, 1991b] describe a method based on Lucas' earlier work [Lucas and Kanade, 1981] for identifying feature windows based on local image gradient information and then tracking the movement of those windows over time. As feature windows deform in successive images, parameters for window movement and deformation are estimated. When a feature window becomes too dissimilar from its original viewing, it is discarded. Subpixel registration is estimated by maximizing a correlation function between the first viewing of a window and the current image by varying window shift and distortion parameters. When the correlation measurement falls below a threshold, the window is discarded as unreliable. It is an open question however whether a feature's location is better estimated by the metric originally used to select a feature window (by subpixel accuracy if needed) or by minimizing a shifted and distorted function. Shi and Tomasi initially select feature windows with integer pixel accuracy and then compute subpixel shifts of those windows. Their motivation for using windows is based in part on strict needs for maintaining and verifying feature correspondence.

In contrast to the the applications of Shi and Tomasi, the approach taken here is that it is sufficient to simply track local peaks of the metric function used to identify likely features. Features become unsuitable for tracking when their peak value (in the metric function) falls below a threshold. The savings of not computing correlation windows is offset by the greater chance of mistaking one feature for another. If this happens, it will be detected at the reconstruction level, not at the feature tracking level. To avoid aliasing of peaks, it is implicitly assumed that feature peaks cannot jump arbitrarily far — a reasonable assumption in a real-time and incremental system. Identifying features only to single pixel accuracy is sufficient unless the "noise" added by this process becomes significant. This approach simplifies the feature tracking problem by bypassing the computation of correlation windows around each feature.

The limitation of features moving no more than than a fixed distance between images constrains the movement of the camera (or user's head) depending on the field-of-view of the camera. This restriction is somewhat arbitrary depending on the technology used. This demonstration system sets a 10-pixel limit on feature movement between images and analyzes images at 30 Hz. This means that with a 28 degree field-of-view camera and 640 pixel wide images, the camera cannot rotate more than 13 degrees/second without losing feature tracking:

$$\frac{10 \text{ pixels} \times 30 \text{ Hz} \times 28 \text{ degrees}}{640 \text{ pixels}} = 13.125 \text{ degrees / second.} \tag{4.1}$$

When sampling at 60 Hz (using both fields of the camera images) this limit would be doubled. A wider angle lens would also increase the limit. Holloway [Holloway, 1995, pages 131–132] reports mean rotation of 20 degrees/second with peaks as high as 50 degrees/second in the heads of test subjects wearing HMDs. With current technology, this approach is barely adequate for tracking features, but it is not hard to imagine sampling images at a rate faster than 60 Hz in the future and/or performing feature tracking directly on the CCD image plane. A similar dynamic constraint exists for camera (head) translation although at arm's length rotation is the dominant source of image shift [Holloway, 1995].

As features are tracked over time, knowledge of the camera's position can be used to match feature peaks. This is done by searching for feature matches in the metric image based on where features ought to appear given their previously estimated 3-D positions. This matching is not possible at first when the 3-D positions of features are not well estimated. This extends the range of feature movement somewhat but still requires each feature to remain within a certain window of expectation.

The working assumption for identifying features on surfaces is that there are features to identify. In addition to a surface's natural textures, features could take the form of shadows cast across a surface or a texture projected onto a surface. There can be complications with specular (view-point dependent) highlights and "T" junctions created by occlusion which are addressed during reconstruction.

Features selected and tracked by this method exhibit a certain amount of drift according to changes in viewpoint. The amount of drift varies with the scale of the underlying functions used.

4.2.1 Metric Function

The function used for feature selection is similar to that used by Tomasi and Shi [Tomasi and Kanade, 1991b] [Shi and Tomasi, 1994] who define a feature as something which can be tracked reliably from one image to another. Considering an image as a height surface, their criteria for correlating feature windows require that a feature point have high curvature in both principal directions. A perfectly flat window having low curvature in both directions doesn't have enough information for good correlation. Edge points have high curvature across an edge but no curvature along it reflecting the possibility of accurately correlating a window across an edge but not along it. Points representing dots (parabolic points) and saddle points (hyperbolic points) which have significant curvature in both principal directions are selected as features using this curvature criteria.

The metric function is computed from a curvature matrix formed by second-order derivatives of the image function. The eigenvalues of this matrix are the principal curvatures at each image point [O'Neill, 1966].

$$\begin{bmatrix} \frac{\partial^2 I}{\partial^2 u} & \frac{\partial^2 I}{\partial u \partial v} \\ \frac{\partial^2 I}{\partial v \partial u} & \frac{\partial^2 I}{\partial^2 v} \end{bmatrix} = \text{Curvature Matrix.} \tag{4.2}$$

The eigenvalues (or principal curvatures) e_1 and e_2 of the curvature matrix are computed by:

$$\text{radical} = \sqrt{\left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right)^2 - 4\left(\frac{\partial^2 I}{\partial^2 u}\frac{\partial^2 I}{\partial^2 v} - \left(\frac{\partial^2 I}{\partial u \partial v}\right)^2\right)}, \tag{4.3}$$

$$= \sqrt{\left(\frac{\partial^2 I}{\partial^2 u} - \frac{\partial^2 I}{\partial^2 v}\right)^2 + 4\left(\frac{\partial^2 I}{\partial u \partial v}\right)^2}, \tag{4.4}$$

$$e_1, e_2 = \frac{\left(\left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right) \pm \text{radical}\right)}{2}. \tag{4.5}$$

The metric function is the smaller magnitude eigenvalue at each point.

$$\text{Metric Function} = \text{smallest magnitude eigenvalue of } e_1 \text{ and } e_2. \tag{4.6}$$

This is the function used by Tomasi and Shi.

Ideally this function would be computed at different scales. In practice it has been sufficient to simply compute the curvature function at a single relatively important scale. The scale selection tradeoff is that at large scales features are identified more coarsely than at small scales. At large scales fewer but more important features are identified, however the feature points selected are not the ones one would usually identify as points. For example, at a small scale, the peak of curvature for a corner is very near the actual corner. At a larger scale, the peak of curvature for a corner occurs further in the interior of the corner. The drift in feature location due to changes in viewpoint is magnified at larger scales.

Empirically the scale $\sigma = 10$ pixels has been a good compromise. At this scale is has been sufficient to ignore saddle-point areas completely (where the product of eigenvalues is negative) and to identify features only in parabolic regions. This makes it possible to substitute an approximate function similar to the original function but one that is easier to compute because it lacks a square root function evaluation.

A simpler function to compute is:

$$\text{Simple Metric Function} = \frac{\partial^2 I}{\partial^2 u}\frac{\partial^2 I}{\partial^2 v} - \left(\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}\right)^2 \tag{4.7}$$

To see how this works, consider points in parabolic regions where the smaller eigenvalue (principal curvature) is maximum. This occurs at umbilic points where the eigenvalues are equal and the radical (equation 4.3) is 0:

$$\text{radical} = \quad 0 = \sqrt{\left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right)^2 - 4\left(\frac{\partial^2 I}{\partial^2 u}\frac{\partial^2 I}{\partial^2 v} - \left(\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}\right)^2\right)} \tag{4.8}$$

$$0 = \left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right)\sqrt{1 - \frac{4\left(\frac{\partial^2 I}{\partial^2 u}\frac{\partial^2 I}{\partial^2 v} - \left(\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}\right)^2\right)}{\left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right)^2}}. \tag{4.9}$$

From equation 4.9, if the quantity

$$\frac{4\left(\frac{\partial^2 I}{\partial^2 u}\frac{\partial^2 I}{\partial^2 v} - \left(\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}\right)^2\right)}{\left(\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}\right)^2} = 1, \tag{4.10}$$

the radical becomes 0. This quantity is normalized with respect to curvature. By scaling this value by the square of the curvature and discarding the constant 4, we have the simpler metric function (equation 4.7).

At points in parabolic regions where there is a maximum in the smaller eigenvalue (principal curvature), the radical (equation 4.3) is 0 and the principal curvatures are:

$$e_1, e_2 = \left(\frac{\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}}{2}\right). \tag{4.11}$$

At this point, there is a peak in the simple metric function:

$$\text{Peak of Simple Metric Function} = \left(\frac{\frac{\partial^2 I}{\partial^2 u} + \frac{\partial^2 I}{\partial^2 v}}{2}\right)^2. \tag{4.12}$$

A curvature cutoff criteria selects the strongest curvature peaks of the metric function. Empirically a cutoff of 1/8 of the highest function value in each image works well.

4.2.2 Why Not Lines?

The chief argument against recognizing lines as primitive features in an image sequence is that lines are not local image structures. Recognizing them in real-time by image-space parallelism is not as straightforward as detecting local point-like features. It would be possible to use lines in the reconstruction process and to help segment the generated surfaces but that subject is not pursued here.

4.3 3-D Reconstruction Method

Incremental optimal (Kalman) filtering [Lewis, 1986] is used to reconstruct 3-D feature positions from their 2-D positions in the image sequence. This is fairly straightforward because the tracking arm provides a good estimate for the camera position for each image. For each feature, its position in each successive camera image defines a ray from the current camera position which intersects the feature at some unknown distance from the camera. The reconstruction is carried out by combining the intersection of successive rays from the camera into an updated 3-D gaussian probability distribution. As the camera views each feature from more and more different positions, the gaussian probability distribution for each feature becomes more focused on its actual position.

The camera model (described in appendix A), the position of the mechanical tracking arm, and the 2-D image position of each feature are used to compute rays from the camera on which each feature in each image might lie. Each ray is then converted into a long thin gaussian probability distribution representing the relative uncertainty of each feature's distance from the camera along a particular viewing ray and relative certainty perpendicular to a particular viewing ray. The gaussian probability distributions are incrementally combined into the the highest probability intersection point for rays of each feature.

The camera viewing position and ray direction for each feature are computed by first expressing the position reported by the mechanical arm as a 3×4 matrix, $TP_{3\times 4}$, which transforms homogeneous points from world coordinates into tracking arm position coordinates. Using the transformation matrix $LP_{3\times 4}$ and the distortion function $PR_{nonlinear}$ (defined by calibration using the camera model in appendix A) we have:

$$PR_{nonlinear} \circ LP_{3\times 4} \circ TP_{3\times 4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} U' \\ V' \\ 1 \end{bmatrix}, \text{ or}$$

$$M_{3\times 4}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = PR^{-1}_{nonlinear}\begin{bmatrix} U' \\ V' \\ 1 \end{bmatrix} = \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}, \tag{4.13}$$

where $M_{3\times 4} = LP_{3\times 4} \circ TP_{3\times 4}$, $[X, Y, Z, 1]^T$ represents the unknown homogeneous world coordinates of a feature, $[U', V', 1]^T$ represents the known homogeneous image coordinates of a feature in a particular image, and $[U, V, 1]^T$ represents the distortion correction of $[U', V', 1]^T$. Writing out equation 4.13 and normalizing for homogeneous coordinates, we have two plane equations whose intersection is the line containing the camera viewing point and feature position:

$$(m_{1,1} - U\ m_{3,1})X + (m_{1,2} - U\ m_{3,2})Y + (m_{1,3} - U\ m_{3,3})Z = U\ m_{3,4} - m_{1,4} \tag{4.14}$$
$$(m_{2,1} - V\ m_{3,1})X + (m_{2,2} - V\ m_{3,2})Y + (m_{2,3} - V\ m_{3,3})Z = V\ m_{3,4} - m_{2,4}. \tag{4.15}$$

Thinking of $M_{3\times 4}$ as a system of linear equations, the solution to $M_{3\times 4}$ is the center of projection, or camera position. The cross product of the orientations of the planes defined by equations 4.14 and 4.15 identifies the direction of the ray (modulo a sign change) from the camera position to a feature. The sign of the ray direction is correct when its dot product with the Z axis of $M_{3\times 4}$ (the third row of $M_{3\times 4}$) is positive(looking through the front of the camera, not the back).

Once a ray is constructed, it is converted into an equivalent gaussian probability representation for inclusion in the optimal filtering process. The major axis of the probability distribution along the ray is large, representing the uncertainty in distance between the camera and a feature. The minor axes of the distribution perpendicular to the ray are relatively small, corresponding with the spatial uncertainty from tracking and camera calibration errors. The center (mean) of the distribution is estimated to be the same distance from the camera along the viewing ray as the current feature position estimate. If no previous estimate is available, an arm's length guess of 50 cm in front of the camera is used.

The covariance matrix representing a ray's probability distribution is formed by a similarity transformation [Strang, 1988]. In the local coordinate system of the gaussian distribution, the covariance is represented by a 3×3 diagonal matrix whose entries are the variances along each axis:

$$D_{3\times 3} = \begin{bmatrix} \text{var}_{\text{major}} & 0 & 0 \\ 0 & \text{var}_{\text{minor}} & 0 \\ 0 & 0 & \text{var}_{\text{minor}} \end{bmatrix}. \tag{4.16}$$

The covariance in the world coordinate system is formed by the product of a rotation transformation from world coordinates into covariance distribution coordinates, the simple covariance matrix $D_{3\times 3}$, and the inverse rotation transformation from distribution coordinates back into world coordinates:

$$R_{3\times 3} = L^{-1}_{3\times 3} \circ D_{3\times 3} \circ L_{3\times 3}, \tag{4.17}$$

where $R_{3\times3}$ is the covariance matrix representing a ray's probability distribution in world coordinates and $L_{3\times3}$ is the rotation matrix which transforms world coordinates to gaussian distribution coordinates.

The equations to update each feature position for each image are simple and can be computed in real-time. First system noise (drifting uncertainty) is added to a feature's covariance matrix, $P_{3\times3}$:

$$Pold_{3\times3} = Pold'_{3\times3} + \begin{bmatrix} noise & 0 & 0 \\ 0 & noise & 0 \\ 0 & 0 & noise \end{bmatrix}. \tag{4.18}$$

Next, the feature covariance matrix and feature position are updated with the new measurement:

$$K_{3\times3} = Pold_{3\times3}(Pold_{3\times3} + R_{3\times3})^{-1} \tag{4.19}$$

$$Pnew_{3\times3} = (I - K_{3\times3})Pold_{3\times3} \tag{4.20}$$

$$x_{\text{new}} = x_{\text{old}} + K_{3\times3}(z_{\text{current}} - x_{\text{old}}), \tag{4.21}$$

where $Pold_{3\times3}$ and $Pnew_{3\times3}$ are the old and new (updated) covariance matrices for a feature, respectively, and $()^{-1}$ represents matrix inversion. The common factor $K_{3\times3}$ in equations 4.20 and 4.21 is referred to as the Kalman gain [Lewis, 1986, page 70]. x_{old} and x_{new} are the old and updated feature position estimates, respectively. z is the estimated mean of the measurement seen by the camera. This value is estimated to be at the same depth from the camera as the previous estimate of a feature's position but along the current ray. If no previous position is available, an arm's length guess of 50 cm from the camera position is used. Guessing a feature's position and then estimating near that position converges to the right answer because the intersection of long and narrow probability distributions strongly affects the position update equation (4.21).

The value selected for system noise ($noise$) affects the performance of the system. If it is too high, making the current estimate for a feature position unreliable, feature positions will be updated to effectively follow the position of the camera and therefore will not be useful in correcting the camera position. If system noise is too low, features will not converge quickly or shift easily if they are displaced.

As the filtering process proceeds, a cutoff is defined for when a feature has been accurately located versus when its position is still relatively unknown. The largest magnitude entry in a feature's covariance matrix is a good measurement for this. It represents (roughly) the largest variance (σ^2) of the error distribution in some direction. Until the largest magnitude entry of a feature's covariance matrix falls below a threshold, the feature's position is not considered to be reliable.

The cutoff determination partially addresses the problems of features identified and tracked along moving contours ("T" junctions) and features generated by specular highlights. Because these feature positions change with viewing angle, they do not converge to a single point and fail the cutoff test. This is not perfect however as specular reflections (or just simple reflections) often alias to appear at points corresponding to their reflected positions.

To apply optimal filtering theory, the underlying assumption is that the errors in this system can be approximated a gaussian probability distributions. The true probability distribution for a feature seen in an image is conical growing larger at distances further from the camera. To approximate this cone as a gaussian probability distribution, the "width" of the gaussian distribution is selected to represent the width of this cone at a nominal distance from the camera. Errors in the camera position due to calibration inaccuracies probably do not have zero mean, although no analysis of this is offered. In addition, the metric function used for feature selection tends to select a different "point" for a feature depending on viewing angle.

4.3.1 Interpolating the Camera Position

Camera position accuracy directly effects the accuracy of 3-D reconstruction. The accuracy of the camera position depends on the accuracy of the FARO tracking arm which exhibits both absolute and dynamic errors. The absolute error of the tracking arm (reported to be 0.46 mm for 2 sigma error) is taken as given.

The factors affecting the dynamic error of the tracking arm are:

1. Interpolating the arm's position between tracking reports, and

2. Compensating for the time offset between tracking reports and video frames.

Video frames are acquired at nearly 30 Hz while the tracking arm generates positional reports at roughly 27 Hz. Because of pipelining, both in the video and tracking systems, the tracking reports and video frames are not synchronized when they are recorded in computer memory. However because they are recorded in real-time there is a fixed time offset between the two systems.

To get an estimate of the camera position when camera images are taken, linear interpolation is used between adjacent tracking arm reports. Position and orientation are interpolated separately. The position is computed by simple linear interpolation between adjacent tracking reports while the orientation is linearly interpolated in quaternion space and then renormalized to produce an intermediate orientation value.

The fixed time offset between tracking reports and video frames is estimated empirically. When examining waveforms of both the expected position of a feature in an image sequence and its actual position the time shift is readily apparent. In the test system the video frames lead the tracking reports by approximately 0.06 seconds. This measurement doesn't reflect the total delay in the system, which is a separate measurement. This measurement reflects the timing difference between the video and tracking system pipelines.

4.3.2 Extension to 2 or More Cameras (3-D matching)

The extension to using two (or more) cameras with this reconstruction algorithm is to compute correspondence in 3-D world space instead of 2-D image space. Initially each camera is treated independently. When the 3-D position of a particular feature seen in one camera is known accurately enough, it can be matched with other views of the same feature seen in other cameras. This idea is not explored here.

4.4 3-D Reconstruction Accuracy

Figures 4.3, 4.4, 4.5, and 4.6 show the 3-D reconstruction error for a point on a test fixture. Each of the first three figures (figures 4.3 - 4.5) shows the signed error in the computed position in one of the X, Y, and Z dimensions. The last figure (figure 4.6) shows the total unsigned error. The horizontal axis on each graph is the time in camera frame numbers and the vertical axis is the distance measurement. Frames were collected at 30 Hz. In this sequence it took 30 frames (1 second) to converge within approximately 5 mm of the correct measurement. After 120 frames (4 seconds) the measurement was accurate to less than 2 mm. The rate of convergence is most strongly influenced by movement of the camera. A feature's position converges quickly once the camera as moved far enough to make an accurate triangulation.

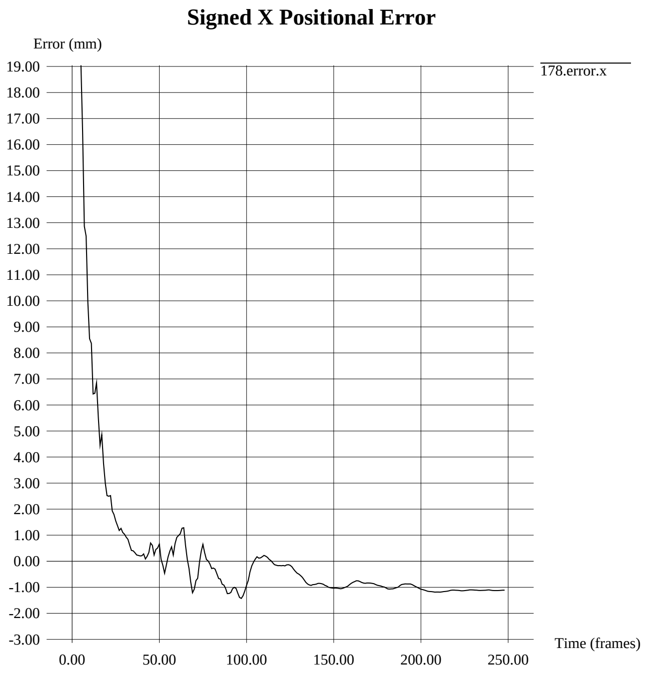

Figure 4.3: Reconstructed 3-D Positional Error in X

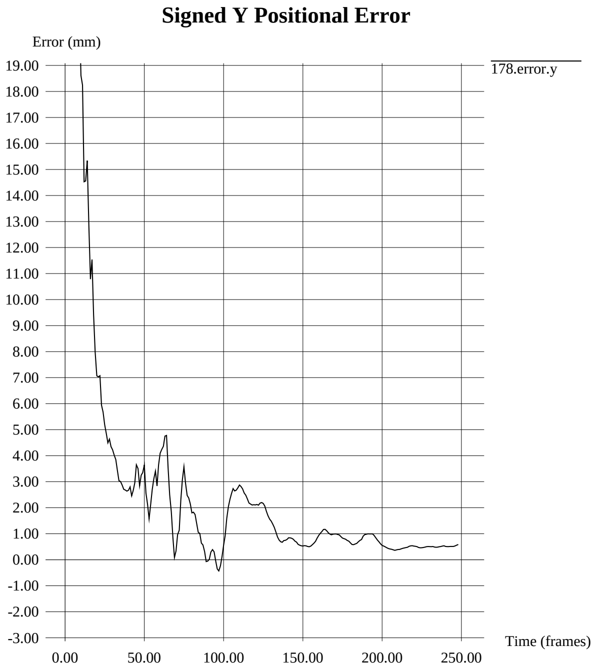

Figure 4.4: Reconstructed 3-D Positional Error in Y

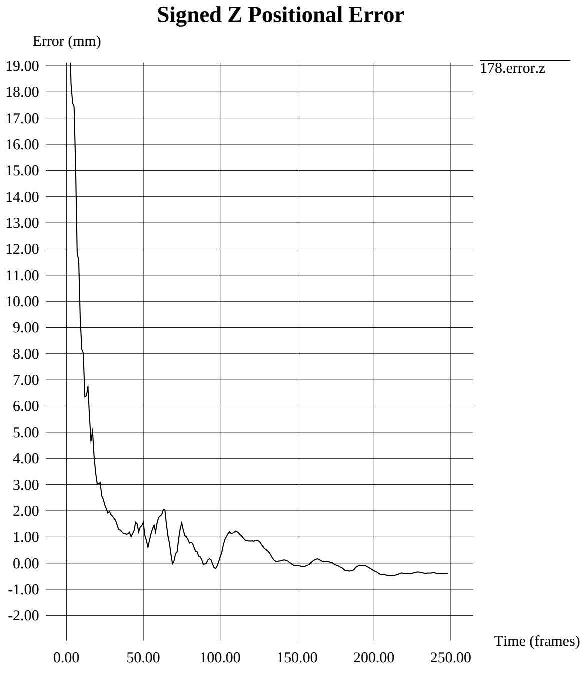

Figure 4.5: Reconstructed 3-D Positional Error in Z

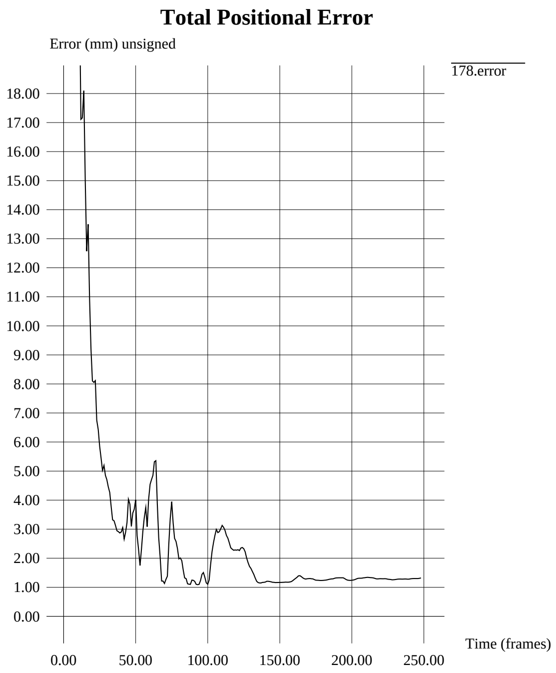

Figure 4.6: Total Reconstructed 3-D Positional Error

4.5 Generating a Mesh

Once features seen by the camera are known accurately enough (their covariance matrices are sufficiently small), they are used as vertices in a triangular mesh to define a surface. The simplest surface definition is a 2-D Delunnay triangulation of the 3-D points from the point of view of the camera. This helps avoid introducing erroneous folds in the reconstructed surface. This is arguably not the best or most accurate surface reconstruction possible. A simple modification is to avoid tessellating large triangles which can prevent construction of some false geometry, particularly around the borders of the reconstructed surface [Kanade, 1996a]. Other more involved methods for interpolating surface data include using multiresolution wavelet transforms [Yaou and Chang, 1994], grid-based hierarchies [Terzopoulos, 1988], surface patches [Liao and Medioni, 1996], and constrained Delunnay triangulation. A simple 2-D Delunnay triangulation approximation is reasonable in certain environments and is sufficient for certain applications. It can be computed in real time incrementally or by $O(n \log n)$ algorithms.

4.6 Correcting Camera Position

Using precalibrated features (fiducials) to correct for errors in camera pose has already been demonstrated [Bajura and Neumann, 1995b] [State et al., 1996a]. The contribution here is to use features which are not precalibrated to correct camera pose. This is possible under the assumption that the camera's error is normally distributed over time. The filtering process which reconstructs 3-D feature positions in some sense averages out the camera's error. In theory this means that computing camera pose from the 3-D positions of well-known features is potentially more accurate than the camera pose as reported by the tracking arm. This idea is explored more generally by [Shekhar and Chellappa, 1992].

A method for computing camera pose from a set of fiducials is presented in appendix A. Assuming enough accurate feature positions are visible, camera pose can be computed directly by equation A.15. Alternately the estimated pose from the tracking system can be used as a point for minimization by equation A.9. This approach is not ideal however. The minimization for camera position should only be carried out over the space of extrinsic camera parameters (translations and rotations) and the other characteristics of the camera should remain fixed. This approach is also not ideal from the viewpoint of signal processing and optimally filtered estimation. It minimizes the error seen in composite images by adding noise to the camera position estimate. State [State et al., 1996a] presents several approaches for correcting camera position depending on the number of features visible.

This system corrects camera pose by performing rotations about the camera's center of projection to align points in a central window of each image. The computed projection of visible features is compared with their actual position in each image. A gradient descent algorithm over 3 parameters of rotation searches for an optimal rotation correction which reduces the reprojection error. Figure 4.7 shows the average error correction/feature (in pixels) for the experimental sequence described in section 4.9.

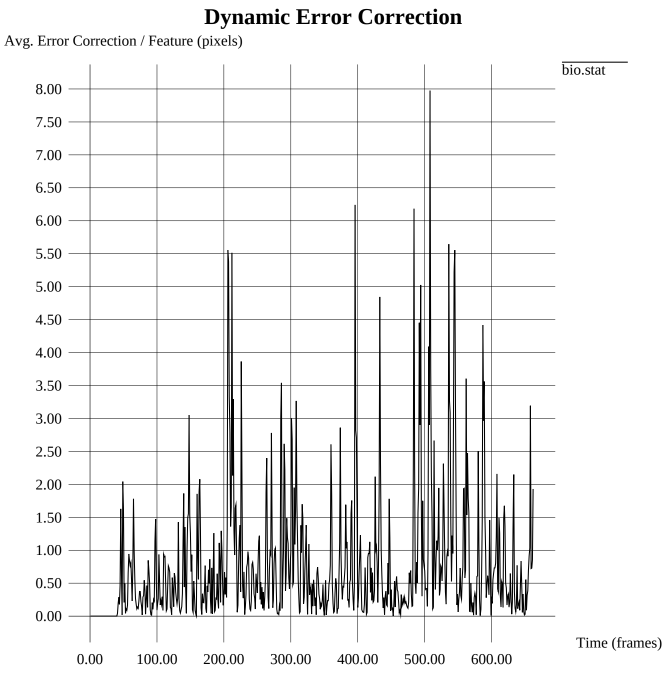

Figure 4.7: Average Error Correction / Feature by Rotating Camera

4.7 Rendering

Once a mesh has been generated to represent surfaces in the real environment, rendering composite real and synthetic images is relatively straightforward. Synthetic images using the current (or corrected) camera position and camera model are rendered using the real surface mesh to obscure synthetic objects. Composite images are then formed from the corresponding real and synthetic images pixel by pixel. Pixels are copied from the synthetic image if a synthetic object is visible and not obscured by the surface mesh. Otherwise pixels are copied directly from the real image. Kanade [Kanade, 1996b] calls this operation z-keying, after the chroma-keying technique currently used in video production.

There are two alternatives for rendering synthetic images depending on how the camera model is applied:

1. Ray-trace the synthetic images with the complete camera model. Apply the inverse distortion function for non linear projection and the linear projection matrix to produce a rays for ray tracing.

2. Render the synthetic images with the linear-projection matrix first and then interpolate either the real image or the synthetic image with the non linear-projection distortion function or its inverse before fusing the two images.

There are tradeoffs with each of these approaches. The advantage of the second interpolation approach is that rendering can optimized around a linear projection model, for example a scan line rendering algorithm or most hardware graphics accelerators. The disadvantage is that care must be taken when resampling either the real or synthetic images. Resampling without interpolation (pixel-to-pixel mapping) can cause aliasing which is particularly apparent during movement. The aliasing problem is more pronounced in more highly distorting cameras which can generate larger bands of one-to-many pixel mappings. The ability to render images with a strictly linear projection camera model and the ability to warp images (pixel-to-pixel) has been well demonstrated in commercial products.

The advantage of the ray tracing approach is that the resampling problem is implicitly addressed during ray tracing. The disadvantage is that ray tracing algorithms typically do not execute in real-time.

4.8 Computational Cost

This section outlines the computational cost of the algorithm presented in this chapter for recovering scene geometry. The cost can be divided into three parts: 1) image processing and feature point identification, 2) feature point matching, optimal filtering, surface reconstruction and simulation, and 3) rendering and compositing final images. Each cost is described below with an analysis of needs for a real-time implementation

4.8.1 Image Processing and Feature Point Identification

Feature points are identified as peaks in the simplified metric function (equation 4.7). The cost of computing this function is the cost of computing its underlying scaled derivative functions plus the cost of computing the function itself.

Differentiation is implemented as a two-pass recursive filter by Deriche [Deriche, 1990] which very nearly approximates a derivative of gaussian convolution kernel. The filter is applied twice to compute second order derivatives. Execution of the filter function requires a constant number of operations per output pixel independent of gaussian kernel and image sizes. 16 multiplications and 14 additions per pixel are required for each pass of the filter.

Computing the simple metric function requires 5 applications of the filter, followed by 2 multiplies and 1 addition for a total of 153 operations per pixel.

$$(16 \times 5 + 2 = 82 \text{ multiplies}) + (14 * 5 + 1 = 71 \text{ additions}) = 153 \frac{\text{operations}}{\text{pixel}}. \qquad (4.22)$$

For a 640×480 image, 47 M operations are required:

$$153 \frac{\text{operations}}{\text{pixel}} \times 640 \times 480 = 47 \frac{\text{M operations}}{\text{image}}. \qquad (4.23)$$

Another way to evaluate this cost is to count the number of times the original image or a copy of it must be read from and written to main memory. Assuming the entire image is larger than the local CPU cache, but that one row or column of pixels can remain in the CPU cache during processing, each filtering operation requires 2 reads and 2 writes of an entire image (one read and write for each of the vertical and horizontal passes). For an image of size 640×480 represented as a 32-bit floating point array, computing the simplified metric function translates into a total memory transfer of 29 MB per image:

$$640 \times 480 \times [(2 \frac{\text{reads}}{\text{filter}} + 2 \frac{\text{writes}}{\text{filter}}) \times 5 \text{ filters} + 3 \text{ reads} + 1 \text{ write}] \times 4 \frac{\text{bytes}}{\text{pixel}} = 29 \frac{\text{MB}}{\text{image}}. \qquad (4.24)$$

Peaks are located by identifying local horizontal peaks while computing the final pass of the simplified metric function image. These points are used as seeds to start an uphill search for local peaks. As the search proceeds, points along the search paths are marked to accelerate future searches which might follow the same path to the same terminal peak. The cost of computing peaks is relatively small. The initial seed points can be identified and saved during computation of the simplified metric function. The amount of searching depends on the number of local peaks to be found, but it is a relatively one dimensional search compared to the two dimensional cost of computing the simplified metric function.

On a single 200 MHz MIPS R4400 processor + R4010 Floating point unit in an SGI ONYX system, image filtering and peak identification takes just over 1 second for a 645×485 image, including conversion from 32-bit binary RGB to 32-bit floating point. This is consistent with the observed performance of approximately 30 MB/sec transfer between the central processing unit(CPU) and memory. This is roughly a factor of 40 slower than what is needed for real-time performance.

Real-time performance without image subsampling could be achieved by:

- Image space parallelization:

  Computation of the simplified metric function and identification of peaks is relatively local according to filter kernel size. Computation costs vary directly with the size of an image or subimage computed. The SGI ONYX system can support up to 36 processors except the system bus tends to saturate when more than 16 processors are utilized. On the SGI ONYX system, a conservative factor of 10× speedup could be achieved. This prediction is aided by the probability of more cache hits during processing because each processor would be working on a smaller subimage than the current single processor per image timing measurement.

- Faster processing:

  Optimizing the current C++ code by hand coding its inner loops in assembly language and/or upgrading to faster processors. This could account for as much as a factor 4× speedup. 250 Mhz R4400 processors are currently available with expectations of faster processors to come [1] . The estimates for operation counts or memory transfer predict a factor of 2 to 4 better performance than is achieved with the current implementation.

The summary is that feature point identification is currently an interactive-rate process which can be or will soon be performed in real-time.

4.8.2 Feature Point Matching, Optimal Filtering, Surface Reconstruction, and Simulation

Matching feature points in one image with those from a previous image can be performed in approximately $O(n \log n)$ time by sorting the new and old feature points in one dimension before comparing them. Points which don't have a match within a given radius remain unmatched. Another approach which yields good results is to sort the feature points into bins and compare adjacent bins for matches. Typical images contain approximately 200 feature points. Point matching is feasible in real-time on current architectures.

Optimal filtering for each point requires a small number of 3×3 matrix operations: 2 additions, 2 multiplies, and 1 inversion. Real-time update of 200 points at 30 Hz (= 6000 updates/sec) is well within the computational power of a modern 200 MHz processor.

[1] Digital Equipment Corporation's Alpha processor is available at 500 Mhz[Digital, 1997]

Surface generation by Delunnay triangulation is computed in $O(n \log n)$ time, where n is the number of points in the triangulation [O'Rourke, 1994] [Fortune, 1987].

Interactions with the computed surface (approximately $n = 200$ triangles) can be simulated in real-time by ray-tracing trajectories to intersect the surface and computing cut-plane intersections. Both ray intersections and cut-plane intersections take at most $O(n)$ time to compute and can be accelerated to $O(\log n)$ by tree hierarchies.

4.8.3 Rendering

Real-time rendering of polygonal surfaces, image warping, and image composition is a commercial reality on many hardware platforms.

4.9 Example Application

The example application is medical visualization of virtual ultrasound data "inside" a patient. The envisioned procedure is breast biopsy where a real-time ultrasound scanner is used to guide insertion of a needle to get a tissue sample from a woman's breast. The surface (skin) of the patient is detected by tracking and locating points on the patient's abdomen. These points are used to define a surface into which virtual holes are cut to display ultrasound visualization data. The holes provide a depth cue for the data which improves the 3-D perception of the ultrasound data actually being inside the patient instead of appearing to be in front of her.

The source image sequence is 20 seconds in length and features the camera panning over the surface of the patient's abdomen. Figure 4.8 is a frame from this sequence. Other frames view the same surface from different positions and orientations as the camera moves and pans over a 30 cm range. Figure 4.9 is the same image after correction for lens distortion. The difference from the source image is slight but is several pixels in places. Figure 4.10 shows the reconstructed surface reprojected onto the distortion-corrected source image. It appears to be a 2-D triangulation of the detected features, which it is, but is also a true 3-D representation which is consistent with this camera view. It took just over 100 frames (3.3 seconds of video) to identify and locate points on the surface accurately enough to define a virtual surface model. This convergence rate depended mostly on the motion of the camera and not on the reconstruction method. Figure 4.11 shows a simulated transducer sweeping the reconstructed surface with a virtual hole cut into it to display ultrasound data inside the patient. The registration of the virtual hole in the final figure is corrected by modifying the orientation of the virtual camera based on the computed positions of visible features. The motion of the transducer is simulated by computing intersection points with the virtual surface in each frame. This visualization is not possible without applying the techniques in this chapter.

Figure 4.8: Source Image (frame 262)

Figure 4.9: Distortion-corrected Source Image (frame 262)

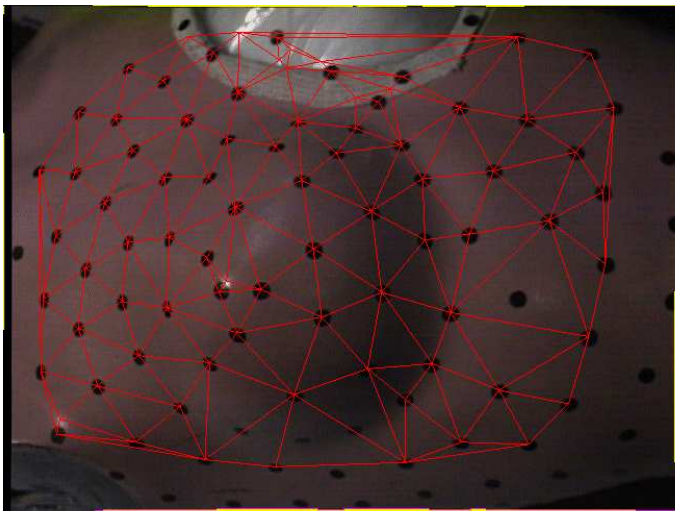

Figure 4.10: Wire-frame Surface Reconstruction (frame 262)

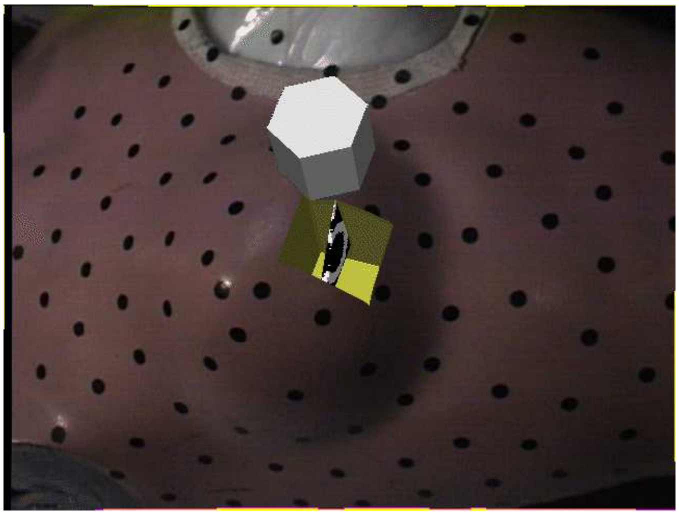

Figure 4.11: Final Image with Virtual Ultrasound Transducer and Synthetic Hole (frame 262)

Chapter 5

Conclusions and Future Directions

5.1 Conclusions

The two main conclusions of this dissertation closely follow the contributions outlined in Chapter 1:

1. Future AR systems will use optical feedback to achieve accurate image registration. The current "open loop" model for AR system design is a brute force attempt to perfect image registration by perfecting 3-D registration with perfect system components which either have no delay or are able to predict the future. This cannot work. The finesse or "closed loop" approach is to measure and correct for image registration errors directly — closing the loop between the processes causing 3-D registration errors and their effect on composite images. This represents a change in thinking by emphasizing minimizing 2-D image registration errors instead of the 3-D registration errors which caused them. This approach yields results which have not been achieved with other methods.

2. It is nearly possible to use simple real-time computer vision algorithms to detect depth in AR systems and to improve the AR experience by providing occlusion and interaction cues. Uncalibrated natural surface features can also be used as fiducial marks to help correct image registration errors. As technology continues to improve, more complex vision algorithms will be possible. A fairly large body of research exists in the computer vision area, but unfortunately many of the results are achieved with human intervention (they not automatic) and hard numbers for 3-D reconstruction accuracy are not presented. The ability to correct for image registration errors depends on the ability of computer vision algorithms to measure them.

A final observation is that AR systems require a surprisingly higher level of precision and calibration than corresponding strictly VR systems. Errors of position or delay are often easily tolerated in a VR system because they result in relative errors for which a user naturally and unthinkingly compensates. In an AR system, these same errors cause absolute image registration errors which quickly destroy the illusion of a fused reality. Even with the "closed loop" model, calibration is still a difficult and important part of any working system.

5.2 Future Directions

The next step is to think carefully about how automatic 3-D scene reconstruction and camera positioning fits in with the current notion of a "tracking system". The ultimate goal for a tracking system is to operate in an unknown environment strictly from visual input: Gary Bishop's Self-Tracker [Bishop, 1984]. The ultimate goal of depth recovery is to accurately recover both an unknown scene's structure and a camera's motion within it solely from camera images (the structure-from-motion problem). Neither of these problems has been solved well enough yet to be the basis of a working AR system, however even partial solutions to these problems (using information from a tracking system) may be able to relax the requirements for tracking systems as we currently think of them and/or push the design of AR systems in new directions.

Appendix A

Video Camera Calibration and Results for Augmented-Reality Systems

A.1 Abstract

This appendix presents an end-to-end camera model which consists of separate non linear-projection and linear-projection (perspective-projection) functions. The advantage of this approach is that each function can be found independently yielding a simpler solution method than with other combined approaches. This method achieves a standard deviation error of 0.45 mm for a camera positioned by a mechanical tracking arm accurate to 0.23 mm standard deviation error. A factorization of this model is presented which recovers camera parameters and residual image corrections needed to register real and synthetic images for augmented-reality (AR) applications. Because the non perspective-projection function can be estimated independently of a camera's intrinsic and extrinsic parameters, this idea can be extended to other calibraition methods which only model perspective (linear) projection.

A.2 Introduction

The goal of camera calibration is to establish an accurate correspondence between points in a camera's image plane and rays in space. This correspondence is generally not a linear-projection relationship although a linear-projection (perspective-projection) model has proven adequate for many applications. A linear (perspective) projection is a non-linear mapping function which can be described by projection through a single point. A non linear-projection mapping is defined as a non-linear mapping which cannot be described by projection through a single point. The linear-projection model often fails in camera systems which produce quantitative measurements or which have distorting lenses which can not be modeled well by linear projection only.

The camera calibration model described here consists of a linear-projection function (perspective projection) composed with a function which cannot be described by a linear projection. This decomposition is advantageous because each function can be estimated separately and more accurately than with a combined model.

Section A.3 reviews other models for camera calibration and the problems with them. Section A.4 explains a new model for separating a non linear(perspective)-projection image distortion function from a linear(perspective)-projection function, how to extract various camera parameters from the model, and how this model can be adapted to other linear-projection calibration methods. Section A.5 describes an implementation and results of this method using a camera attached to a mechanical tracking arm. Conclusions follow in section A.6. Notes analyzing the error induced by the model and error in localizing fiducial marks follow in sections A.7 and A.8 respectively.

A.3 Standard Solution Method

Camera calibration is generally accomplished by locating features in images produced by the camera to be calibrated. There are essentially two different approaches for finding camera parameters in this way. Both approaches assume the various camera adjustments such as focal length and aperature remain constant. The first approach assumes that the 3-D (X, Y, Z) world-space positions of features are known and that they can be matched with their 2-D (U, V) image space positions in the camera's image plane. This creates a set of (X, Y, Z, U, V) 5-tuples which are used as input to a calibration algorithm. The second approach assumes we don't know the 3-D (X, Y, Z) positions of features but that we know the correspondence of features in two (or more in some cases) images taken from different camera positions. This creates a set of $(((U_0, V_0), (U_1, V_1))$ 4-tuples which are input to an algorithm. In addition to computing parameters for a camera model, algorithms based on 4-tuple data must also implicitly solve for the unknown relative 3-D (X, Y, Z) spatial positions of each feature. To date 4-tuple methods have not incorporated lens distortion models [Faugeras et al., 1992] [Liu et al., 1990] [Luong and Faugeras, 1993] [Maybank and Faugeras, 1992] [Shashua, 1994].

Several authors have studied methods of camera modeling based on the (X, Y, Z, U, V) 5-tuple approach. Sutherland [Sutherland, 1974] describes a linear-projection (non-distorting) camera model which can be computed as the solution to a linear system of equations. Ganapathy [Ganapathy, 1984a] shows how this transformation can be decomposed into camera parameters such as position, orientation, and field-of-view but does not account for general projective distortions.

Tsai [Tsai, 1987] and Weng [Weng et al., 1992] add non-linear distortion terms and non-linear minimization to essentially the same equations used by Sutherland [Sutherland, 1974]. The problem with these later approaches is that they fail to separate the lens distortion function from the inherent linear-projection function of a camera system. This makes it difficult to know whether their non-linear minimization solutions are the correct ones. Often the form of the distortion is modeled as a symmetric function at a particular image location, which may or may not be the same as the projective center point[Willson and Shafer, 1993]. Wei [Wei and Ma, 1993] and Qiu [Qiu and Ma, 1995] use more general models for lens distortion similar to the method described here. We review the standard 5-tuple approach and its problems by starting with the linear-projection model and adding non linear-projection terms to it.

Linear projection from $\Re^3 \rightarrow \Re^2$ without distortion can be expressed elegantly in homogeneous coordinates [1] by a 3×4 projection matrix. An example of a linear projection matrix which transforms 3-D homogeneous coordinates $[X,Y,Z,W]^T$ into 2-D homogeneous coordinates $[U,V,W']^T$ is:

$$\begin{bmatrix} lp_{1,1} & lp_{1,2} & lp_{1,3} & lp_{1,4} \\ lp_{2,1} & lp_{2,2} & lp_{2,3} & lp_{2,4} \\ lp_{3,1} & lp_{3,2} & lp_{3,3} & lp_{3,4} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = LP_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} U \\ V \\ W' \end{bmatrix} \tag{A.1}$$

With enough (X,Y,Z,U,V) 5-tuples, we can solve for the terms of $LP_{3\times4}$. Writing out A.1 in non-homogeneous coordinates $(W'=1)$ we have:

$$U = \frac{lp_{1,1}X + lp_{1,2}Y + lp_{1,3}Z + lp_{1,4}}{lp_{3,1}X + lp_{3,2}Y + lp_{3,3}Z + lp_{3,4}} \tag{A.2}$$

$$V = \frac{lp_{2,1}X + lp_{2,2}Y + lp_{2,3}Z + lp_{2,4}}{lp_{3,1}X + lp_{3,2}Y + lp_{3,3}Z + lp_{3,4}} \tag{A.3}$$

We can write linear equations for the terms of $LP_{3\times4}$ by multiplying equations A.2 and A.3 by their denominators:

$$X\ lp_{1,1} + Y\ lp_{1,2} + Z\ lp_{1,3} + lp_{1,4} - (U\ X\ lp_{3,1} + U\ Y\ lp_{3,2} + U\ Z\ lp_{3,3} + U\ lp_{3,4}) = 0 \tag{A.4}$$

$$X\ lp_{2,1} + Y\ lp_{2,2} + Z\ lp_{2,3} + lp_{2,4} - (V\ X\ lp_{3,1} + V\ Y\ lp_{3,2} + V\ Z\ lp_{3,3} + V\ lp_{3,4}) = 0. \tag{A.5}$$

With at least 8 separate (X,Y,Z,U,V) data points, $LP_{3\times4}$ can be found by directly by least squares methods from equations A.4 and A.5 [Sutherland, 1974]. Note that this defines $LP_{3\times4}$ only up to a scale factor because of division by W' (we set $W'=1$ to write out equations A.2 and A.3).

[1] Homogeneous, or projective, coordinates are the same as regular coordinates with the addition of a projective component W. For example, a 3-D homogeneous coordinate is expressed by a 4-tuple $[X,Y,Z,W]^T$. To find the regular coordinate value of a homogeneous coordinate, we divide each element of a homogeneous coordinate vector by W. For example: Given the homogeneous 3-D coordinate $[X=2,Y=4,Z=6,W=2]^T$, the corresponding 3-D coordinate is $[X=1,Y=2,Z=3]^T$. The homogeneous coordinate of any 3-D coordinate is $[X,Y,Z,W=1]^T$.

When there is non-linear projective distortion in the lens system, equations A.2 and A.3 are modified by adding image-space distortion functions [Tsai, 1987] [Weng et al., 1992]:

$$U + f_U(U,V) = \frac{lp_{1,1}X + lp_{1,2}Y + lp_{1,3}Z + lp_{1,4}}{lp_{3,1}X + lp_{3,2}Y + lp_{3,3}Z + lp3,4} \tag{A.6}$$

$$V + f_V(U,V) = \frac{lp_{2,1}X + lp_{2,2}Y + lp_{2,3}Z + lp_{2,4}}{lp_{3,1}X + lp_{3,2}Y + lp_{3,3}Z + lp3,4} \tag{A.7}$$

where $f_U()$ and $f_V()$ are non-linear functions of U and V. Solving these equations given a data set of (X,Y,Z,U,V) 5-tuples is difficult. Typically the linear solution with $f_U() = 0$ and $f_V() = 0$ is used as a seed for a non-linear minimization algorithm. Non-linear minimization then varies the parameters of $f_U()$, $f_V()$, and $LP_{3\times 4}$ in hopes of finding a global minimum. The form of the error function used for non-linear minimization is:

$$\begin{aligned} \text{Error} = {} & \left(\textstyle\sum_{i=1}^{\text{\#of Data points}} \text{Image\_Plane\_Error}(\text{Data point}_i, LP_{3\times 4}, f_U(), f_V())\right) + \\ & \text{Constraint\_Function}(LP_{3\times 4}, f_U(), f_V()). \end{aligned} \tag{A.8}$$

Picking an appropriate constraint function is difficult because the distortion parameters are not separate from the projection parameters. Because of the complex function being minimized, it's also hard to tell whether the minimization result is the true minimum or simply a local one.

There is also a problem in finding $LP_{3\times 4}$ by solving a least-squares system on equations A.4 and A.5: the data points are unevenly weighted by their projective components. In transforming equations A.2 and A.3 into equations A.4 and A.5 we multiplied by the denominator which roughly represents the distance between a point and the camera. Data points which are further away from the camera are multiplied by a larger value than data points which are closer to the camera. This effectively gives more weight to far points than to near points. This is not desirable because the solution to the camera calibration ought to be independent of the coordinate system used for test data. In spite of its problems, non-linear minimization of equation A.8 doesn't unevely weight different data points and suggests that the best way to find $LP_{3\times 4}$ in the linear case is to use a least-squares method to get a close solution and then use non-linear minimization on an equation of the form(similar to A.8):

$$\text{Error} = \sum_{i=1}^{\text{\#of Data points}} \text{Image\_Plane\_Error}(\text{Data point}_i, LP_{3\times 4}) \tag{A.9}$$

There is still a small problem with this however, the solution is unfairly biased toward the larger densities of data in the image plane.

A.4 New Camera Model and Solution Method

The problem with the previous models is that they don't separate the function of non linear-projection(distortion) from linear projection. This model separates non linear projection from linear projection by modeling the camera system as a non linear-projection function composed with a strictly linear-projection function. This model is derived by first starting with the linear projection camera model and seeing how distortion changes it. Next from this model we show how to derive camera position and orientation, how to reproject a distorted image into an undistorted one, and how to project undistorted images into distorted ones. [2]

A.4.1 Deriving the Model

Lemma 1 *The composition of an $\Re^2 \to \Re^2$ linear projection with an $\Re^3 \to \Re^2$ linear projection is simply another $\Re^3 \to \Re^2$ linear projection:*

To see this, recall how to express a linear projection from 3-D homogeneous coordinates into 2-D homogeneous coordinates with a 3×4 linear projection matrix [Mundy and Zisserman, 1992]:

$$LP_{3\times4}\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} lp_{1,1} & lp_{1,2} & lp_{1,3} & lp_{1,4} \\ lp_{2,1} & lp_{2,2} & lp_{2,3} & lp_{2,4} \\ lp_{3,1} & lp_{3,2} & lp_{3,3} & lp_{3,4} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \to \begin{bmatrix} U \\ V \\ W' \end{bmatrix} \tag{A.10}$$

Recall also that a projection from one 2-D image to another 2-D image can be expressed (also in homogeneous coordinates) by a 3×3 linear projection matrix:

$$PR_{3\times3}\begin{bmatrix} U \\ V \\ W' \end{bmatrix} = \begin{bmatrix} pr_{1,1} & pr_{1,2} & pr_{1,3} \\ pr_{2,1} & pr_{2,2} & pr_{2,3} \\ pr_{3,1} & pr_{3,2} & pr_{3,3} \end{bmatrix}\begin{bmatrix} U \\ V \\ W' \end{bmatrix} \to \begin{bmatrix} U' \\ V' \\ W'' \end{bmatrix} \tag{A.11}$$

By matrix multiplication of equations A.10 and A.11, it's clear that the composition of an $\Re^2 \to \Re^2$ linear projection with an $\Re^3 \to \Re^2$ linear projection is also another $\Re^3 \to \Re^2$ linear projection:

$$PR_{3\times3} \circ LP_{3\times4}\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = LP'_{3\times4}\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \to \begin{bmatrix} U' \\ V' \\ W'' \end{bmatrix}. \tag{A.12}$$

By lemma 1, when modeling a camera system by an $\Re^3 \to \Re^2$ linear projection, it doesn't make sense to model $\Re^2 \to \Re^2$ linear reprojections of camera images separately because the composition of an $\Re^2 \to \Re^2$ linear projection with an $\Re^3 \to \Re^2$ linear projection reduces to another $\Re^3 \to \Re^2$ linear projection. If a camera system is not modeled well by linear projection the situation is different.

In the distorting camera case we can think of image formation as a linear projection $LP_{3\times4}$ onto an intermediate image plane followed by a non linear-projection $PR_{nonlinear}$ onto a final image

[2] Distorted images are produced by a non linear-projection projection function and undistorted images are produced by a linear (perspective) projection function.

plane. The result no longer reduces to another linear projection:

$$PR_{nonlinear} \circ LP_{3\times4} = \text{An ugly non-linear function.} \tag{A.13}$$

However if we can define $PR_{nonlinear}$ as a function which maps an intermediate linear projection plane into the actual distorted image plane accounting for all non linear-projection distortions, we're still in business. By applying $PR^{-1}_{nonlinear}$ to $[U, V, W]^T$, we again have an $\Re^3 \rightarrow \Re^2$ linear projection of the undistorted image points:

$$PR_{nonlinear} \circ LP_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} U \\ V \\ W \end{bmatrix}, \text{ or} \tag{A.14}$$

$$LP_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = PR^{-1}_{nonlinear} \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{A.15}$$

It remains to be shown how to find the $LP_{3\times4}$ and $PR_{nonlinear}$ functions. Assuming $PR^{-1}_{nonlinear}$ is known, $LP_{3\times4}$ can be found from (X, Y, Z, U, V) sample data by applying $PR^{-1}_{nonlinear}$ to each (U, V) sample and finding a least squares solution to equations A.4 and A.5. $LP_{3\times4}$ can then be more accurately computed by minimizing equation A.9 so that far data points are not unevenly weighted. Because $PR^{-1}_{nonlinear}$ accounts for distortions which are not linear projections, there isn't as much danger as with other methods of finding an incorrect minimum for $LP_{3\times4}$.

$PR_{nonlinear}$ can be found by imaging a registered plane with the camera system. The mapping from the registered plane to the camera's physical image plane is the function $PR_{nonlinear}$. In the case of a perfect linearly projecting camera, this relationship could be expressed as a 3×3 matrix $PR_{3\times3}$. In the imperfect case, this mapping is a general function which can be approximated by a 3×3 matrix which can be exact at at least 4 points in the image plane. The function $PR^{-1}_{nonlinear}$ can be reconstructed by imaging fiducial marks(samples) in a registered plane and interpolating the $\Re^2 \rightarrow \Re^2$ mapping between the fiducial coordinate system and the camera image coordinate system. The coordinates for the registered plane are arbitrary and are recovered by factorizing the resulting solution to $LP_{3\times4}$ later on.

This completes the camera model. The next section explains how to extract various camera parameters, distortion and undistortion functions, and projection functions from the model.

A.4.2 Working with the Model

This section shows how to extract different parameters and functions from the model. This is done by factoring a camera projection matrix $CP_{3\times4}$ to expose camera parameters and by grouping elements of this factorization to define distortion, undistortion, and projection functions. After this, simpler camera models which omit certain camera parameters or distortion functions can be evaluated by examining the magnitude of those parameters in this more general model.

The first step is to define $PR^{-1}_{3\times 3}$, an $\Re^2 \rightarrow \Re^2$ linear projection function, which approximates $PR^{-1}_{nonlinear}$:

$$PR^{-1}_{3\times 3} \simeq PR^{-1}_{nonlinear}. \tag{A.16}$$

Next we define $CD_{nonlinear}$ as the residual between $PR^{-1}_{3\times 3}$ and $PR^{-1}_{nonlinear}$. $CD_{nonlinear}$ represents the component of camera distortion which is not represented by strictly linear projection:

$$CD_{nonlinear} = PR_{nonlinear} \circ PR^{-1}_{3\times 3}. \tag{A.17}$$

In a perfectly linearly projecting camera, $CD_{nonlinear}$ would be the identity function. Depending on how $PR^{-1}_{3\times 3}$ is defined, we'll get slightly different answers for $CD_{nonlinear}$, and also camera skew and orientation later on.

The camera projection matrix, $CP_{3\times 4}$, is defined as the composition of the matrix $PR_{3\times 3}$ (defined by $PR^{-1}_{3\times 3}$ above) and the original $LP_{3\times 4}$ matrix. This grouping of functions does not change the original camera model:

$$\begin{aligned} CP_{3\times 4} &= PR_{3\times 3} \circ LP_{3\times 4}, \text{ so that} & \text{(A.18)} \\ CD_{nonlinear} \circ CP_{3\times 4} &= PR_{nonlinear} \circ PR^{-1}_{3\times 3} \circ PR_{3\times 3} \circ LP_{3\times 4} & \text{(A.19)} \\ &= PR_{nonlinear} \circ LP_{3\times 4} & \text{(A.20)} \\ &= \text{Original Model.} & \text{(A.21)} \end{aligned}$$

Camera position, orientation, center-of-projection, field-of-view, and skew are recovered by factoring the $CP_{3\times 4}$ matrix.

Defining the camera distortion function $CD_{nonlinear}$

$CD_{nonlinear}$ is defined as the function which maps undistorted camera images into distorted ones. Ideally, this is done by finding the linear projection function which most nearly approximates $PR_{nonlinear}$ and applying it's inverse (A.17). This could be found by minimizing the squared error between $PR_{nonlinear}$ and its linear projection approximation, $PR_{3\times 3}$, over the camera's image plane:

$$PR_{3\times 3} : \int\int_{\text{Image Plane}} (PR_{nonlinear}(u,v) - PR_{3\times 3}(u,v))^2 du dv = \min. \tag{A.22}$$

This is a difficult task in general because $PR_{nonlinear}$ could be any general function and this minimization could have more than one local minimum. However it turns out any reasonable approximation is sufficient for the camera model because the definitions for $CD_{nonlinear}$ and $CP_{3\times 4}$ do not change the overall model (equation A.19). Differences in approximating the $CD_{nonlinear}$ function affect only the computed orientation of the camera, not its position.

An arbitrary but workable approach is to fit a function $PR_{3\times3}$ which is exact for 4 points in the image plane, for example the corners of the camera's image. The mapping of 4 points defines an $\Re^2 \rightarrow \Re^2$ linear projection which can be found by a matrix solution. The equation to solve is:

$$PR_{3\times3}\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} pr_{1,1} & pr_{1,2} & pr_{1,3} \\ pr_{2,1} & pr_{2,2} & pr_{2,3} \\ pr_{3,1} & pr_{3,2} & pr_{3,3} \end{bmatrix}\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} U' \\ V' \\ W' \end{bmatrix} \tag{A.23}$$

Where $[U, V, W = 1]^T$ is the set of points in the registered plane and $[U', V', W' = 1]^T$ is the mapping by $PR_{nonlinear}$ of those points in the camera's image plane. Similar to equations A.4 and A.5 we can write out:

$$U\ pr_{1,1} + V\ pr_{1,2} + pr_{1,3} - (U'\ U\ pr_{3,1} + U'\ V\ pr_{3,2} + +U' pr_{3,3}) = 0 \tag{A.24}$$

$$U\ pr_{2,1} + V\ pr_{2,2} + pr_{2,3} - (V'\ U\ pr_{3,1} + V'\ V\ pr_{3,2} + +V' pr_{3,3}) = 0 \tag{A.25}$$

With 4 points this gives 8 equations in 9 unknowns. But since we can only determine a projection matrix up to a scale factor there are really 8 unknowns. In most cases, unless there is extreme perspective distortion, we can safely assume $pr_{3,3} = 1$ and solve for the remaining terms of $PR^{-1}_{3\times3}$. This defines the camera distortion function $CD_{nonlinear}$ by equation A.17.

Note that definition of $CD_{nonlinear}$ only requires only that the function $PR_{nonlinear}$ be defined. This is not yet dependent on any information about the camera. This could be a useful function to correct for errors in camera calibration approaches which assume projective distortion only, particularly those based on 4-tuple data [Faugeras et al., 1992] [Liu et al., 1990] [Luong and Faugeras, 1993] [Maybank and Faugeras, 1992] [Shashua, 1994]. This function could also be used to map distorted images into undistorted ones for registration correction in augmented-reality systems.

Factoring the camera projection matrix $CP_{3\times4}$

By factoring the $CP_{3\times4}$ matrix into the product of a 3×3 skew matrix, $SKEW_{3\times3}$, an orthonormal 3×3 rotation matrix, $ROT_{3\times3}$, and a 3×4 projection matrix $PROJ_{3\times4}$, camera position, orientation, field-of-view, center-of-projection, and image skew are recovered. The factorization is:

$$CP_{3\times4} = SKEW_{3\times3} \circ ROT_{3\times3} \circ PROJ_{3\times4} \tag{A.26}$$

where:

$$SKEW_{3\times3} = \begin{bmatrix} s_u & s_{uv} & u_{cp} \\ 0 & s_v & v_{cp} \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.27}$$

$$ROT_{3\times3} = \begin{bmatrix} - & x_3 & - \\ - & y_3 & - \\ - & z_3 & - \end{bmatrix} \text{(a } 3\times3 \text{ orthonormal matrix), and} \tag{A.28}$$

$$PROJ_{3\times 4} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \end{bmatrix}. \tag{A.29}$$

$[x, y, z]^T$ is the point of projection (location) of the camera. The rows (top to bottom) of $ROT_{3\times 3}$ define the orientations of the x, y, and z viewing planes. u_{cp} and v_{cp} are the center-of-projection in *undistorted* image coordinates(coordinates in the registered plane). s_{uv} is the relative skew between the x and y image planes. The fields-of-view are given from the scaling factors s_u and s_v:

$$\text{field-of-view(u)} = \tan^{-1}\left(\frac{u_{max} - u_{min}}{2s_u}\right) \tag{A.30}$$

$$\text{field-of-view(v)} = \tan^{-1}\left(\frac{v_{max} - v_{min}}{2s_v}\right) \tag{A.31}$$

where $u_{max}, u_{min}, v_{max}$, and v_{min} are the extents of the camera image in pixels.

The factorization is performed by first factoring $CP_{3\times 4}$ into $PROJ_{3\times 4}$ and a residual($FOO_{3\times 3}$):

$$CP_{3\times 4} = FOO_{3\times 3} \circ PROJ_{3\times 4} \tag{A.32}$$

Next $FOO_{3\times 3}$ is decomposed into an orthonormal $ROT_{3\times 3}$ matrix and an upper triangular $SKEW_{3\times 3}$ matrix by QR factorization [Strang, 1988]:

$$FOO_{3\times 3} = SKEW_{3\times 3} \circ ROT_{3\times 3} \tag{A.33}$$

If the skew between the u and v image planes, s_{uv}, is a problem(u and v image planes not orthogonal), the $SKEW_{3\times 3}$ matrix can be further factored into:

$$\begin{bmatrix} s_u & s_{uv} & u_{cp} \\ 0 & s_v & v_{cp} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{s_{uv}}{s_v} & -\frac{s_{uv} v_{cp}}{s_v} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_u & 0 & u_{cp} \\ 0 & s_v & v_{cp} \\ 0 & 0 & 1 \end{bmatrix} = SKEW_{XY\ 3\times 3} \circ SKEW'_{3\times 3}. \tag{A.34}$$

$SKEW_{XY\ 3\times 3}$ can then be grouped with the $CD_{nonlinear}$ function instead of the $CP_{3\times 4}$ function. This factorization can be useful when trying to register real images with synthetic images generated with a camera model which doesn't allow for skewed viewing planes.

A.5 Experimental Results

This section describes alibration results from applying this method to a Panasonic GP-KS102 color CCD video camera with a Cosmicar 12.5 mm $f/1.8$ C-mount lens (approx. 28 degree field-of-view) rigidly attached to a FARO [Faro, 1997] 3-D digitizing mechanical tracking arm with approximately 1 m working radius. The aim of this calibration is to characterize the camera in the coordinate system of the tracking arm so that features seen by the camera can be accurately positioned in 3-D later on.

A.5.1 Method

The calibration test fixture is a planar grid of registered black dots on a white background. The grid was generated on a pen plotter which is addressable to 0.025 cm. The dots are 4 mm in diameter and are positioned on a 1.5 cm pitch. The mechanical arm is first used to measure the position of the grid in the arm's coordinate system. Next, the camera is connected to the mechanical arm and the positions of the dots are transformed back into the tracking arm's coordinate system as camera images of the grid are taken. The mechanical arm is reported to be accurate to .46 mm positional (2 sigma error) and .05 degrees (roughly) orientation error.

To measure the non linear-projection distortion properties of the camera lens, a roughly perpendicular image of the planar grid is taken. The position and orientation of the camera are not needed here. It doesn't matter if this image isn't perfectly perpendicular because the projective component of this image is recovered later. This image is used to define a mapping from camera pixel coordinates to a non-distorted 2-D coordinate system on the test fixture. The origin of the coordinate system of the test fixture is arbitrary but remains fixed. Cubic interpolation is used to interpolate pixel positions between grid points. To make the interpolation simpler, the grid image is parabolically extrapolated beyond the edge of the image so that table lookup and cubic interpolation can be used to map all points in the camera's image plane. The inverse of this mapping is found by table lookup and gradient descent minimization. The minimization converges because there are no folds in the mapping. This defines the functions $PR_{nonlinear}$ and $PR^{-1}_{nonlinear}$. For example, pixel (100,100) in the camera's image plane maps to coordinates (2.345, -1.34) in the test fixture image plane. The accuracy of this interpolation depends on the sample density in the camera image plane — the more samples the better the interpolation. The theoretical accuracy of locating dots in the camera's image plane is described below in section A.8. The positional error of grid positions due to local lens distortion and perspective distortion is assumed to be minimal.

Six (6) images of the planar grid at different distances from the camera are taken while the position of the tracking arm is recorded. The coordinates of grid dots are transformed into the tracking arm's coordinate system for each image. There are 1249 data points, less 4 outliers which were rejected after function minimization. $LP_{3\times4}$ is computed using equation A.15 by applying a least squares solution to A.4 and A.5. This solution is used as a starting point for a gradient descent search to minimize image-plane error using equation A.9.

A.5.2 Data

Figure A.1 is image used to compute $PR^{-1}_{nonlinear}$. The images used to capture positional data are similar but at differing distances from the calibration dots.

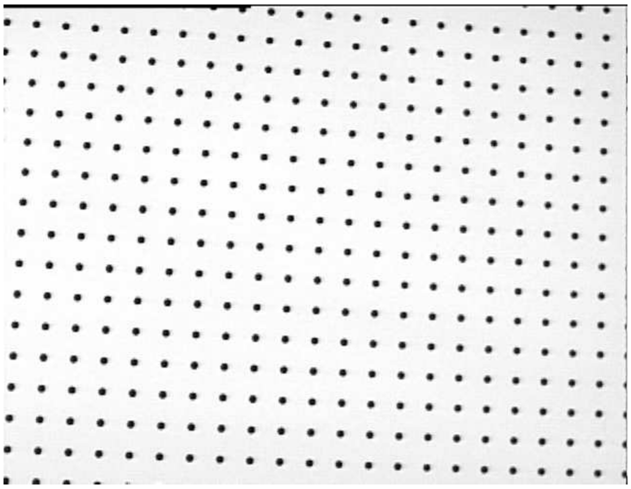

Figure A.1: Image for Distortion Table

A.5.3 Results

Two calibrations are computed. The first is the full camera model, including distortion correction. The second is the linear-projection camera model only:

	Distortion Correction	
	Yes	No
Standard Deviation (model w/calibration data)	$\sigma = 0.85$ pixels $\sigma = 0.45$ mm	$\sigma = 0.94$ pixels $\sigma = 0.50$ mm
Standard Deviation (mechanical arm)	$\sigma = 0.23$ mm	$\sigma = 0.23$ mm
Max. error (model w/calibration data)	2.28 pixels 1.21 mm	3.13 pixels 1.67 mm
Camera Position in arm coordinate system (cm) (x,y,z)	$1.90, 9.55, -1.52$	$1.90, 9.52, -1.49$

Table A.1: Calibration Results

The standard deviation error (in mm) for the distortion-correcting case is computed by reprojecting pixel-based error back into 3-D coordinate space. The camera has approximately a 28 degree field-of-view and is digitized to 640 pixels across. The far data points are approximately 70 cm from the camera.

$$\text{error} = \sin\left(28 \text{ degrees} \times \frac{0.85 \text{ pixels}}{640 \text{ pixels}}\right) \times 700 \text{ mm} = 0.45 \text{ mm}. \qquad (A.35)$$

The other distance errors are computed similarly.

The outstanding result is that this camera is calibrated for imaging points to within 1 mm at 70 cm distance. This error ($\sigma = 0.38$ mm) is approximately twice the error of the FARO mechanical arm ($\sigma = 0.23$ mm). This is about what one would expect: $\sigma = 0.23$ error in measuring the position of the calibration fixture (with the mechanical arm), and $\sigma = 0.23$ error in measuring the camera's position (also with the mechanical arm). Even though this camera is modeled quite well by a linear projection only, the calibration is improved with distortion-correction mapping.

A.6 Conclusions

This calibration method comes within the range of the back-of-the-envelope best error which could be expected with this equipment. The computed error reflects end-to-end measurements of the entire system without analyzing specific error sources like camera and frame-grabber stability.

This method for separating non linear-projection distortion from linear projection works well and could be applied to other calibration methods with distorting lenses.

A.7 Justification for Linear Projection followed by Distortion Model

The camera model consisting of linear projection through a specific point followed by a distortion function isn't exactly what happens in real lens systems. Positive (pincushion) and negative (barrel) distortion are typically caused by lenses with different magnification at different points across their face [Hecht, 1987]. Images are still in focus because an aperature limits the light on different parts of the image plane to specific areas of the lens. This effectively moves the center-of-projection either in or out along the center axis of projection for different areas of an image generating different magnification and slightly different viewpoints for different viewing angles. To first order, the distortion is a radial function because of lens symmetry.

Because camera images are not produced by projection through a single point, we estimate the effective shift of the projection point and its effect on images. For objects infinitely far from a camera, a shift of projection point is meaningless because image location is dependent on viewing angle only. In this case the differences in lens magnification can be mapped exactly by a non linear-projection distortion function. For points closer to the camera, a non linear-projection distortion function accounts only for changes in magnification due to a moving the projection point, but not for changes due to viewing from a different position. Roughly speaking, the non linear-projection distortion function can compensate for distortions (magnifications) due to moving the effective center-of-projection parallel to a particular viewing direction but not for image differences due to moving the center-of-projection perpendicular to a particular viewing direction.

Let's consider the wide angle 110 degree fish-eye lens which is an option with the Panasonic camera. It exhibits significant pincushion distortion near the edges. Estimating the radius of the CCD array to be 5 mm and the distortion to be about 20% near the edges (so that objects one would expect to see at 6 mm radius are imaged at 5 mm radius), by trigonometry we can estimate the center-of-projection shifts toward the image plane by about $\frac{1}{\tan(55°)} = 0.7$mm between on-axis angles and 55-degree off axis angles.

Next, we fix the center-of-projection at the point closest to the image plane and examine the effective lateral shift of the projection point at various angles. The error at maximum angle (55 degrees) is 0 because that's the effective projection point for off-axis viewing. The error along the central axis is also 0 because the shifted center-of-projection is co-linear with the actual center-of-projection - changing the magnification but not the view. Assuming that the center-of-projection shifts linearly with viewing angle along the lens axis, the effective lateral displacement in viewing direction is:

$$\text{lateral displacement} = l(1 - \frac{\theta}{\phi}) \sin\theta. \tag{A.36}$$

Where l is the distance the center-of-projection moves, θ is the viewing angle relative to the central axis, and ϕ is the maximum viewing angle. The peak of this function is at roughly $\theta = \frac{\phi}{2}$.

Next, compute the viewing error at a particular angle, which is a function of lateral displacement and distance from the projection point:

$$\text{error (degrees)} = \arctan\left(\frac{\text{lateral displacement}}{\text{distance}}\right) \tag{A.37}$$

In the image plane, the error exhibits a 1/distance behavior moving on radial lines from the center-of-projection, assuming that the distortion function is mapped for infinite distance. If the distortion function is mapped at finite distance, the error form resembles $a/\text{distance} - b$ which is 0 at the distance the distortion function is mapped and b at infinite distance.

Putting in some numbers, for the 110 degree lens at 27 degrees off axis, with 640 pixels across the image, and the distortion function mapped at infinity, we have:

Distance	error(degrees)	error(pixels)
30 cm	.030	.17
50 cm	.018	.10
70 cm	.013	.07
far	0	0

Table A.2: Estimated Max. Error for 110 Degree Lens (worst case)

In practice however, the distortion function is typically mapped with a fixture somewhere in the working volume. This effectively shifts the error function by a constant (the error form is $a/\text{distance} - b$). If the distortion function mapping fixture is fixed at 50 cm, the error is now:

Distance	error(degrees)	error(pixels)
30 cm	.012	.07
50 cm	0	0
70 cm	.005	.03
far	.018	.10

Table A.3: Estimated Typical Error for 110 Degree Lens (worst case)

The 28 degree field-of-view lens used in this experiment is well approximated because it has significantly less distortion than this wide angle lens by at least a factor of 10 (estimated). For the wide angle lens, the approximation is still probably not a problem. This error is a fundamental limitation of any camera model which assumes projection through a single point. This is good for working volumes which relatively are not too close to a camera. Further reducing the error for close distances would call for a more complex model than the one proposed here.

A.8 Localization of Fiducial Marks

This section discusses the theoretical accuracy of locating a fiducial point in a digitized image. It is interesting to note that this computation appears to defy the nyquist theorem by apparently reconstructing a signal to greater precision than was present in the initial signal sampling. This is not the case however, because centroid computation makes assumptions about the input signal's form (symmetry) which cannot be made when reconstructing arbitrary signals. The equations to compute a 2-D centroid after segmentation are:

$$u = \frac{\sum_{i=0}^{\#of\ Pixels} \rho_i u_i}{\sum_{i=0}^{\#of\ Pixels} \rho_i}, v = \frac{\sum_{i=0}^{\#of\ Pixels} \rho_i v_i}{\sum_{i=0}^{\#of\ Pixels} \rho_i} \tag{A.38}$$

where:

$$\rho_i = \begin{cases} 0 & \text{if } Pixel\_Value_i < T_{low} \\ 1 & \text{if } Pixel\_Value_i > T_{high} \\ \frac{Pixel\_Value_i - T_{low}}{T_{high} - T_{low}} & \text{otherwise} \end{cases} \tag{A.39}$$

$$u_i, v_i = u, v \text{ pixel coordinates.} \tag{A.40}$$

T_{low} and T_{high} are low and high segmentation thresholds respectively. T_{low} is typically set to a value high enough to disambiguate a fiducial point from its local background. T_{high} is typically set to the maximum pixel value although some authors claim that lowering this ceiling can improve noise immunity [Mellor, 1995b].

This accuracy of this centroid calculation depends on:

- The distorting properties of the imaging system. Given an arbitrary image distortion, the centroid of a distorted fiducial is not the same as the distorted position of the fiducial's actual center. In general this effect is worse for larger image-sized fiducials because they encompass a larger distortion area where the "constant distortion" approximation breaks down. An exception to this is when both the fiducial plane and image plane are parallel and there is only projective distortion.

- The stability of the imaging system. When digitizing images from a video camera, stability refers to the synchronization between the video camera and the video digitizer.

- The signal/noise ratio of the imaging system and the sharpness of edges of the centroid image. Thinking about a 1D centroid, the accuracy in locating its center is the same as that of locating its edges. Given a sampled signal, reconstructing the exact signal to find its threshold crossing is a difficult problem. The key is that we aren't interested in reconstructing the signal exactly, we just want the centroid. This means that if we reconstruct symmetrically to find the threshold, the accuracy of the centroid is roughly the same as the precision of the reconstruction, even if the reconstruction is in error. By considering the minimal curve possible for each threshold crossing, the precision of the crossing is located to within:

  $$\text{precision of threshold crossing } = \pm\frac{\text{noise magnitude}}{\text{slope at threshold}}\text{pixel}$$

  For example, for a 1D edge characterized by a slope of 40 levels/pixel with a noise uncertainty of ± 1 level, the uncertainty in finding the edge is $\pm 1/40$ pixel. The accuracy in locating a centroid between two such edges is also $\pm 1/40$ pixel.

- The size of the fiducial in image space. The estimate above is for finding a 1D centroid. Thinking of a 2D centroid calculation as the averaging of several 1D measurements (equation A.39), the larger the fiducial, the greater the number of scan-lines it covers and the more measurements there are. Assuming the 1D measurements are independent, the improvement made by averaging them goes as $1/\sqrt{\text{diameter}}$.

By this argument, ignoring image distortion and image stability, a typical centroid used in the calibration which is 8 pixels in diameter with a slope of 40 levels/pixel at its edges should be localized to within:

$$\frac{1}{40} \times \frac{1}{\sqrt{8}} = \pm.009 \text{ pixel.}$$

This suggests that errors in this calibration are not due to errors from sampling and computing each fiducial's center of mass. A greater problem could be the error induced by computing the center of mass as a discrete sum instead of an integral approximation. The bound of the maximum error falls off quickly as the width of the centroid (in pixels) grows (figure A.2). The equations describing this function are:

$$\text{actual} = \frac{\text{width}}{2} \tag{A.41}$$

$$\text{computed} = \frac{(\text{floor}(\text{width}))^2/2 + (\text{floor}(\text{width}) + 1/2)(\text{width} - \text{floor}(\text{f}))}{\text{width}} \tag{A.42}$$

$$\text{error} = \text{computed} - \text{actual}. \tag{A.43}$$

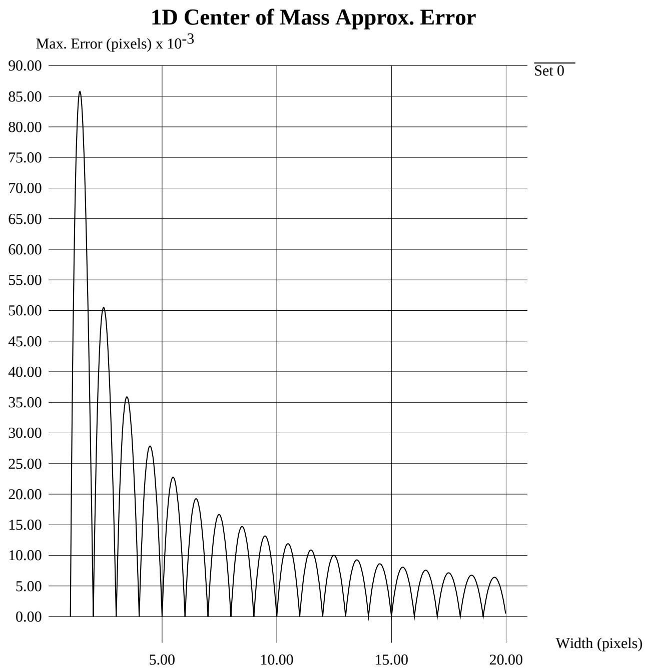

Figure A.2: Maximum error for 1D Center of Mass Computation

This result is consistent with Peuchot's [Peuchot, 1993] result of locating points to within 1% of a pixel in static 2-D plane measurements and Stone's [Stone, 1989] result that fitting a gaussian function to an (expected) gaussian symmetric measurement gives similar accuracy to computing a centroid, provided the signal levels are high enough.

Bibliography

[Adachi et al., 1995] Adachi, Y., Kumano, T., and Ogino, K. (1995). Intermediate representation for stiff virtual objects. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 203–210.

[Adelstein et al., 1992] Adelstein, B., Johnston, E., and Ellis, S. (1992). A testbed for characterizing dynamic response of virtual environment spatial sensors. In *Proceedings 5th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 15–22, Monderey, CA. ACM SIGGRAPH/SIGCHI.

[Akeley, 1993] Akeley, K. (1993). Reality engine graphics. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 109–116.

[Ascension, 1997] Ascension (1997). *Flock of Birds magnetic tracking system.* Ascension Technology Corporation, P.O. Box 527, Burlington, VT 05402, (http://www.ascension-tech.com).

[Azarbayejani and Pentland, 1994] Azarbayejani, A. and Pentland, A. (1994). Recursive estimation of motion, structure, and focal length. Technical Report Perceptual Computing Technical Report #243, Submitted to IEEE PAMI July 1994, MIT Media Laboratory.

[Azuma and Bishop, 1994] Azuma, R. and Bishop, G. (1994). Improving static and dynamic registration in an optical see-through hmd. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 197–204.

[Azuma and Bishop, 1995] Azuma, R. and Bishop, G. (1995). A frequency-domain analysis of head-motion prediction. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 401–408.

[Bajura, 1993] Bajura, M. (1993). Camera calibration for video see-through head-mounted display. Technical Report TR93-048, UNC Chapel Hill.

[Bajura et al., 1992] Bajura, M., Fuchs, H., and Ohbuchi, R. (1992). Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 203–210. Published as Proc. ACM SIGGRAPH '92, Computer Graphics, volume 26, number 2.

[Bajura and Neumann, 1995a] Bajura, M. and Neumann, U. (1995a). Dynamic registration correction in augmented-reality systems. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 189–196.

[Bajura and Neumann, 1995b] Bajura, M. and Neumann, U. (1995b). Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60.

[Besl, 1989] Besl, P. J. (1989). Active optical range imaging sensors. In Sanz, J. L. C., editor, *Advances in Machine Vision*, chapter 1, pages 1–63. Springer-Verlag, New York.

[Bhatnagar, 1993] Bhatnagar, D. K. (1993). Position trackers for head-mounted display systems: A survey. Technical Report TR93-010, University of North Carolina at Chapel Hill.

[Bishop, 1984] Bishop, G. (1984). *Self-Tracker: A Smart Optical Sensor on Silicon.* PhD thesis, University of North Carolina at Chapel Hill, Department of Computer Science.

[Blake and Isard, 1994] Blake, A. and Isard, M. (1994). 3d position, attitude and shape input using video tracking of hands and lips. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 185–192.

[Bolles et al., 1987] Bolles, R. C., Baker, H. H., and Marimont, D. H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55.

[Bouguet and Perona, 1995] Bouguet, J. Y. and Perona, P. (1995). Visual navigation using a single camera. In *International Conference on Computer Vision (ICCV)*, pages 645–652. IEEE.

[Broida and Chellappa, 1986] Broida, T. J. and Chellappa, R. (1986). Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):90–99.

[Buttolo and Hannaford, 1995] Buttolo, P. and Hannaford, B. (1995). Pen-based force display for precision manipulation in virtual environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 217–224.

[Caudell and Mizell, 1992] Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proc. Hawaii Intl. Conference on System Sciences (HICCS) 1992.*, volume 2, pages 659–669.

[Chen, 1995] Chen, S. E. (1995). Quicktime vr - an image-based approach to virtual environment navigation. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 29–38. ACM.

[Cyberware, 1997] Cyberware (1997). *Cyberware Scanners*. Cyberware Inc., 2110 Del Monte Avenue, Monterey, CA 93940 (http://www.cyberware.com).

[Deriche, 1990] Deriche, R. (1990). Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87.

[Digital, 1997] Digital (1997). *Alpha Microprocessors*. Digital Equipment Corporation, (http://www.digital.com/info/semiconductor).

[Durlach and Mavor, 1994] Durlach, N. I. and Mavor, A. S., editors (1994). *Virtual Reality: Scientific and Technological Challenges*. National Academy Press, Washington, D.C.

[Edwards et al., 1993] Edwards, E., Rolland, J., and Keller, K. (1993). Video see-through design for merging of real and virtual environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 223–233, Seattle, WA.

[Ens and Lawrence, 1993] Ens, J. and Lawrence, P. (1993). An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):97–108.

[Faro, 1997] Faro (1997). *FaroArm*. Faro Technologies, Inc., Industrial Division, 125 Technology Park, Lake Mary, FL 32746 (http://www.faro.com).

[Faugeras et al., 1992] Faugeras, O., Loung, Q., and Maybank, S. (1992). Camera self-calibration: Theory and experiments. In *International Conference on Computer Vision (ICCV)*, pages 321–334. IEEE.

[Feiner et al., 1992] Feiner, S., Macintyre, B., and Seligmann, D. (1992). Annotating the real world with knowledge-based graphics on a see-through head-mounted display. In *Proc. Graphics Interface 1992*, pages 78–85. Canadian Information Proc. Soc.

[Feiner et al., 1993] Feiner, S., Macintyre, B., and Seligmann, D. (1993). Knowlege-based augmented reality. *Communications of the ACM*, 36(7):53–61.

[Forstner, 1994] Forstner, W. (1994). A framework for low level feature extraction. In *European Conference on Computer Vision*, pages 383–394.

[Fortune, 1987] Fortune, S. (1987). A sweeping algorithm for vornoi diagrams. *Algorithmica*, 2(2):153–74.

[Fuchs et al., 1989] Fuchs, H., Poulton, J., Eyles, J., Greer, T., Goldfeather, J., Ellsworth, D., Molnar, S., Turk, G., Tebbs, B., and Israel, L. (1989). Pixel-planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 79–88.

[Ganapathy, 1984a] Ganapathy, S. (1984a). Decomposition of transformation matrices for robot vision. *Pattern Recognition Letters*, 2(6):401–412.

[Ganapathy, 1984b] Ganapathy, S. (1984b). Real-time motion tracking using a single camera. Technical Report 11358-841105-21-TM, AT&T Bell Labs.

[GeneralReality, 1997] GeneralReality (1997). *CyberEye Head-Mounted Displays*. General Reality Company, 124 Race St., San Jose, CA 95216, (http://www.genreality.com).

[Gortler et al., 1996] Gortler, S. J., Grueszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The lumigraph. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 43–54.

[Gottschalk and Hughes, 1993] Gottschalk, S. and Hughes, J. (1993). Autocalibration for virtual environments tracking hardware. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 65–72.

[Gruss et al., 1992] Gruss, A., Tada, S., and Kanade, T. (1992). A vlsi smart sensor for fast range imaging. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Hecht, 1987] Hecht, E. (1987). *Optics*. Addison-Wesley, second edition edition.

[Hildebrand and Köthe, 1993] Hildebrand, A. and Köthe, U. (1993). SMART: system for segmentation matching and reconstructi on. *SPIE: OE/Aerospace and Remote Sensing, State-of-the- Art Mapping*, 1943.

[Holloway, 1995] Holloway, R. (1995). *Registration Errors in Augmented Reality Sytems*. PhD thesis, University of North Carolina at Chapel Hill.

[Horaud et al., 1989] Horaud, R., Conio, B., and Leboulleux, O. (1989). An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics and Image Processing*, 47:33–44.

[Janin et al., 1993] Janin, A. L., Mizell, D. W., and Caudell, T. P. (1993). Calibration of head-mounted displays for augmented reality applications. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*.

[Kaiser, 1997] Kaiser (1997). *Full Immersion Head Mounted Display System*. Kaiser Electro-Optics, Inc., Mr. Douglas DeFoe, (619)438-9255 x205, keo@cts.net (http://esto.sysplan.com/ETO/Displays/HMD/).

[Kanade, 1996a] Kanade, T. (1996a). Comments from seminar. Chapel Hill.

[Kanade, 1996b] Kanade, T. (1996b). A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings: 1996 ARPA Image Understanding Workshop*, pages 805–811. ARPA.

[Kanade et al., 1992] Kanade, T., Okutomi, M., and Nakahara, T. (1992). A multiple-baseline stereo method. In *Proceedings of the Darpa Image Understand Workshop*, pages 409–426.

[Lee and Cooper, 1993] Lee, C.-Y. and Cooper, D. B. (1993). Structure from motion: A region based approach using affine transformations and moment invariants. In *IEEE Intl. Conference on Robotics and Automation*, volume 3, pages 120–127. IEEE.

[Levoy and Hanrahan, 1996] Levoy, M. and Hanrahan, P. (1996). Light field rendering. In *Proceedings of ACM Siggraph, Computer Graphics.* ACM.

[Lewis, 1986] Lewis, F. L. (1986). *Optimal Estimation (with an introduction to stochastic control theory).* John Wiley & Sons.

[Liang et al., 1991] Liang, J., Shaw, C., and Green, M. (1991). On temporal-spatial realism in the virtual reality environment. In *Proceedings 4th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 19–25, Hilton Head, SC. ACM SIGGRAPH/SIGCHI.

[Liao and Medioni, 1996] Liao, C.-W. and Medioni, G. (1996). Surface approximation and segmentation of objects with unknown topology. In *Image Understanding Workshop*, pages 1033–1040. ARPA.

[List, 1984] List (1984). Nonlinear prediction of head movements for helmet-mounted displays. Technical Report AFHRL-TP-83-45, Williams AFB, AZ.

[Liu et al., 1990] Liu, Y., Huang, T. S., and Faugeras, O. D. (1990). Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37.

[Livingston and State,] Livingston, M. A. and State, A. Magnetic tracker calibration for improved augmented reality registration. *To appear in Presence: Teleoperators and Virtual Environments.*

[Longuet-Higgins, 1981] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

[Longuet-Higgins, 1984] Longuet-Higgins, H. C. (1984). The reconstruction of a scene from two projections - configurations that defeat the 8-point algorithm. In *Proceedings of 1st Conference on Artificial Intelligence Applications*, pages 395–397.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Seventh International Joint Conference on Artificial Intelligence IJCAI-81*, volume 2, pages 674–679. The International Joint Conferences on Artificial Intelligence.

[Luong and Faugeras, 1993] Luong, Q.-T. and Faugeras, O. (1993). Self-calibration of a stereo rig from unknown camera motions and point correspondences. Technical Report 2014, Institut National de Recherche en Informatique et en Automatique (INRIA).

[Matthies et al., 1989] Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209–236.

[Maybank and Faugeras, 1992] Maybank, S. J. and Faugeras, O. D. (1992). A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151.

[McMillan and Bishop, 1995] McMillan, L. and Bishop, G. (1995). Plenoptic modeling: an image-based rendering system. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 39–46. ACM.

[MediaLab, 1996] MediaLab (1996). *ALIVE: Artificial Life Interactice Video Environment.* MIT Media Lab, (http://lcs.www.media.mit.edu/projects/alive/ 8/28/96).

[Mellor, 1995a] Mellor, J. (1995a). Enhanced reality visualization in a surgical environment. Master's thesis, Massachusetts Istitute of Technology.

[Mellor, 1995b] Mellor, J. (1995b). Realtime camera calibration for enhanced reality visualization. In Ayache, N., editor, *Proceedings of the first international converence on Computer vision, Virtual reality, and Robotics in Medicine (CVRMed), Nice France.* Springer-Verlag.

[Mine and Bishop, 1993] Mine, M. and Bishop, G. (1993). Just-in-time pixels. Technical Report TR93-005, UNC Chapel Hill.

[Molnar et al., 1992] Molnar, S., Eyles, J., and Poulton, J. (1992). Pixelflow: High-speed rendering using image composition. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 231–240. ACM. Published as Computer Graphics, Proceedings of Siggraph, volume 26, number 2.

[Morita and Kanade, 1994] Morita, T. and Kanade, T. (1994). A sequential factorization method for recovering shape and motion from image streams. Technical Report CMU-CS-94-167, CMU.

[Mundy and Zisserman, 1992] Mundy, J. L. and Zisserman, A. (1992). *Geometric invariance in computer vision*, chapter Appendix, pages 463–519. MIT Press.

[Nayar et al., 1995] Nayar, S. K., Watanabe, M., and Noguchi, M. (1995). Real-time focus range sensor. In *International Conference on Computer Vision (ICCV)*, pages 995–1001. IEEE.

[O'Neill, 1966] O'Neill, B. (1966). *Elementary Differential Geometry.* Academic Press Inc., Orlando, Florida 32887.

[O'Rourke, 1994] O'Rourke, J. (1994). *Computational Geometry in C.* Cambridge ; New York : Cambridge University Press.

[Pausch et al., 1992] Pausch, R., Crea, T., and Conway, M. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence*, 1(3).

[Peuchot, 1993] Peuchot, B. (1993). Camera virtual equivalent model 0.01 pixel detectors. *Computerized Medical Imaging and Graphics*, 17(4/5):289–294.

[PHANToM, 1997] PHANToM (1997). *PHANToM Haptic Interface.* SensAble Technologies, Inc., 26 Landsdowne Street, University Park at MIT, Cambridge, MA 02139 Phone: (617) 621-0150 FAX: (617) 621-0135, email: sensable@sensable.com (http://www.sensable.com).

[Poelman and Kanade, 1992] Poelman, C. J. and Kanade, T. (1992). A paraperspective factorization method for shape and motion recovery. Technical Report CMU-CS-92-208, Carnegie Mellon University.

[Polhemus, 1997] Polhemus (1997). *FASTRACK(R) and ISOTRACK(R) magnetic tracking systems.* Polhemus, P.O. Box 560, Colchester, VT 05446, (http://www.polhemus.com).

[Qiu and Ma, 1995] Qiu, M. and Ma, S. (1995). The nonparametric approach for camera calibration. In *International Conference on Computer Vision (ICCV)*, pages 224–229. IEEE.

[Ravela et al., 1996] Ravela, S., Draper, B., Lim, J., and Weiss, R. (1996). Tracking object motion across aspect changes for augmented reality. In *Proceedings: 1996 ARPA Image Understanding Workshop*, pages 1345–1352. ARPA.

[Regan and Pose, 1994] Regan, M. and Pose, R. (1994). Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 155–162.

[Robinett and Rolland, 1991] Robinett, W. and Rolland, J. (1991). A computational model for the stereoscopic optics of a head-mounted display. Technical Report TR91-009, University of North Carolina.

[SAC, 1994] SAC (1994). *Model GP-12 3-D Digitizer.* Science Accessories Corporation, 2 Research Drive, P.O. Box 825, Shelton, CT 06484-0825, phone: (203)925-1661 fax: (203)929-9636.

[Shashua, 1994] Shashua, A. (1994). Projective structure from uncalibrated images: Structure from motion and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):778–790.

[Shekhar and Chellappa, 1992] Shekhar, C. and Chellappa, R. (1992). Passive ranging using a moving camera. *Journal of Robotic Systems*, 9(6):729–752.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600.

[Siegel and Leary, 1992] Siegel, M. and Leary, M. (1992). Range from focus pixel-by-pixel. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2035–2040.

[Skifstad, 1991] Skifstad, K. D. (1991). *High-Speed Range Estimation Based on Intensity Gradient Analysis*. Springer Series in Perception Engineering. Springer-Verlag.

[State et al., 1996a] State, A., Hirota, G., Chen, D., Garrett, W., and Livingston, M. (1996a). Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 429–438.

[State et al., 1996b] State, A., Livingston, M., Garrett, W., Hirota, G., Whitton, M., Pisano, E., and Fuchs, H. (1996b). Technologies for augmented-reality systems: Realizing ultrasound-guided needle biopsies. In *Proceedings of ACM Siggraph, Computer Graphics*, pages 439–446.

[Stone, 1989] Stone, R. C. (1989). A comparison of igital centering algorithms. *The Astronomical Journal*, 97(4):1227–1237.

[Strang, 1988] Strang, G. (1988). *Linear Algebra and Its Applications*. Harcourt Brace Janovich, Orlando, FL, 3rd edition.

[Sutherland, 1965] Sutherland, I. E. (1965). The ultimate display. In *Proc. IFIP Congress*, volume 65, pages 506–508.

[Sutherland, 1968] Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Fall Joint Computer Conference, AFIPS Conference Proceedings*, volume 33, pages 757–764.

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

[Szeliski and Kang, 1993a] Szeliski, R. and Kang, S. B. (1993a). Recovering 3d shape and motion from image streams using non-linear least squares. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 752–753.

[Szeliski and Kang, 1993b] Szeliski, R. and Kang, S. B. (1993b). Recovering 3d shape and motion from image streams using non-linear least squares. Technical report, Digital Equipment Corporation Cambridge Research Lab.

[Terzopoulos, 1988] Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438.

[Thomas and Oliensis, 1992] Thomas, J. I. and Oliensis, J. (1992). Recursive multi-frame structure frommotion incorporating motion error. In *Darpa Image Understanding Workshop*.

[Tomasi and Kanade, 1991a] Tomasi, C. and Kanade, T. (1991a). Shape and motion from image streams: A factorization method 2. point features in 3D motion. Technical Report CMU-CS-91-105, Carnegie Mellon University.

[Tomasi and Kanade, 1991b] Tomasi, C. and Kanade, T. (1991b). Shape and motion from image streams: A factorization method part 2. detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

[Tomasi and Kanade, 1992] Tomasi, C. and Kanade, T. (1992). Shape and motion from image streams: A factorization method 2. full report on the orthographic case [parts 2,8,10]. Technical Report CMU-CS-92-104, Carnegie Mellon University.

[Tsai, 1987] Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344.

[Tsai and Huang, 1984] Tsai, R. Y. and Huang, T. S. (1984). Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):13–26.

[Uenohara and Kanade, 1995] Uenohara, M. and Kanade, T. (1995). Vision-based object registration for real-time image overlay. In *Proceedings of the first international converence on Computer vision, Virtual reality, and Robotics in Medicine (CVRMed), Nice France*. Springer-Verlag.

[Ward et al., 1992] Ward, M., Azuma, R., Bennet, R., Gottschalk, S., and Fuchs, H. (1992). A demonstrated optical tracker with scalable work area for head-mounted display systems. In *Proceedings of Symposium on Interactive 3D Graphics, Cambridge, Mass.*, pages 43–52.

[Watson and Hodges, 1995] Watson, B. A. and Hodges, L. F. (1995). Using texture maps to correct for optical distortion in head-mounted displays. In *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, pages 172–178.

[Wei and Ma, 1993] Wei, G. and Ma, S. (1993). A complete two-plane camera calibration method and experimental comparisons. In *International Conference on Computer Vision (ICCV)*, pages 439–446. IEEE.

[Weng et al., 1993] Weng, J., Ahuja, N., and Huang, T. S. (1993). Optimal motion and structure estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):864–884.

[Weng et al., 1992] Weng, J., Cohen, P., and Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980.

[Willson and Shafer, 1993] Willson, R. G. and Shafer, S. A. (1993). What is the center of the image? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–671.

[Wloka and Anderson, 1995] Wloka, M. M. and Anderson, B. G. (1995). Resolving occlusion in augmented reality. In *ACM Symposium on Interactive 3D Graphics*, pages 5–12. ACM.

[Yaou and Chang, 1994] Yaou, M.-H. and Chang, W.-T. (1994). Fast surface interpolation using multiresolution wavelet transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):673–688.