Interactive Augmented Reality

by

James R Vallino

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Christopher M Brown

Department of Computer Science
The College
Arts and Sciences

University of Rochester
Rochester, New York

1998

Dedication

To my father.

One of the gentlest men I've known.

He taught me to always keep looking.

Curriculum Vitae

The author was born in Forest Hills, New York on 26 June 1954. He attended the Cooper Union for the Advancement of Arts and Science from 1971 to 1975, and graduated with a Bachelor of Engineering degree in Mechanical Engineering in 1975. He then attended the University of Wisconsin - Madison from 1975 to 1976, and received a Master of Science degree in Electrical Engineering in 1976. From 1976 until 1993 he held several positions in industry at AT&T Bell Laboratories, Audio Visual Labs and Siemens Corporate Research. He came to the University of Rochester in the Fall of 1993 and began graduate studies in Computer Science. He pursued his research in augmented reality under the direction of Professor Christopher Brown and received the Master of Science degree in Computer Science in 1995.

Acknowledgments

I must start by acknowledging the person who has been my closest companion through this doctoral experience, Charlene Varnis. Without her emotional and financial support I question whether I would have undertaken this journey. She picked up my slack when I had no time, said the right things to keep me going when I was down, and made sure that someone was taking care of me when I wasn't doing it myself.

And to my friends Marty and Sue. Even though I see them rarely now that miles separate us, they were on this journey too. They don't know how many times my sanity was saved by viewing the "Wall of Shame" or letting my mind wander back to the Grand Canyon, Switzerland, the Adirondaks or any of the other numerous places we have hiked, biked, skied, laughed and generally enjoyed each other's good company.

This thesis is the achievement of a long time goal of mine. I would like to thank my travelling companions Maureen, Art and Bill who were with me through the Summer of 1996 when I achieved another long time goal—bicycling across the U.S. Accomplishing that goal launched me into the final leg of this one.

My parents are responsible for laying the foundation upon which this work is built. I can not thank them enough for that. Even though my father is no longer here to witness this, I am sure that he is looking down with a smile and carefully checking my work to make sure I didn't leave any "holidays".

Your advisor is the person who leads you through the Ph.D. gauntlet. I could not ask for a better leader than my advisor, Chris Brown. Our discussions kept me headed in the right direction and gave me the motivation to continue when I wanted to fold up my tent. His confidence and loyalty were unwavering.

It is only occasionally that you come in contact with a truly creative person who is a wellspring of interesting, and challenging ideas. Fortunately for me, Kyros Kutulakos is such a person and one of his ideas was the spark for this thesis work. His assistance could not have been replaced especially during the early part of my work when I was clueless about affine geometry.

I would like to thank the other members of my committee, Dana Ballard, Randal Nelson and Murat Tekalp, for their insights and suggestions that have helped shape this work. And finally, I want to thank the students who started with me back in September of 1993—Garbis, Mark, Zaki, Wagner, Aaron, Colm, Umesh, Mauricio,

Mike and Terry. All of them did not make it through this entire process. But along the way having someone with whom I could share my ups, downs, frustrations, accomplishments, and failures made the experience more interesting, challenging, exciting, fun and tolerable for me.

I would like to express grateful appreciation to the following people who have given me permission to include images from their work in this thesis:

Figure 5 and Figure 11 - European Computer-Industry Research Centre (ECRC), Fraunhofer Institute for Computer Graphics (IGD), Department Visualization & Virtual Reality (A4), Group for Augmented Reality at ZGDV - work supported by Ove Arup & Partners and Sir Norman Foster & Partners, UK

Figure 7 - Dr. Eric Grimson, Computer Vision Group, MIT AI Lab

Figure 8 - Courtesy of the Department of Computer Science, University of North Carolina at Chapel Hill

Figure 9 - Dr. Simon Gibbs, GMD - German National Research Center for Information Technology

Figure 12 - Anu Rastogi, Dr. Paul Milgram, Dr. Julius Grodski, University of Toronto - work sponsored by the Defence and Civil Institute of Environmental Medicine, Department of National Defence, Canada

Figure 13 - Copyright 1993 by Steven Feiner, Blair MacIntyre, and Doree Seligmann, Columbia University

Figure 14a - Ulrich Neumann and Youngkwan Cho, Computer Science Department, University of Southern California

Figure 14b - Dr. David Mizell, The Boeing Company

This material is based upon work supported by the National Science Foundation under grant numbers IRI-9306454 and IRI-9406481 and by the National Institutes of Health under grant number 1 P41 RR 09283. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF or the NIH.

Abstract

Augmented reality is the merging of synthetic sensory information into a user's perception of a real environment. Until recently, it has presented a passive interface to its human users, who were merely viewers of the scene augmented only with visual information. In contrast, practically since its inception, computer graphics—and its outgrowth into virtual reality—has presented an interactive environment. It is our thesis that the augmented reality interface can be made interactive. We present: techniques that can free the user from restrictive requirements such as working in calibrated environments, results with haptic interface technology incorporated into augmented reality domains, and systems considerations that underlie the practical realization of these interactive augmented reality techniques.

Table of Contents

List of Tables

List of Figures

List of Symbols

$\mathbf{O}$	Object-to-world transform
$\mathbf{C}$	World-to-camera transform
$\mathbf{P}$	Camera-to-image plane transform
$\begin{bmatrix} x_w & y_w & z_w & w_w \end{bmatrix}^T$	Homogeneous world coordinates of a point in three-dimensional space
$\begin{bmatrix} u & v & h \end{bmatrix}^T$	Homogeneous coordinates of the projection in the image plane of a point in three-dimensional space
Π	Affine projection matrix
$\begin{bmatrix} x & y & z & w \end{bmatrix}^T$	Homogeneous affine coordinates of a point in three-dimensional space
f	Focal length of the camera lens viewing the real scene
z_{avg}	Average distance from camera for a set of points in three-dimensional space
$p_0, \ldots, p_n$	Collection of points in three-dimensional space
I_i	The *i*-th image in a sequence of images
χ, ψ	Vectors representing the direction of the rows and columns of the image plane in the affine camera
ζ	Affine viewing direction
$\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}, \mathbf{a_o}$	Affine coordinates for bounding box enclosing a virtual object
$\mathbf{T}_{gv}$	Graphics-to-video transform
$\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}$	Position of alignment marks in graphics image coordinate system
$\mathbf{i_1}, \mathbf{i_2}, \mathbf{i_3}$	Position of detected alignment marks in tracker image coordinate system
$\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}$	Homogeneous affine coordinates for four points used to align Phantom to the global affine coordinate system
$\mathbf{T}_{ap}$	Phantom-to-affine transform

1 Introduction and Motivation

1.1 The Interactive Augmented Reality Interface

Since its inception computer graphics has been interactive in nature. One of the novel aspects of the seminal computer graphics work in Ivan Sutherland's Sketchpad system (Sutherland 1963) was the ability of a user to interact with the system. A classic text in computer graphics, Newman and Sproull's *Principles of Interactive Computer Graphics*, (Newman and Sproull 1973) highlights this interactive nature directly in the title. The creation of interactive virtual environments originated in the computer graphics domain. The ultimate interactive environment is the real world. The goal of augmented reality systems is to combine the interactive real world with an interactive computer-generated world in such a way that they appear as one environment. Figure 1 shows a virtual clock tower placed in a real three-dimensional scene. As the user moves about the real scene and views it from different viewpoints, the image of the clock tower is continually updated so that it is perceived as a real object in the scene. This verisimilitude carries through to human interactions with the virtual object such as moving or lifting it, and the object's interaction with real objects, such as collisions. This thesis is concerned with the

Figure 1 - An augmented reality scene

visual and haptic (tactile) augmented reality interface. In particular, it addresses the issues of: (1) how to generate the augmented image composed from live video and computer-generated images, and (2) how to use a haptic device, the PHANToM™, to interact with the virtual objects.

1.2 The Major Challenges for Augmented Reality Systems

The major challenge for augmented reality systems is how to combine the real world and virtual world into a single augmented environment. To maintain the user's illusion that the virtual objects are indeed part of the real world requires a consistent registration of the virtual world with the real world. The nature of this registration problem in augmented reality systems is seen in Figure 2 where a virtual wireframe is

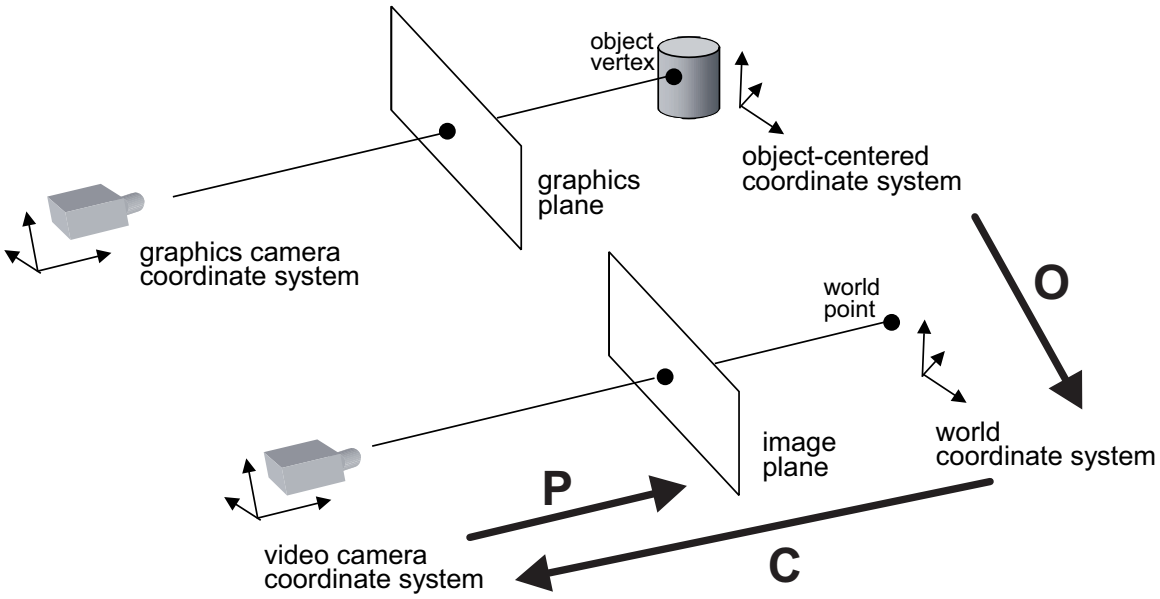
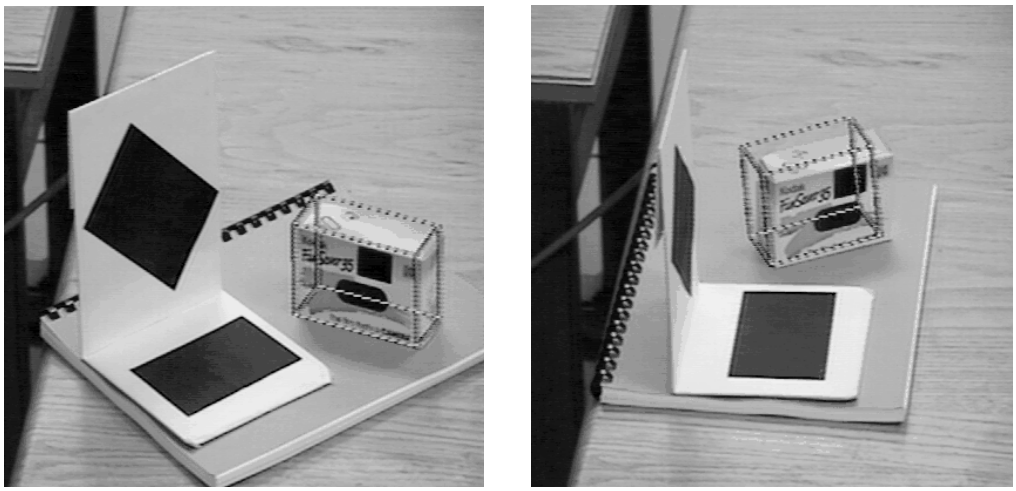


Figure 2 - Augmented reality coordinate systems

augmenting two views of a tabletop scene. The registration requirement for creating a high-fidelity augmented reality environment can be stated in terms of the relationships that must be determined. These relationships are the object-to-world, **O**, world-to-camera, **C**, and camera-to-image plane, **P**, transforms (Figure 2). The object-to-world transform specifies the position and orientation of a virtual object with respect to the world coordinate system that defines the real scene. The world-to-camera transform defines the pose of the video camera that views the real scene. Finally, the camera-to-image plane transform specifies the projection the camera performs to create a 2D image of the 3D real scene.

To create an image of the three-dimensional virtual objects consistent with the user's current view of the world and the object's placement in the real world requires the definition of the geometric relationships between the virtual and physical objects shown in Figure 2. Any errors in the determination of these relationships appear to the user as inconsistencies in the appearance of the virtual objects in the real scene (Figure 3a,b). These errors in registering the two images are classified as either static or dynamic. The user perceives static errors as differences in the placement or appearance of the virtual objects when viewed from different viewpoints. The dynamic errors are caused by the graphical augmentation being out of synchronization



(a) (b)

Figure 3 - Registration in augmented reality. (a) correct registration, (b) incorrect registration

with the real-world scene. Computational, sensing, or control latencies—resulting in system time lags—are the usual causes of dynamic errors. Delays can cause the augmented reality system to miss its real-time deadlines and can encourage solutions, such as prediction and estimation, whose own errors can contribute to the symptoms. The visual effect of these dynamic errors is a shift in the position of the virtual objects when there is motion in the system.

The discussion so far has been concerned with the visual augmentation of the real world scene. To faithfully create the interactions with the virtual objects there is an additional registration problem between the real world and the system generating the haptic display. There is a haptic-to-world transform that relates the generation of haptic senses of touch and feel to the world coordinate system. Visually and haptically correct merging of virtual objects with the view of the real scene requires computation of all these relationships. Accurately performing this computation while maintaining real-time response and a low latency is the major challenge for an augmented reality system.

This thesis provides a solution for the interactive augmented reality interface. The main contributions are:

- A new method to maintain the registration between the graphic display, haptic display and view of the real world. This technique uses affine representations to define a global coordinate system and unlike most previous augmented reality systems does not require calibration of the video camera viewing the scene or sensing of its position in the real scene.

- A method for the user to interact in a natural manner with the virtual objects seen in the augmented reality display. This interaction includes haptic sensing of the virtual objects, virtual objects exhibiting dynamic and gravitational behaviors, and interactions between virtual and real objects.

- An analysis of the static and dynamic registration error using our method of image registration and the effect of simple predictive filtering on dynamic errors.

- A system architecture that allows for the real-time operation of this interactive augmented reality system.

The outline of this thesis is as follows. Chapter 2 develops the idea of the ideal augmented reality system, motivating it by a large variety of current and safely-predictable near future applications. The challenges that must be overcome to reach this ideal are difficult given the hardware and software technology that exist today. Chapter 3 first makes explicit the current state of augmented reality hardware and software technology and follows up with a survey on the technical aspects of current approaches for solving the problem of augmenting reality. By the end of Chapter 3 we see that current techniques have not fully achieved the requirements specified in Chapter 2, and we preview our extensions to previous approaches that support our contributions. Chapters 4 and 5 present our contributions in detail and suggest promising future applications and research directions. Chapter 6 provides a summary of these contributions and future opportunities.

2 Augmented Reality: Potential and Challenges

An augmented reality system is a system that creates a view of a real scene by incorporating computer-generated virtual objects, including those with full three-dimensional properties, into the scene. As the user of such a system moves about the real scene the virtual objects appear as if they actually exist in the scene. Ideally, the virtual objects should interact with the user and real objects in the scene in a natural manner. The application domains described in Section 2.3 reveal that augmented reality can take on a number of different forms. In all the applications that are discussed later, augmented reality enhances the user's performance in and perception of the world.

The ultimate goal is to create a system such that the user can not tell the difference between the real world and the virtual augmentation of it. To the user of this ultimate system it would appear that he is working in a single real environment. Figure 4 shows a view of what the user might see while using an augmented reality

Figure 4 - Medical domain example of augmented reality

system in the medical domain. It depicts the merging and correct registration of data from a pre-operative imaging study onto the patient's head. Providing this view in the operating theater would enhance the surgeon's performance and possibly eliminate the

need for any other calibration fixtures during the procedure. Consider another application scenario as depicted in Figure 5. The city planners want to visualize what the landscape will look like when a new footbridge is built. They would go to the proposed site (Figure 5a) and (donning an augmented reality display device) see the area with the new bridge merged into their view of the landscape (Figure 5b). If adjustments were needed they could be performed directly on the visual model of the new bridge.

(a) (b)

Figure 5 - Preview of a new footbridge using an augmented reality display (Fraunhofer Institute for Computer Graphics 1997)

2.1 How Does Augmented Reality Differ from Virtual Reality?

Virtual reality is a technology that encompasses a broad spectrum of ideas. It defines an umbrella under which many researchers and companies express their work. The phrase was originated by Jaron Lanier, the founder of VPL Research—one of the original companies selling virtual reality systems. The term was defined as "a computer-generated, interactive, three-dimensional environment in which a person is immersed." (Aukstakalnis and Blatner 1992). There are three key points in this definition. First, this virtual environment is a computer-generated three-dimensional scene that requires high performance computer graphics to provide an adequate level of realism. The second point is that the virtual world is interactive. A user requires real-time response from the system to interact with it in an effective manner. The last point is that the user is immersed in this virtual environment. One of the identifying marks of a virtual reality system is the head-mounted display often worn by users. These displays block out all the external world and present to the wearer a view that is under the complete control of the computer. The user is completely immersed in an

artificial world and becomes divorced from the real environment. For this immersion to appear realistic the virtual reality system must accurately sense how the user is moving and determine what effect that will have on the scene being rendered in the head mounted display.

In contrast, an augmented reality system generates a composite view for the user. The user sees an image that is the combination of the real scene being viewed and a computer-generated virtual scene. The virtual scene augments the real environment with additional information. There are still the requirements that as the user moves through the workspace the effect on the computer generated world must be determined. Once that is done the operation of combining the virtual scene with the real scene must be performed.

The discussion above highlights the general similarities and differences between virtual reality and augmented reality systems. A very visible difference between these two types of systems is the immersiveness of the system. Virtual reality strives for a totally immersive environment. The visual, and in some systems aural and proprioceptive, senses are under control of the system. This requires the virtual reality system to model its artificial world completely. Depending on the desired level of verisimilitude, this is a very complex simulation problem. In contrast, an augmented reality system augments the real world scene and attempts to maintain the user's sense of being in the real world. The rationale behind this is twofold. First, real environments contain a wealth of information much of which is impossible to model and simulate by computer. Secondly, if the end goal is to enhance the performance of a real-world task, it will be performed most naturally if the user still feels immersed in the task environment. To maintain the user's immersion in the real world an augmented reality system merges the virtual images with a view of the real scene to create the augmented display. This merging requires a mechanism to combine the real and virtual that is not present in other virtual reality work. Developing the technology for merging the real and virtual image streams either optically or in video is an active research topic (Roland, Holloway et al. 1994) that is not addressed in this thesis. The technology is briefly described in Section 3.1.1.

Requiring a means to visually merge the views of the real and virtual worlds is not the major hurdle brought on by the inclusion of the real world in an augmented reality system. The nature of the visual interface between computer and user is different. Both virtual reality and augmented reality systems give the user a sense of immersion in the virtual environment by ensuring that the user receives a consistent set of sensory inputs. The primary performance goal for a virtual reality system is to present visual stimuli that are consistent with the changes in body position sensed by the user. This requires that motions or changes made by the user will result in the appropriate changes in the perceived virtual world. Because the user is looking at a virtual world there is no natural connection between the user's internal proprioceptive coordinate system and the virtual world coordinate system. A connection must be artificially created (Azuma 1993). Any inconsistency the user perceives results from a misregistration between the coordinate system the user maintains internally to describe

body position and the coordinate system that describes the graphics system's viewpoint in the virtual scene. Errors are perceived here as conflicts between the visual system and the kinesthetic or proprioceptive systems. The phenomenon of visual capture gives the vision system a stronger influence in our perception (Welch 1978). This stronger influence allows a user of a virtual reality system to accept or adjust to the visual stimulus by overriding the discrepancies occurring with input from the other sensory systems.

Contrast this to the primary performance goal for an augmented reality system which is to render views of virtual objects that are consistent with the user's view of the real environment containing the objects. In this case, errors of misregistration are between two visual stimuli which we are trying to fuse to see as one scene. The observer is more sensitive to these errors (Azuma 1993; Azuma 1995). Any inconsistency, which manifests itself as a difference between two visual stimuli, i.e. the virtual and real images, derives from a misregistration between the coordinate system describing the user's viewpoint in the real scene and the graphics system's viewpoint in the virtual scene. This imposes a tougher registration requirement on augmented reality systems compared to virtual reality systems.

2.2 Milgram Reality-Virtuality Continuum

Milgram (Milgram and Kishino 1994; Milgram, Takemura et al. 1994) describes a taxonomy that identifies how augmented reality and virtual reality are related. He defines the Reality-Virtuality continuum shown as Figure 6. The real

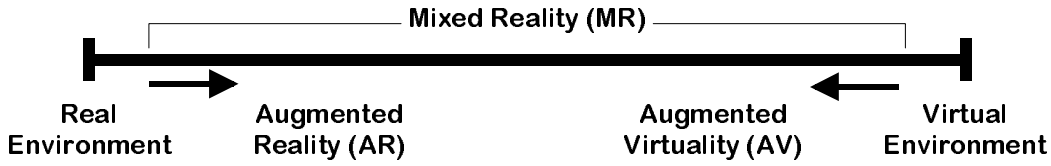


Figure 6 - Milgram's Reality-Virtuality Continuum

world and a totally virtual environment are at the two ends of this continuum with the middle region called Mixed Reality. Augmented reality lies near the real-world end of the spectrum with the predominate perception being the real-world augmented by computer generated data. Augmented virtuality is a term created by Milgram to identify systems that are mostly synthetic with some real world imagery added—such as texture mapping video onto virtual objects. An example of this technology is live video texture-mapped onto the graphics image of an avatar—a computer-generated

virtual character that is a stand-in for the user's presence within a virtual environment (Capin, Noser et al. 1997).

Milgram further defines a taxonomy for the Mixed Reality displays. The three axes he suggests for categorizing these systems are: Reproduction Fidelity, Extent of Presence Metaphor and Extent of World Knowledge. Reproduction Fidelity relates to the quality of the computer generated imagery ranging from simple wireframe approximations to complete photorealistic renderings. The real-time constraint on augmented reality systems forces them to be toward the low end on the Reproduction Fidelity spectrum. The current graphics hardware capabilities can not produce real-time photorealistic renderings of the virtual scene. Milgram also places augmented reality systems on the low end of the Extent of Presence Metaphor. This axis measures the level of immersion of the user within the displayed scene. This categorization is closely related to the display technology used by the system. There are several classes of displays used in augmented reality systems that are discussed in Section 3.1.1. Each of these gives a different sense of immersion in the virtual environment presented to the user. In an augmented reality system, some display technologies utilize the user's direct view of the real world. Immersion in that environment comes from the user simply having his eyes open. It is contrasted to systems where the merged view is presented to the user on a separate monitor for what is sometimes called a "Window on the World" (Feiner, MacIntyre et al. 1993a) view.

The third, and final, dimension that Milgram uses to categorize Mixed Reality displays is Extent of World Knowledge. Augmented reality does not simply mean the superimposition of a graphic object over a real world scene. This is technically an easy task. To do a realistic merging of virtual objects into a real scene, knowledge about the world is needed. One difficulty in augmenting reality, as defined here, is the need to maintain accurate registration of the virtual objects with the real world image This requires detailed knowledge of the relationship between the frames of reference for the real world, the camera viewing it and the user as described in Section 1.2. To properly compute the visual interactions between real and virtual objects, data about their locations and orientations in three-dimensional space are needed. Correct dynamic interactions with virtual objects require knowledge of the dynamic characteristics of both the real and virtual objects in the augmented environment. In some domains models for the real and virtual worlds are well known, which makes the task of augmenting reality easier or might lead the system designer to use a completely virtual environment.

2.3 Application Domains for Augmented Reality

Only recently have the capabilities of real-time video image processing, computer graphic systems and new display and haptic technologies converged to make

possible the creation of an augmented environment. In this environment, images of three-dimensional virtual objects are correctly registered with the view of the 3D environment surrounding the user and the user can interact naturally with the virtual objects. Researchers working with augmented reality systems have proposed them as solutions in many domains. The literature discusses application areas ranging from entertainment to military training. Many of the domains, such as medical (Rosen, Laub et al. 1996), are also considered domains for virtual reality systems. This section highlights some of the proposed applications for augmented reality

2.3.1 Medical

Because imaging technology is so pervasive throughout the medical field, it is not surprising that this domain is viewed as one of the more important for augmented reality systems. Most of the medical applications deal with image-guided surgery. Pre-operative imaging studies, such as CT or MRI scans, of the patient provide the surgeon with the necessary view of the internal anatomy. From these images the surgery is planned. The surgeon visualizes the path through the anatomy to the affected area where, for example, a tumor must be removed, by first creating a 3D model from the multiple views and slices in the preoperative study. Through their extensive training, surgeons become very adept at mentally creating the three-dimensional visualization that is needed to render a diagnosis. Some newer systems do have the ability to create 3D volume visualizations from the imaging study. Figure 7 shows how augmented reality can be applied so that the surgical team sees the CT or MRI data correctly registered on the patient in the operating theater while the procedure is progressing. Being able to accurately register the images at this point will enhance the performance of the surgical team and may eliminate the need for the painful and cumbersome stereotactic frames (Mellor 1995a) currently used for registration. Descriptions of other work in the area of image-guided surgery using augmented reality can be found in (Lorensen, Cline et al. 1993; Grimson, Lozano-

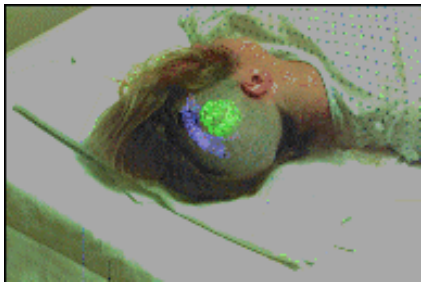
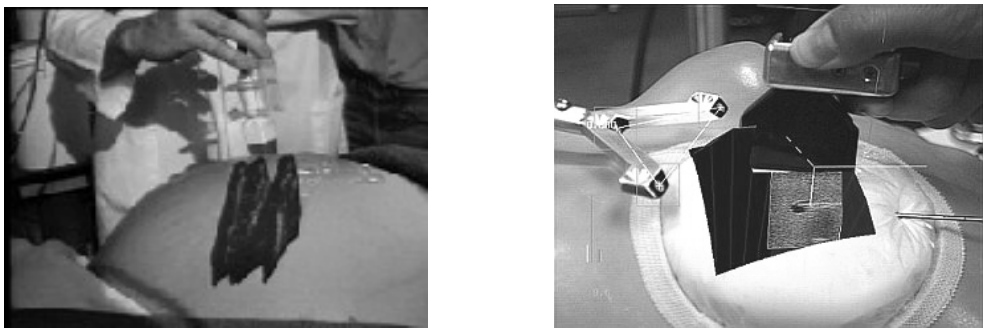


Figure 7 - Image guided surgical procedure (Ettinger, Grimson et al. 1998)

Perez et al. 1994; Betting, Feldmar et al. 1995; Grimson, Ettinger et al. 1995; Mellor 1995a; Uenohara and Kanade 1995; Jannin, Bouliou et al. 1997).

Another application for augmented reality in the medical domain is in ultrasound imaging (State, Chen et al. 1994). Using an optical see-through display (Section 3.1.1) the physician can view a volumetric rendered image of the fetus overlaid on the abdomen of the pregnant woman. The image appears as if it were inside the abdomen and is correctly rendered as the user moves. Figure 8a shows an image from the system along with a second application in ultrasound imaging seen in Figure 8b. In this second application, the augmented reality image helps the surgeon guide a biopsy needle to the site of a suspected tumor during a mock breast biopsy procedure. The V-shaped object in the left part of the image is used for registering the ultrasound image with the view of the real scene that the surgeon is seeing.



(a) (b)

Figure 8 - Ultrasound imaging using augmented reality displays (a (UNC - Chapel Hill 1995); b (UNC - Chapel Hill 1997))

Finally, a future application is the extension of the virtual reality based craniofacial surgery visualization and simulation systems (Patel, Vannier et al. 1996; Taylor, Funda et al. 1996) by adding an augmented reality display. These systems currently allow the surgeon to experiment, within the safety of a virtual environment, with different treatment approaches for the reconstructive work to be done. The model that the surgeon works on is a 3D volume visualization derived from a pre-operative CT or MR study. Augmented reality would allow the surgeon to see the final results directly on the patient rather than only with the volume visualization.

2.3.2 Entertainment

A simple form of augmented reality has been in use in the entertainment and news business for quite some time. When you watch an evening weather report the weather reporter often stands in front of changing weather maps. In the studio the

reporter is actually standing in front of a blue or green screen. This real image is augmented with computer generated maps using a technique called chroma-keying (Section 3.1.1). It is also possible to create a virtual studio environment so that the actors appear to be positioned in a studio with computer generated decorating (Gibbs 1995) an example of which is shown in Figure 9. The work with virtual studios goes

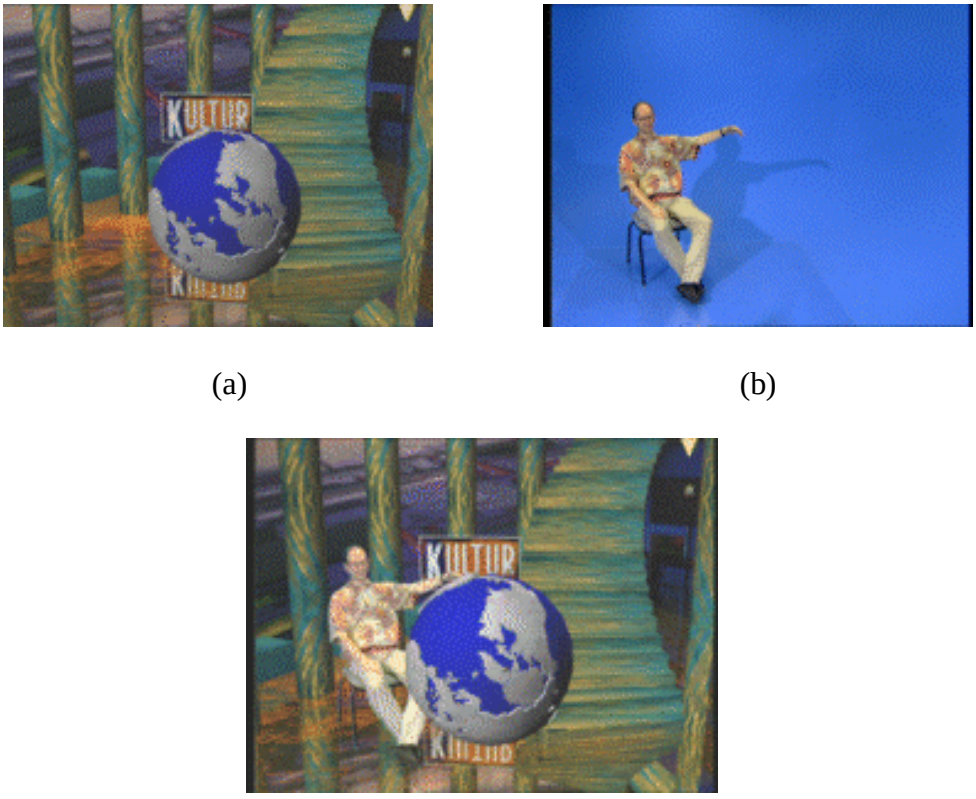


Figure 9 - Virtual set technology. (a) virtual set background, (b) live action, (c) combined video (Gibbs 1995).

beyond the simple chroma-keying of an actor over a flat two-dimensional background image. It extends the traditional keying to define the background in three-dimensions and key the live action as a three-dimensional image also. Balcisoy and Thalmann (1997) present a virtual studio populated by virtual humans that interact with human actors. The interactions are very carefully scripted and the human actor does not have the ability to feel a virtual actor. With tracking of the human, appropriate visual interactions can be accomplished. This system uses calibrated cameras and careful measurements of the location of objects in the real studio.

Movie special effects make use of digital compositing to create illusions (Pyros and Goren 1995). Strictly speaking, with current technology, this is not considered augmented reality because it is not generated in real-time. Most special effects are created off-line, frame by frame with a substantial amount of user interaction and computer graphics system rendering. But some work is progressing in computer analysis of the live action images to determine the camera parameters and use this to drive the generation of the virtual graphics objects to be merged (Zorpette 1994).

Princeton Electronic Billboard has developed an augmented reality system that allows broadcasters to insert advertisements into specific areas of the broadcast (National Association of Broadcasters 1994). For example, while broadcasting a baseball game this system places an advertisement in the image so that it appears on the outfield wall of the stadium. The electronic billboard requires calibration to the stadium by taking images from typical camera angles and zoom settings in order to build a map of the stadium including the locations in the images where advertisements will be inserted. By using pre-specified reference points in the stadium, the system automatically determines the camera angle being used and referring to the pre-defined stadium map inserts the advertisement into the correct place. The approach used for mapping these planar surfaces is similar to that which is presented in Chapter 4 of this thesis. A French company, Symah Vision, has also developed a similar application. Another application in sports broadcasting is Fox network's FoxTrax system (Cavallaro 1997) for tracking the path of a hockey puck during a game. The path of the puck is overlaid on the image of the hockey rink as a blue streak. The streak changed color based on the speed of the puck. This system requires a detailed calibration process for each television camera and the ice ring itself.

Augmented reality can be applied to enhance games that people play. A system (Jebara, Eyster et al. 1997) developed for pocket billiards players uses a head-mounted display and wearable computer to analyze the layout of the table and suggest possible shots for the player to take. The trajectory of the shot is displayed as augmented graphics over the image of the pool table. Or consider a futuristic game of paintball where players wear augmented reality headsets. The image that the players see is not only of the game area and their real opponents but virtual players are also playing along with them.

2.3.3 Military

The military has been using displays in cockpits that present information to the pilot on the windshield of the cockpit or the visor of their flight helmet. This is a form of augmented reality display. SIMNET, a distributed war games simulation system, is also embracing augmented reality technology. By equipping military personnel with helmet mounted visor displays or a special purpose rangefinder (Urban 1995) the activities of other units participating in the exercise are seen. While looking at the horizon, for example, the display-equipped soldier sees a helicopter rising above the

tree line (Metzger 1993). Actually, another participant is flying this helicopter in simulation. In wartime, the display of the real battlefield scene could be augmented with annotation information or highlighting to emphasize hidden enemy units.

The University of Rochester is participating in the Video Surveillance and Monitoring (VSAM) project funded by the Defense Advance Research Projects Agency (DARPA). Figure 10 shows the scenario for using augmented reality in this project. Aerial reconnaissance units fly overhead and generate reference marks for

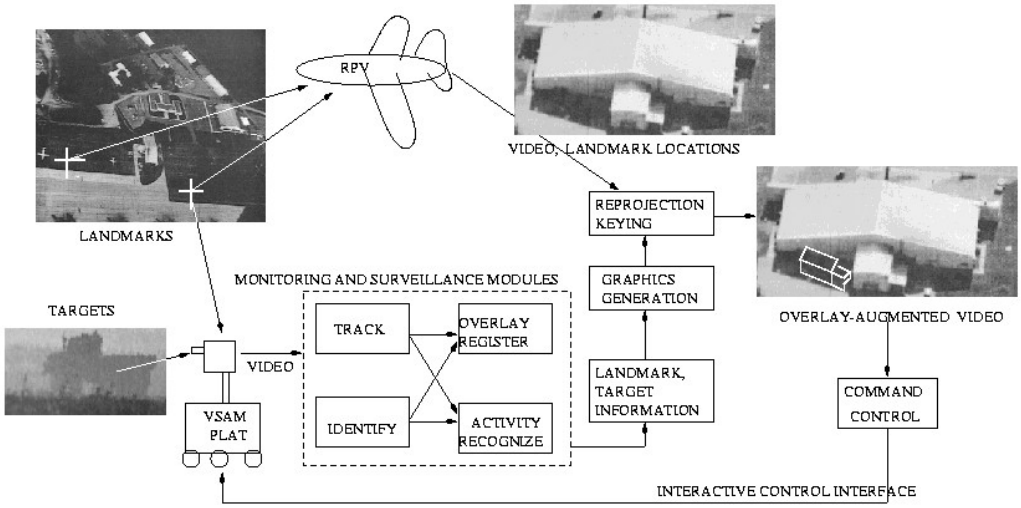


Figure 10 - Video Surveillance and Monitoring (VSAM) scenario

registration. Autonomous ground units with video surveillance equipment monitor sections of the same area. Aerial views augmented by the information from the surveillance units are generated for strategic command and control. Another scenario equips ground level warfighter or urban crime/terrorist fighters with special displays. Aerial reconnaissance units identify suspect objects and transmit the location of these objects to the ground units. The suspect objects may be hidden from the view of the ground forces but will appear in the augmented view on their displays.

2.3.4 Engineering Design

Imagine that a group of designers is working on the model of a complex device for their clients. The designers and clients want to do a joint design review even though they are physically separated. If each of them had a conference room that was equipped with an augmented reality display this could be accomplished. The physical prototype that the designers have mocked up is imaged and displayed in the client's conference room in 3D. The clients walk around the display looking at different aspects of it. To hold discussions the client points at the prototype to highlight sections and this is reflected on the real model in the augmented display that the designers are using (Figure 11). Or perhaps in an earlier stage of the design, before a prototype is built, the view in each conference room is augmented with a computer generated image of the current design built from the CAD files describing it. This allows real time interaction with elements of the design so that either side can make adjustments and changes that are reflected in the view seen by both groups (Ahlers,

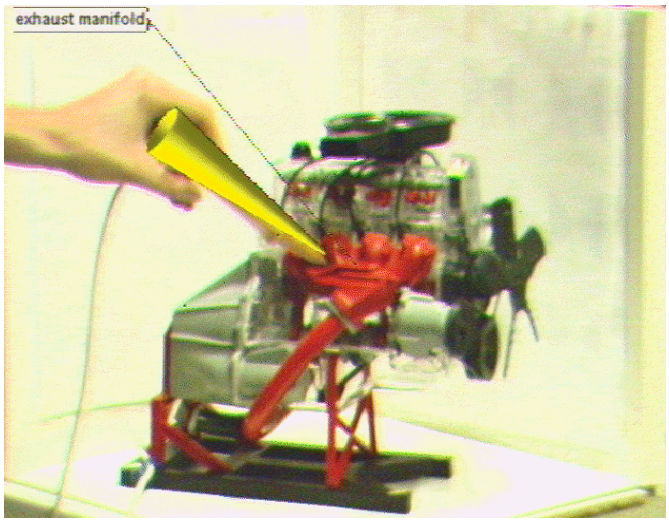


Figure 11 - Engineering design using an augmented reality display (Breen 1995)

Kramer et al. 1995). A technique for interactively obtaining a model for 3D objects called 3D stenciling that takes advantage of an augmented reality display is being investigated in our department by Kyros Kutulakos (Kutulakos and Vallino 1996a).

2.3.5 Robotics and Telerobotics

In the domain of robotics and telerobotics an augmented display can assist the user of the system (Kim, Schenker et al. 1993; Milgram, Zhai et al. 1993). Croby and Nafis (1994) describe an augmented reality telemanipulation system for nuclear reactor inspection operations. In a telerobotic system, the operator uses a visual image of the remote workspace to guide the robot. Annotation of the view would still be useful just as it is when the scene is in front of the operator. There is an added potential benefit. Since often the view of the remote scene is monoscopic, augmentation with wireframe drawings of structures in the view facilitates visualization of the remote 3D geometry. If the operator is attempting a motion he first practices it on a virtual robot that he sees as an augmentation to the real scene (Figure 12). The operator can decide to proceed with the motion after seeing the results. The robot executes the motion pattern directly which in a telerobotics application eliminates the oscillations often present due to long communication delays to the remote site.

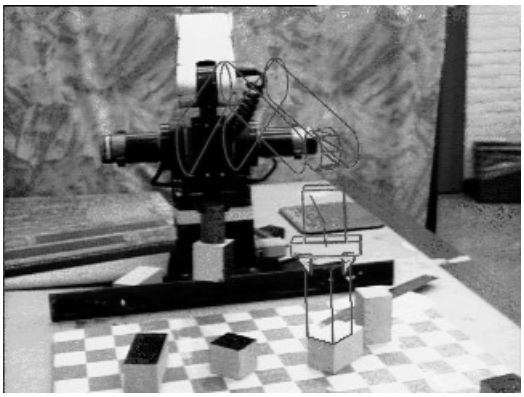


Figure 12 - Augmented reality in robotics (Rastogi, Milgram et al. 1995)

2.3.6 Manufacturing, Maintenance and Repair

When a maintenance technician approaches a new or unfamiliar piece of equipment instead of opening several repair manuals he simply puts on an augmented reality display. In this display an augmented reality system shows the image of the equipment augmented with annotations and information pertinent to the repair. For example, the location of fasteners and attachment hardware that must be removed are highlighted. As part of the next step an inside view of the machine highlights the boards that need to be replaced (Feiner, MacIntyre et al. 1993b; Uenohara and Kanade 1995). Figure 13 shows an example of this. The military has developed a wireless vest worn by personnel that is attached to an optical see-through display (Urban 1995). The wireless connection allows the soldier to access repair manuals and images of the equipment. Future versions might register those images on the live scene and provide animation to show the procedures that must be performed.

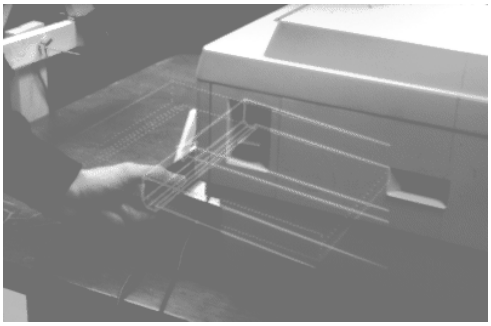
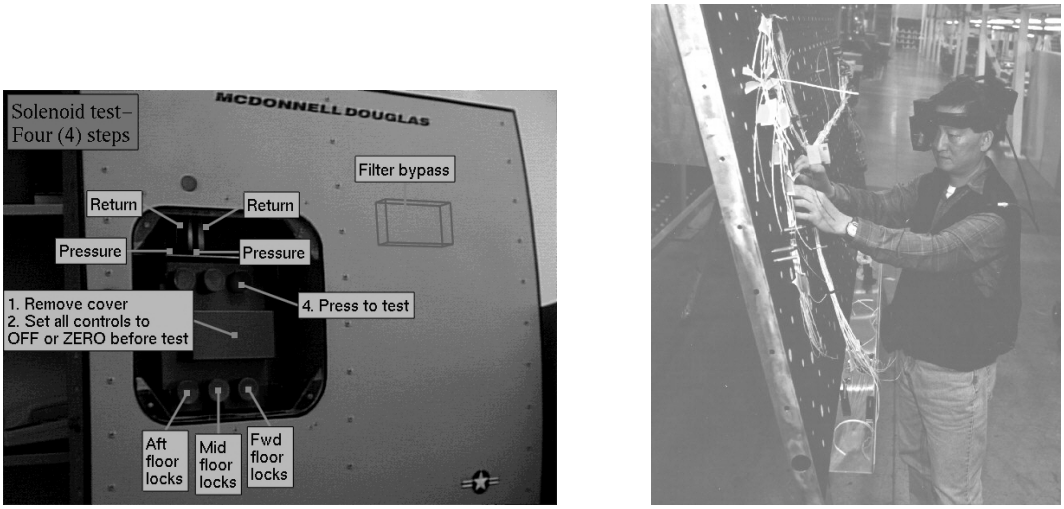


Figure 13 - Equipment maintenance application for augmented reality (Feiner, MacIntyre et al. 1995)

Aircraft manufacturers are particularly active in incorporating augmented reality systems into their manufacturing and maintenance activities. Boeing researchers developed an augmented reality display to replace the large work frames used for making wiring harnesses for their aircraft (Caudell 1994; Sims 1994). Using this experimental system (Figure 14b), the technicians are guided by the augmented display that shows the routing of the cables on a generic frame used for all harnesses. The augmented display allows a single fixture to be used for making the multiple harnesses. Research undertaken by the University of Southern California (Neumann and Cho 1996) in conjunction with McDonnell-Douglas is using an augmented reality system to guide technicians through maintenance and repair sequences Figure 14a.



(a) (b)

Figure 14 - Aircraft manufacturing use of augmented reality (a) (Neumann and Cho 1997); b) Dr. David Mizell, The Boeing Company)

2.3.7 Consumer Applications

Virtual reality systems are already used for consumer design. There are programs available for home computers that assist the homeowner with remodeling projects, such as building a new deck. These computer graphics programs allow you to graphically build the new deck attached to the back of a general model of a house. It is conceivable that a future system would allow you to make a video tape of your house shot from various viewpoints in your backyard. The video is input to your computer so that the design program could show the new deck in its finished form attached to your house. Or use a tape of your current kitchen and the augmented reality program replaces your current kitchen cabinetry with virtual images of the new kitchen that you are designing.

Applications in the fashion and beauty industry that would benefit from an augmented reality system can also be imagined. If the dress store does not have a particular style dress in your size an appropriate sized dress could be used to augment the image of you. As you look in the three sided mirror you see an image of the new dress on your body. You view changes in hem length, shoulder styles or other particulars of the design before you place the order. When you head into some high-tech beauty shops today you see what a new hair style will look like on a digitized image of yourself. But with an advanced augmented reality system you see the new

styling as you move. If the dynamics of hair are included in the description of the virtual object you also see the motion of your hair as your head moves.

2.4 Performance Goals

The application domains and projects described so far in this chapter motivate the desire to develop interactive augmented reality systems. These current and future applications help to shape the requirements specification for an interactive augmented reality system. Consider the requirements specified in this section to be the ideal requirements. It is against these requirements that the reader will compare the achievements of the previous approaches to augmenting reality (Chapter 3) and the approach put forward by this thesis (Chapter 4). These comparisons should be tempered somewhat by the particular application under consideration. Some applications will not demand an "ultimate" augmented reality system.

An augmented reality system with a high degree of verisimilitude and utility will possess the following ideal characteristics:

- *Constrained cost to allow for broader usage.* To allow augmented reality systems to be deployed in a wide range of applications the cost for the system should be constrained. This leads to a goal of using inexpensive consumer grade video cameras, personal computer processing and lower resolution display technology.

- *Perfect static registration of virtual objects in the augmented view.* When a virtual object has been placed at a location in the real scene it should appear to the user to remain at that same position in 3D space unless an object has interacted with it. If the system can not meet this goal for all static viewpoints, it will not be possible to met the following more difficult dynamic registration requirement.

- *Perfect dynamic registration of virtual objects in the augmented view.* Visual updates should be performed at a rate of, at least 15 Hz, and preferably, 30 Hz. Perhaps more importantly, latencies should be minimized. If changes in the rendering of the virtual objects lag behind the user action triggering them the virtual objects will appear to "swim" around in three-dimensional space.

- *Perfect registration of visual and haptic scenes.* This can be phrased as WYSIWYF or "What you see is what you feel." The visual image of a virtual object should match with its haptic counterpart. The user should

feel the surface of a virtual object at the same time and in the same place that the augmented view shows the contact.

- *Virtual and real objects are visually indistinguishable.* In addition to photorealistic rendering of the virtual objects—the usual consideration for computer graphics applications—there are additional requirements specific to interactive augmented reality applications. Visual occlusions between virtual and real objects must occur correctly. This is not only for virtual objects occluding real ones, but also for the more difficult case of real objects occluding virtual ones. Lighting in the augmented view must be matched between the real and virtual worlds.

- *Virtual objects exhibit standard dynamic behavior.* When the user interacts with a virtual object it should move with the same dynamic behavior that an equivalent real object would exhibit. This includes correctly rebounding from collisions between virtual objects or between virtual and real objects. To accommodate this characteristic, the system's internal representation of objects should help to compute the graphics rendering of virtual objects and the interactions and dynamics of all objects.

- *The user has unconstrained motion within the workspace.* The system should allow movement without constraints or limitations. It would be ideal to have no mechanical limitations, blind spots or motion constraints.

- *Minimal apriori calibration or run-time setup is required.* To determine the location of the viewer many augmented reality systems require calibration of the video camera viewing the scene. This calibration process is tedious to perform and will often limit operation to a single focal length. Lenses on standard consumer grade video cameras can not be zoomed in or out because they do not provide feedback of zoom position. During start-up of the system the user should not have to perform extensive setup such as measurement of the locations of fiducials, or complicated procedures for placing objects into the scene.

To date, no augmented reality system, including the one developed as part of this thesis work, has met all of the performance goals for the ideal system. Comparing these goals against the capabilities of the previous approaches to augmenting reality described in Chapter 3 and the approach using affine representations presented by this thesis (Chapter 4) the reader can determine how close the state-of-the-art comes to achieving all of the performance goals for the ultimate interactive augmented reality system.

3 Augmented Reality: Technology and Practice

This chapter describes the technology and practice of augmented reality systems. There is some unique technology involved upon which augmented reality systems are built. The first section of this chapter discusses the display and haptic technology that is available for use in an interactive augmented reality system. The chapter continues with a discussion of the previous practice for realizing an augmented reality interface by, first, presenting a simple taxonomy for the methods used to register the real and virtual images. A review of the prior research in registering the virtual and real images follows. The chapter concludes with a discussion of the small amount of haptic work associated with augmented reality interfaces.

3.1 System Components

A system that creates an augmented reality interface is a tightly connected set of components operating in real time. This section describes the components of augmented reality systems and highlights the state-of-the-art technology that is available for their implementation specifically in the areas of display and haptic technology.

3.1.1 Augmented reality display technology

Combining real and virtual images into a single image presents new technical challenges for designers of augmented reality systems. How to do this merging of the two images is a basic decision the designer must make. Section 2.2 discussed the continuum that Milgram (Milgram and Kishino 1994; Milgram, Takemura et al. 1994) uses to categorize augmented reality systems. His Extent of Presence Metaphor directly relates to the display that is used. At one end of the spectrum is monitor-based viewing of the augmented scene. This has sometimes been referred to as "Window on the World" (Feiner, MacIntyre et al. 1993a) or Fish Tank virtual reality (Ware, Arthur et al. 1993). The user has little feeling of being immersed in the

environment created by the display. This technology, diagrammed in Figure 15, is the simplest available. It is the primary technology that the work in this thesis uses as do several other systems in the literature (Drascic, Grodski et al. 1993; Ahlers, Breen et al. 1994).

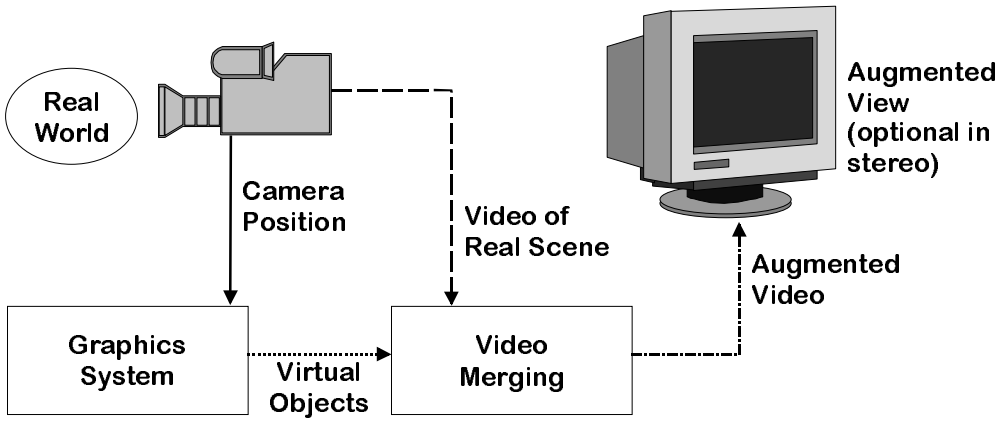


Figure 15 - Monitor-based augmented reality display

To increase the sense of presence other display technologies are needed. Head-mounted displays (HMD) are widely used in virtual reality systems. Augmented reality researchers work with two types of HMD. These two types are called video see-through and optical see-through. The "see-through" designation comes from the need for the user to be able to see the real world view that is immediately in front of him even when wearing the HMD. The standard HMD used in virtual reality work

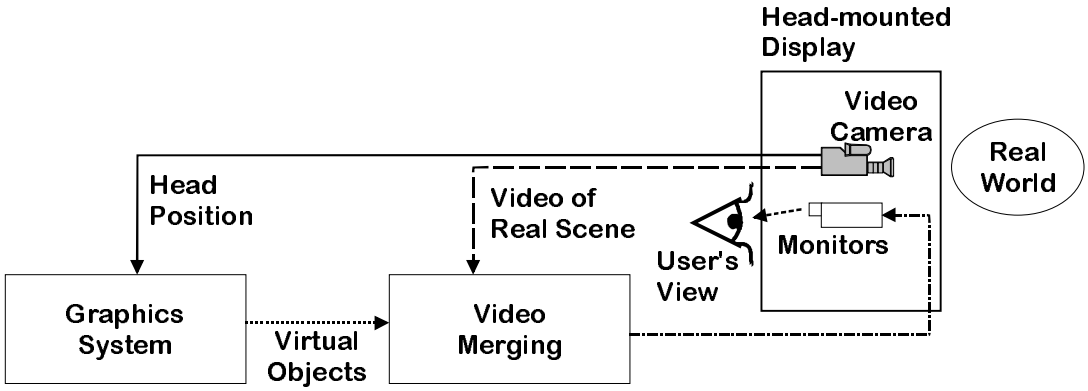


Figure 16 - Video see-through augmented reality display

gives the user complete visual isolation from the surrounding environment. Since the display is visually isolating the system must use video cameras that are aligned with the display to obtain the view of the real world. A diagram of a video see-through system is shown in Figure 16. Video see-through is really the same architecture as the monitor-based display except that now the user has a heightened sense of immersion in the display. Some experiments with a video see-through system have been done as part of our work (Section 4.5.4). Figure 17 shows the configuration of two video cameras mounted on an HMD.

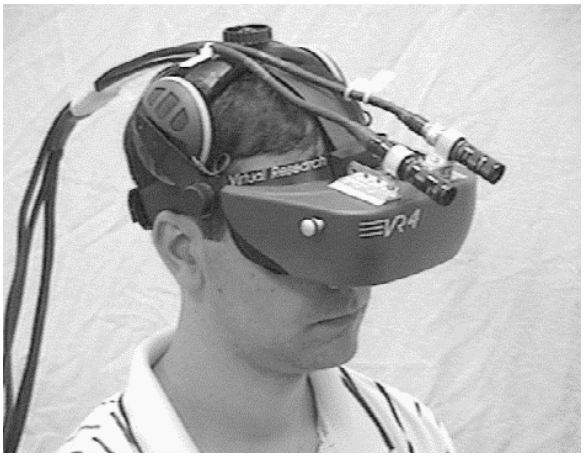


Figure 17 - Video see-through HMD

Both the monitor-based and video see-through display technologies require a method to merge two video channels—real and virtual—into a single image. This is an operation quite similar to the process of putting a weather forecaster in front of computer-generated weather maps for the nightly TV news. The operation is known as video keying and is based on either luminance or chrominance information in the video signals. One video channel is called the key while the second is the background. The key controls which video signal appears at the output of the keyer on a pixel-by-pixel basis. At every pixel where the key meets the keying criterion the background video signal is displayed or "keyed". At all other pixel locations the key is displayed. With luminance keying the criterion is that the luminance component is below a threshold luminance value. Usually this is set close to the luminance of the darkest region in the key. In those dark regions the background video channel is displayed. Other regions in the key signal—representing objects in the foreground image like the weather forecaster—are most likely above the luminance key value and will appear to be in front of the background. The other style of keying, which uses color as the key criterion, is known as chroma-keying. The operation is identical to luminance keying except the criterion is set to be a match to a certain color, often blue (Figure 9).

The optical see-through HMD (Manhart, Malcolm et al. 1993) eliminates the video channel that is looking at the real scene. Instead, as shown in Figure 18, the merging of real world and virtual augmentation is done optically in front of the user. This technology is similar to heads up displays (HUD) that commonly appear in military airplane cockpits and recently some experimental automobiles. In this case, the optical merging of the two images is done on the head mounted display, rather than the cockpit window or auto windshield, prompting the nickname of "HUD on a head".

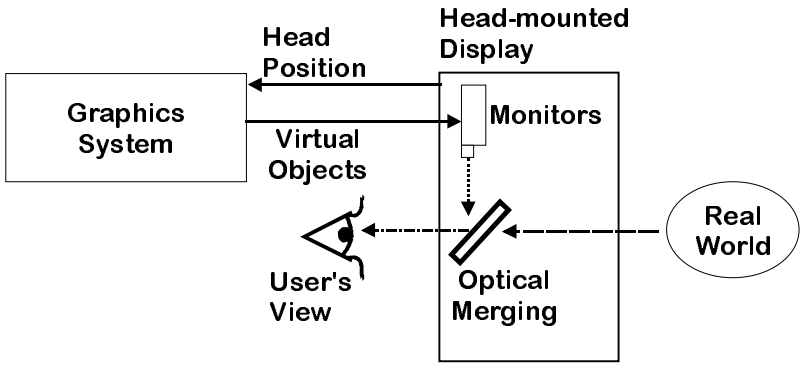


Figure 18 - Optical see-through augmented reality display

There are advantages and disadvantages to each of these types of displays. They are discussed in greater detail by Azuma (Azuma 1995). There are some performance issues, however, that will be highlighted here. With both of the displays that use a video camera to view the real world there is a forced delay of at least one frame time to perform the video merging operation. At standard frame rates that will be potentially a 33.33 millisecond delay in the view the user sees. Since the computer controls everything the user sees, compensation for this delay could be made by correctly timing the other paths in the system. Or, alternatively, if other paths are slower then the video of the real scene could be delayed. With an optical see-through display the view of the real world is instantaneous so it is not possible to compensate for system delays in other areas. On the other hand, with monitor-based and video see-through displays a video camera is viewing the real scene. Access to the image generated by the camera can be put to advantage if the system analyzes the video data to extract tracking information. The technique for augmenting reality described in Chapter 4 exploits this advantage. The optical see-through display does not have this additional information. The only position information available with that display is what can be provided by position sensors on the head mounted display itself.

3.1.2 Haptic Technology

Haptic technology provides the user with an ability to experience touch sensations. In the context of interactive augmented reality this is not referring to the apparatus used in flight and driving simulators. The user's kinesthetic and proprioceptive receptors are being stimulated by those systems. In an interactive augmented system the emphasis is on user interaction with the augmented environment. This requires force reflexive components that give the user a sense of feeling the virtual objects, touching their surface, and interacting with them in a dynamic fashion. Some work in virtual reality applications uses a haptic interface. In Project GROPE at the University of North Carolina (Brooks, Ouh-Young et al. 1990) the user manipulates molecules in a virtual environment using a large scale manipulator. The Rutgers Dexterous Master (Fabiani, Burdea et al. 1996) is a pneumatic cylinder system that applies force feedback to several fingers and allows the user to feel the general shape and stiffness of an object. Other force reflexive exoskeletons that apply forces to the arms and legs of the user are also commercially available. The current state of the art in haptic interfaces is reviewed in Srinivasan and Basdogan (Srinivasan and Basdogan 1997) and Ziegler (Ziegler 1996).

None of these haptic components is quite satisfactory for the interactions wanted in our augmented reality system. To allow realistic interactions with the virtual objects we wanted the user to feel the weight of the objects. This eliminated bladder gloves and the Rutgers Dexterous Master that can not simulate gravitational forces. Exoskeletons and large scale manipulators were too large for the work area envisioned. We chose the PHANToM™, manufactured by Sensable Technologies. This

Figure 19 - PHANToM™ haptic interface

device, shown in Figure 19, looks like a miniature robot arm. In many ways it is. Each joint is driven by a small motor and the controller coordinates the motor operation to generate force feedback at the end effector. The standard end effector is a thimble into which the user inserts a finger. With the supplied GHOST library the system can define a world of objects that create the haptic scene to be rendered. The Phantom gives the user a very compelling sense of touching these virtual objects. Mass can be assigned to the objects so if the user places his finger under an object and lifts, the weight of the object is felt resting on the finger. It is also possible to simulate characteristics such as object surface compliance and texture.

Haptic simulation comes with a high computational load requirement (Mark, Randolph et al. 1996). To provide a veridical sensation, the Phantom low-level processing servo loop requires servicing at a 1 kHz rate. This places a high computational load on the processor supporting this haptic device. We use the smallest device available. It has 3 degree of freedom force feedback and supports the range of motion approximate to hand movement pivoting at the user's wrist. Sensable Technologies also manufactures larger sized devices and the uninstrumented standard end tool can be replaced with an instrumented stylus tool. This last item provides data about the orientation of the user's grasp at the active end of the Phantom.

3.2 A Taxonomy for Registering the Real and Virtual Images

The primary challenge for an augmented reality system is to determine the relationship between the multiple coordinate systems shown in Figure 2. Only when these relationships are known can the images of the virtual objects and the view of the real scene be correctly registered for merging as a single image. Most of the previous work has approached this problem in a straightforward manner using position sensing to obtain the location and orientation of the viewer or the camera viewing the scene. This combined with information about the calibration parameters of the camera completely defines the relationships between coordinate systems.

We classify previous approaches in a taxonomy according to their usage of position sensing and camera or scene calibration. Table 1 shows this taxonomy. Methods using position sensing typically measure the location of the viewer and/or camera viewing the scene with a sensor. The Polhemus magnetic sensor (Polhemus Corporation 1996) is typical. The other axis in Table 1 classifies methods based on whether they require the intrinsic parameters of the camera viewing the scene or knowledge of the metric location of features in the real scene. The intrinsic camera parameters might include lens focal length, and the height and width of the pixel sensor. The scene calibration is usually the location of fiducial points in the three dimensional real scene. The system detects the fiducials in the image at runtime and from their position computes the location of the viewer.

Table 1 - Taxonomy for approaches to registration in augmented reality

	Position Sensing	
Camera and/or Scene Calibration	Yes	No
Yes	Janin, et al (1993) Feiner, et al (1993b) Rastogi, et al (1995) Tuceryan, et al (1995) State, et al (1996)	Grimson, et al (1995) Mellor (1995a) Hoff, et al (1996) Neumann and Cho (1996)
No	State, et al (1994)	Uenohara and Kanade (1995) Kutulakos and Vallino (1996a)

3.3 Position-based Augmented Reality

Correct registration of a virtual image with the image of the real scene requires the system to represent the two images in the same frame of reference. This requires determining the transforms between the various coordinate frames in the system as discussed in Section 1.2. Many augmented reality systems approach the problem of computing the transforms shown in Figure 2 in a straightforward manner using sensing, calibration and measurement to explicitly determine each transform (Feiner, MacIntyre et al. 1993b; Janin, Mizell et al. 1993; Rastogi, Milgram et al. 1995; Tuceryan, Greer et al. 1995). These systems use sensors—based on mechanical, magnetic or optical techniques—to measure the position and angular orientation of the camera with respect to the world coordinate system. These two measurements together are termed the *pose* of the camera and determine the world-to-camera transform, **C**. Quantifying the camera-to-image transform, **P**, requires knowledge of the intrinsic parameters, such as focal length and aspect ratio, of the camera. These can be determined by performing a calibration procedure on the camera (Tsai 1987). The third transform, **O**, is computed by simple measurement. The world coordinate system is a standard three-dimensional Euclidean space. The user measures the desired position and orientation for a virtual object in the real scene. Using the methods just described, all of the necessary transforms are known so that, at least in principle, virtual objects can be rendered and merged correctly with the live video.

These position-based approaches do suffer from limitations. Position tracking and camera calibration techniques are error prone in operation resulting in misregistration of the real and virtual images. So far none of the pose sensors have been completely satisfactory for an augmented reality application (Azuma 1997). Mechanical sensors place limits on the range of the work space and require attachment to a restrictive linkage. Magnetic sensors are susceptible to disturbances in their

generated magnetic field created by metal objects in the workspace. Calibration for these distortions can reduce the errors (Byrson 1992). The magnetic sensors also have latencies that can only be improved with predictive estimation of pose (Azuma and Bishop 1994) Techniques for calibrating a camera to determine its intrinsic parameters are available (Tsai 1987) but it is a tedious process to perform. The intrinsic parameters of a camera may change over time, requiring recalibration. In particular, zoom lenses, like those found on common consumer-grade video cameras, may change focal length either by being intentionally zoomed or because of wear through use. Accurate sensing of zoom setting is not available, so recalibration is required after each change in focal length. If cost containment is not a concern then instrumented zoom lenses can be used, but they require precalibration of the lens at each zoom setting (and possibly occasional recalibration). Any errors introduced by incorrect pose sensing or camera calibration propagate through the system and appear as misregistration in the final augmented reality image.

One position-based reference in Table 1 does not require camera or scene calibration. State, Chen, et. al. (1994) describe the ultrasound visualization system whose operation is shown in Figure 8a. Their system uses Polhemus sensors for measuring the position of the ultrasound imaging wand and the user's position and orientation while wearing an optical see-through display system. These measurements are both in the same coordinate system and thus directly provide the relationship between them. There is no camera, per se, in the system so calibration is not needed.

Durlach and Mavor (1995) come to the conclusion that position tracking as used in augmented reality systems will never work well enough to be the sole tracking technology because of the inaccuracies and delays in the system. They suggest that the most promising technique may combine standard position tracking for gross registration and an image based method for the final fine tuning. State, Hirota, et. al. (1996) have implemented such a tracking methodology and their work will be discussed in the next section with other registration methods that apply computer vision techniques to the problem.

3.4 Computer Vision for Augmented Reality

The initial approaches to augmented reality discussed in the previous section all overlook one source of significant information. Computer vision research has developed techniques for extracting information about the structure of a scene, the intrinsic parameters of the camera, and its pose from images of the scene. Recent augmented reality systems are applying computer vision methods to improve performance. Tuceryan et al. (1995) provides a careful analysis of procedures that rely on computer vision for calibrating a monitor-based augmented reality system. There is a growing number of systems described in the literature that try to mitigate or eliminate the errors due to tracking and calibration by using image processing of the

live video data (Bajura and Neumann 1995; Wloka and Anderson 1995). Other systems (Hoff, Nguyen et al. 1996; Neumann and Cho 1996; Koller, Klinker et al. 1997) use knowledge of the intrinsic camera parameters and tracking of fiducials placed in known locations in the scene to invert the camera projection operation and obtain an estimate of the viewer pose.

A hybrid method—using fiducial tracking in combination with standard magnetic position tracking (State, Hirota et al. 1996)—requires an initialization procedure that determines the intrinsic parameters of the cameras viewing the scene. This system tracks fiducials in two video images. The locations in the scene of the fiducials are known. The position of the viewer is computed by inverting the projection operation. Position data obtained from a magnetic tracker aides in localization of the landmarks. This aide is particularly useful when large motions are encountered between two video frames. Algorithms that rely solely on vision-based tracking often can not determine the interframe correspondences between fiducials when large motions occur between frames. The magnetic tracker position estimates are also used when occlusions prevent the vision system from seeing the required minimum number of fiducials.

A few recent augmented reality systems neither rely on a method for tracking the position of the camera nor require information about the calibration parameters of that camera. These systems solve the problem of registering the virtual objects over the live video as a pose estimation problem. By tracking feature points in the video image these systems invert the projection operation performed by the camera and estimate the camera's parameters. This does, however, require knowledge of the Euclidean 3D location of the feature points so that the camera parameters can be estimated in a Euclidean frame. For each viewpoint, two of the systems (Grimson, Lozano-Perez et al. 1994; Grimson, Ettinger et al. 1995; Mellor 1995b; Mellor 1995a) use a linear method to solve for the camera's projection transformation from the positions of tracked fiducials. The pose—which specifies the 3D position and orientation of the camera—is not directly determined. Instead these systems use the computed projection transformation to render the virtual objects. The camera must be calibrated at initialization time and a laser range finder provides the 3D positions of the fiducials in the scene.

One common theme of this previous work is that all the reference frames are defined in a 3D Euclidean space. To extract this information from the image of the real scene is an error prone process. By relaxing the requirement that all frames have a Euclidean definition it is possible to eliminate the need for this precise calibration and tracking. An approach similar to the one proposed by this thesis is described by Uenohara and Kanade (1995). They visually track features in live video images and represent the points on planar overlays as the linear combination of tracked feature points. Their work did not consider, however, the placement and rendering of three-dimensional virtual objects were not considered however. Chapter 4 describes our method for augmenting reality that requires no a priori metric information about the intrinsic and extrinsic parameters of the camera, where the user is located in the

world, or the position of objects in the world. Our method achieves the capacity to operate with no calibration information by using an affine representation as a common coordinate system, and substituting a simple correspondence problem for the metric calibration problem.

3.5 Interactive Augmented Reality

None of the prior work discussed to this point has included any interaction with the virtual objects except for the visual changes seen in the augmented reality display whenever the user changed viewpoint. One performance goal for an augmented reality system is that the user can naturally interact with the virtual objects. This interaction should include not only moving the objects, but also feeling their surfaces and the forces applied to them by gravity and by other objects in the environment. Haptic relates to the sense of touch. The user of a haptic interface receives tactile feedback that adds the ability to feel objects. There is no work in the literature that describes haptic extensions in augmented reality systems. All the previous haptic research is in the areas of telemanipulation and virtual reality. We can apply the work in these two areas for haptic interaction with the virtual objects but it does not provide insights into the problems of registration with the real scene or interactions between real and virtual objects. One of the reasons stated by Mine, Brooks, et. al. (1997) for the paucity of virtual-environment applications that have left the laboratory setting is the lack of haptic feedback.

Brooks, Ouh-Young, et. al. (1990) describes one of the first examples of a haptic interface used in virtual reality. This system uses a large sized telemanipulation arm that is driven by motors to give force feedback to the user. Molecular docking is the application area addressed by the project. The user operates in an immersive environment experimenting with finding positions for bonding two molecules together. The forces of repulsion and attraction for a bonding operation are correctly simulated and applied to the manipulator. The molecular experts using this system find that this addition of haptic sensation greatly improves their ability to discover novel compound arrangements. Two reported medical applications of haptic interfaces in virtual environments are for medical training. Ziegler, Brandt, et. al. (1997) describe a simulator for arthroscopic surgery that uses force feedback in the virtual environment to train surgeons in the procedure. Using the Rutgers Master II force feedback device Dinsmore, Langrana, et. al. (1997) built a virtual environment for training physicians to locate and palpate tumor masses.

The haptic device that is most suited for our augmented reality applications is the Phantom. Its applicability is due not only to its small size but also for the range of haptic feedback that is available. This tabletop device provides force feedback to the user's fingertip. Because of its size it is well suited for workspaces that fall into the

category of virtual-environment interaction that is “working within arm’s reach” (Mine, Brooks et al. 1997).

The demonstrations of the work of State, Hirota et. al. (1996) show interaction with virtual objects. Correct visual interactions occur when virtual objects move behind a real object. Using the metaphor of a magnetic finger, the user is also able to attach a virtual object to his fingertip and move it within the workspace. This is all done within a framework of hybrid position tracking that requires knowledge of the 3D location of fiducial points in the scene. There is neither haptic feedback nor dynamic interactions between the virtual and real objects. Using the registration technique of Uenohara and Kanade (1995), Yokokohji, Hollis et. al. (1996) demonstrate a haptic interface for an augmented reality system. They use a Puma 560 robot for the force feedback and track a small plane attached to the end effector of the robot. The entire real scene is draped in blue cloth to allow for easy chroma-keying of the video image of the user’s arm and hand. The augmented display merges a virtual cube located where the small plane was detected and live video of the user’s arm and hand. There is neither motion of the virtual object nor interactions between virtual and real objects in their system.

3.6 Limitations of the Previous Work

So far we have seen that most of the techniques used for creating and maintaining a correctly registered augmented view of the real world require some apriori information about the system. This information takes several forms and any particular approach requires one or more of the following pieces of information:

- sensing the position of the video camera viewing the scene,
- metric information about the intrinsic parameters of the camera, or
- the location of fiducial markers in the scene.

The most commonly used magnetic position sensing is susceptible to magnetic field disturbances generated by objects in the workspace and has response times that create unacceptable latencies in system operation (Byrson 1992; Azuma and Bishop 1995). Requiring metric calibration information about the camera adds the calibration step but also limits the flexibility of inexpensive video cameras that have zoom lenses since a change in focal length requires recalibration. In some application environments it may not be possible to know measured locations of fiducial markers and so any approach that relies on that could not be used. Military battlefield applications or, generally, unfamiliar outdoor environments come to mind when considering this limitation. And finally, a scant amount of work is reported in the literature for bringing the haptic technology used in virtual reality applications into augmented reality domains.

With those limitations in mind, in the next chapter we present our method of augmenting reality with affine representations. It has the advantage that it requires neither position sensing nor camera calibration to generate properly registered augmented reality images and, unlike the work of Uenohara and Kanade (1995), it merges full three-dimensional virtual models into the image of the real scene. In addition, we show how we incorporate haptic technology into the system allowing the user to naturally interact with virtual objects that exhibit dynamic behaviors.

4 Augmenting Reality using Affine Representations

4.1 Affine Representations for Augmented Reality

The approach to augmenting reality that we describe in the following sections is motivated by recent computer vision research that has determined structure for objects in a scene and the pose of the camera viewing it without knowledge of the object-to-world, world-to-camera and camera-to-image plane transforms. Koenderink and van Doorn (1991) and Ullman and Basri (1991) provide the following observation:

> *Given a set of four or more non-coplanar 3D points, the projection of all points in the set can be computed as a linear combination of the projection of just four of the points.*

We use this observation to create a global coordinate system in which we express the coordinate systems diagrammed in Figure 2. Additionally, we define this global coordinate system solely from the locations of visible features in the real scene with no knowledge of the intrinsic parameters and pose of the camera.

4.1.1 Affine camera approximation

Accurate determination of where a point on a virtual object will project in the video image is essential for correct registration of the virtual and live-video (Foley, van Dam et al. 1990; Tuceryan, Greer et al. 1995). In homogeneous coordinates, the projection, $\begin{bmatrix} u & v & h \end{bmatrix}^T$, in the video image of a 3D world point $\begin{bmatrix} x_w & y_w & z_w & w_w \end{bmatrix}^T$ can be expressed using the equation:

$$\begin{bmatrix} u \\ v \\ h \end{bmatrix} = \mathbf{P}_{3\times4}\mathbf{C}_{4\times4}\mathbf{O}_{4\times4}\begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix}. \quad (1)$$

The transforms $\mathbf{P}_{3\times4}$, $\mathbf{C}_{4\times4}$, and $\mathbf{O}_{4\times4}$ are shown in Figure 2 and are the camera-to-image plane, world-to-camera and object-to-world transforms respectively. Equation 1 assumes that object, world and camera coordinate systems are independently defined. Previous approaches to augmented reality have been based on an explicit determination of each of the transforms that relate the coordinate systems. Our approach represents the three coordinate systems in a single *non-Euclidean* coordinate system and expresses the projection operation with the equation:

$$\begin{bmatrix} u \\ v \\ h \end{bmatrix} = \Pi_{3\times4}\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}, \quad (2)$$

where $\begin{bmatrix} x & y & z & w \end{bmatrix}^T$ are the new coordinates for the point transformed from $\begin{bmatrix} x_w & y_w & z_w & w_w \end{bmatrix}^T$.

To fully exploit this simplification and the observation of Koenderink and van Doorn (1991) and Ullman and Basri (1991) we use *weak perspective projection* to model the camera-to-image plane transformation $\mathbf{P}$ (Shapiro, Zisserman et al. 1995) Under a weak perspective approximation, the projections in the image plane of 3D points are determined by first projecting the points along parallel rays orthogonal to the image plane. The entire image is then scaled by f / z_{avg} where f is the camera's focal length and z_{avg} is the average distance of the points from the image plane. This approximation is commonly seen in the computer vision literature and holds when the front to back depth of objects along the viewing direction is small compared to the viewing distance and the object's distance from the camera is large compared to the camera lens's focal length (Thompson and Mundy 1987).

4.1.2 Affine object structure

All points in this system are represented with an *affine representation* whose coordinate system is defined using the location of feature points in the image. This representation is invariant when an affine transformation, i.e. translation, rotation, non-uniform scaling, is applied to all points. Transforms caused by the motion of a

weak perspective camera viewing a scene will maintain this affine invariant representation. *Affine reprojection* or *transfer* (Barrett, Brill et al. 1992; Shashua 1993) is used to compute the projection of virtual objects placed into the real scene.

The affine representation for a collection of points, $p_0,\ldots,p_n$, is composed of: the *affine basis points* which are four non-coplanar points, one of which is specially designated as the *origin*; and the *affine coordinates* of each point that represent the point with respect to the affine basis points. The properties of affine point representation are illustrated in Figure 20.

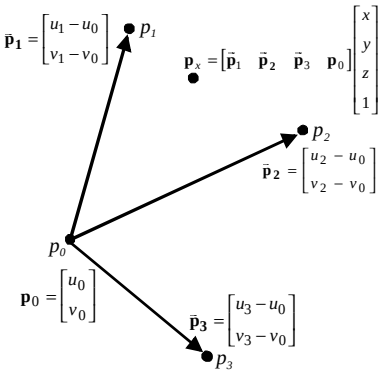


Figure 20 - Affine point representation

4.1.3 Affine reprojection

Property 1 (Affine Reprojection Property) *The projection,* $\begin{bmatrix} u_p & v_p \end{bmatrix}^T$, *of any point, p, represented with affine coordinates* $\begin{bmatrix} x & y & z \end{bmatrix}^T$, *is expressed by the equation:*

$$\begin{bmatrix} u_p \\ v_p \end{bmatrix} = \underbrace{\begin{bmatrix} u_{p_1} - u_{p_0} & u_{p_2} - u_{p_0} & u_{p_3} - u_{p_0} \\ v_{p_1} - v_{p_0} & v_{p_2} - v_{p_0} & v_{p_3} - v_{p_0} \end{bmatrix}}_{\Pi_{2\times 3}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} u_{p_0} \\ v_{p_0} \end{bmatrix} \tag{3}$$

where $\begin{bmatrix} u_{p_i} & v_{p_i} \end{bmatrix}^T, i = 0,\ldots,3$ *are the projections of the origin* p_0, *and the three other basis points,* p_1, p_2, *and* p_3, *that define the affine coordinate system. This can be equivalently expressed in homogeneous coordinates with the equation:*

$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} u_{p_1} - u_{p_0} & u_{p_2} - u_{p_0} & u_{p_3} - u_{p_0} & u_{p_0} \\ v_{p_1} - v_{p_0} & v_{p_2} - v_{p_0} & v_{p_3} - v_{p_0} & v_{p_0} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\Pi_{3\times 4}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (4)$$

Equation 4 provides an explicit definition for the projection matrix $\Pi_{3\times 4}$ seen in Equation 1 and defines the projection of a 3D point in any new image as a linear combination of the projections of the affine basis points in that image. Calculation of the projection of a point on a virtual object has been reduced to knowing the location of the projections of the affine basis points in a new image and the homogeneous affine coordinates for the virtual point. The affine basis points will be defined by visually tracking features in the scene and determining the projections in each new video image. The following property provides the technique for determining the affine coordinates of any 3D point.

Property 2 (Affine Reconstruction Property) *The affine coordinates of any point can be computed from Equation 4 if its projection in at least two views is known and the projections of the affine basis points are also known in those views.*

This results in an over-determined system of equations based on Equation 4. Given two views, $I_1, I_2,$ of a scene in which the projections of the affine basis points, $p_0,\ldots, p_3$, are known then the affine coordinates $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ for any point p can be recovered from the solution of the following equation:

$$\begin{bmatrix} u_p^1 \\ v_p^1 \\ u_p^2 \\ v_p^2 \end{bmatrix} = \begin{bmatrix} u_{p_1}^1 - u_{p_0}^1 & u_{p_2}^1 - u_{p_0}^1 & u_{p_3}^1 - u_{p_0}^1 & u_{p_0}^1 \\ v_{p_1}^1 - v_{p_0}^1 & v_{p_2}^1 - v_{p_0}^1 & v_{p_3}^1 - v_{p_0}^1 & v_{p_0}^1 \\ u_{p_1}^2 - u_{p_0}^2 & u_{p_2}^2 - u_{p_0}^2 & u_{p_3}^2 - u_{p_0}^2 & u_{p_0}^2 \\ v_{p_1}^2 - v_{p_0}^2 & v_{p_2}^2 - v_{p_0}^2 & v_{p_3}^2 - v_{p_0}^2 & v_{p_0}^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (5)$$

where $\begin{bmatrix} u_p^i & v_p^i \end{bmatrix}^T$ and $\begin{bmatrix} u_{p_j}^i & v_{p_j}^i \end{bmatrix}^T$ are the projections of point p and affine basis point p_j, respectively, in image I_i.

4.1.4 Affine projection matrix

One performance goal for an augmented reality system is the ability to operate in real-time which will often limit the photorealism possible in rendering the virtual objects. As a minimum the very basic operation of hidden surface removal (Foley, van Dam et al. 1990) must be performed to have a correct visualization of any three-dimensional virtual object. Hidden surface removal uses a front-to-back ordering of surfaces to determine visibility. The front-to-back ordering is obtained by assigning a depth to each rendered point that represents its distance from the viewpoint. The following two properties extend the familiar notions of "image plane" and "viewing direction" to the affine representation. This extension allows for the required front-to-back ordering of virtual surfaces and use of hardware-supported rendering via z-buffering. The image plane and viewing direction define a 3D coordinate system that describes the orientation of the camera. The graphics system can operate entirely within this global affine coordinate system and completely ignore the original object representation.

Property 3 (Affine Image Plane) *Let* χ *and* ψ *be the homogeneous vectors corresponding to the first and second row of* $\Pi_{2\times 3}$*, respectively. (1) The vectors* χ *and* ψ *are the directions of the rows and columns of the camera, respectively, expressed in the coordinate frame of the affine basis points. (2) The affine image plane of the camera is the plane spanned by the vectors* χ *and* ψ*.*

The unique direction in space along which all points project to a single pixel in the image defines the viewing direction of a camera under our model of weak perspective projection. In the affine case, this direction is expressed mathematically as the null-space of the matrix $\Pi_{2\times 3}$:

Property 4 (Affine Viewing Direction) *When expressed in the coordinate frame of the affine basis points, the viewing direction,* ζ*, of the camera is given by the cross product*

$$\zeta = \chi \times \psi. \tag{6}$$

Property 4 guarantees that the set of points $\{p + t\zeta, t \in \Re\}$ that defines the line of sight of a point p will project to the same pixel under Equation 3. The z-buffer value needed for hidden surface removal that is assigned to every point is the dot product $\begin{bmatrix}\zeta^T & 0\end{bmatrix} \cdot p^T$. The actual magnitude of this value is irrelevant: the important characteristic is that the front-to-back order of virtual points rendered to the same pixel is correctly maintained along the camera viewing direction.

The affine image plane and viewing direction vectors define a 3D coordinate system that in general will not be an orthonormal reference frame in Euclidean 3D space. Despite this, correct visible surface rendering of any point $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ defined in the global affine coordinate system can be performed by applying the transformation:

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} u_{p1}-u_{p0} & u_{p2}-u_{p0} & u_{p3}-u_{p0} & u_{p0} \\ v_{p1}-v_{p0} & v_{p2}-v_{p0} & v_{p3}-v_{p0} & v_{p0} \\ & \zeta^T & & 0 \\ & \mathbf{0} & & 1 \end{bmatrix}}_{\Pi_{4\times4}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{7}$$

where u and v are the graphic image coordinates of the point and w is its assigned z-buffer value.

$\Pi_{4\times4}$ has the same form as the viewing matrix that is commonly used in computer graphics systems to perform transforms of graphic objects. The structural similarity allows standard graphics hardware to be used for real-time rendering of objects defined in the affine coordinate system developed here. In our system, a Silicon Graphics Infinite Reality Engine is directly used to render virtual objects with hardware-supported hidden surface removal.

4.1.5 Virtual object placement

In order to use the affine projection matrix $\Pi_{4\times4}$ to render the virtual objects, those objects must be represented in the global affine coordinate system defined by the tracked basis points. The 3D object-centered Euclidean coordinate system that commonly describes a virtual object must be transformed to the common affine coordinate system developed in the previous sections. The calculation of this object-to-affine transform is done at runtime. In the simplest approach, the user interactively specifies this transform by placing a real object in the scene that defines a bounding box for the virtual object. The bounding box is defined by specifying the location of the *x*, *y*, *z* axes and the origin of the bounding box. In two separate views of the scene the user specifies these four locations. From these four point correspondences and the affine projection matrices, $\Pi_{\mathbf{4\times4}}$, computed in the two images, the affine coordinates for the points, $\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z, \mathbf{a}_o$, are computed using the affine reconstruction property (Section 4.1.3). Assuming that the virtual object's coordinate system has been scaled to a bounding box of unit size, the *object-to-world transform,* $\mathbf{O}_{4\times4}$, from Figure 2 can be calculated from the solution of

$$\begin{bmatrix}\mathbf{a_x} & \mathbf{a_y} & \mathbf{a_z} & \mathbf{a_o}\end{bmatrix} = \mathbf{O}_{4\times4}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 1 & 1 & 1 & 1\end{bmatrix} \quad (8)$$

In this case, the world coordinate system is the common affine coordinate system. The computed transform handles both the change to the common affine coordinate system and placement in the 3D scene.

The user can be further supported in the process of interactively placing virtual objects. By using results from stereo vision, the system can impose constraints on where the user is allowed to specify points representing physical locations in 3D space. Once the user specifies a point in one image, the *epipolar constraint* (Shapiro, Zisserman et al. 1995) determines the line in the second image on which the projection of this point must lie. The user specification of the point in the second image can be "snapped" to the nearest point on this epipolar line. Additionally, techniques for constraining points to be collinear or coplanar with physical edges and surfaces are available (Kutulakos and Vallino 1996a).

4.1.6 Virtual object rendering

The affine projection matrix $\Pi_{4\times4}$ will correctly render virtual objects for merging with the video image provided that those objects are represented in the global affine coordinate system. The projected pixel locations are computed in the coordinate system of the image plane of the feature tracking system. This 2D coordinate system needs to be registered with the 2D projection created when the graphics subsystem renders the virtual objects. There may be an aspect ratio adjustment and translation that needs to be computed. If this transform is computed correctly then, given the affine coordinates of a real point, it is possible to render a virtual marker that will exactly overlay the real point in the merged augmented reality image. This video-to-graphics transform, $\mathbf{T}_{gv}$, is computed by knowing the locations of three marks in both the graphics and video images. The graphics subsystem creates an alignment pattern that the tracker subsystem digitizes. Figure 21 shows the alignment pattern we use. After digitizing the image the tracker determines the location of the three alignment marks. The graphics subsystem provides the location of the fiducials in its window and the transform $\mathbf{T}_{gv}$ between digitized image frame and graphics frame is obtained by solving

$$\begin{bmatrix}\mathbf{g}_1 & \mathbf{g}_2 & \mathbf{g}_3\end{bmatrix} = \mathbf{T}_{gv}\begin{bmatrix}\mathbf{i}_1 & \mathbf{i}_2 & \mathbf{i}_3\end{bmatrix} \quad (9)$$

where $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ are the positions used by the graphics subsystem when rendering the alignment marks and $\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3$ are the corresponding positions in the feature tracker image plane. The complete set of transformation operations for rendering a virtual object are shown in Figure 22.

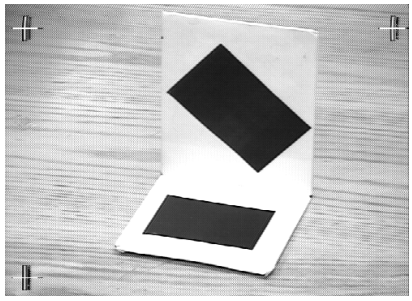


Figure 21 - Digitized alignment pattern

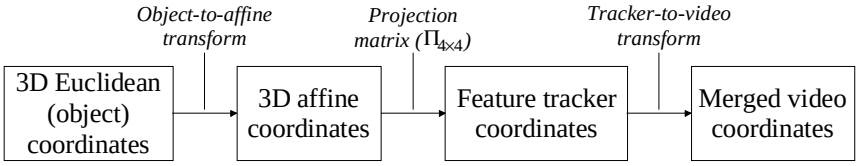
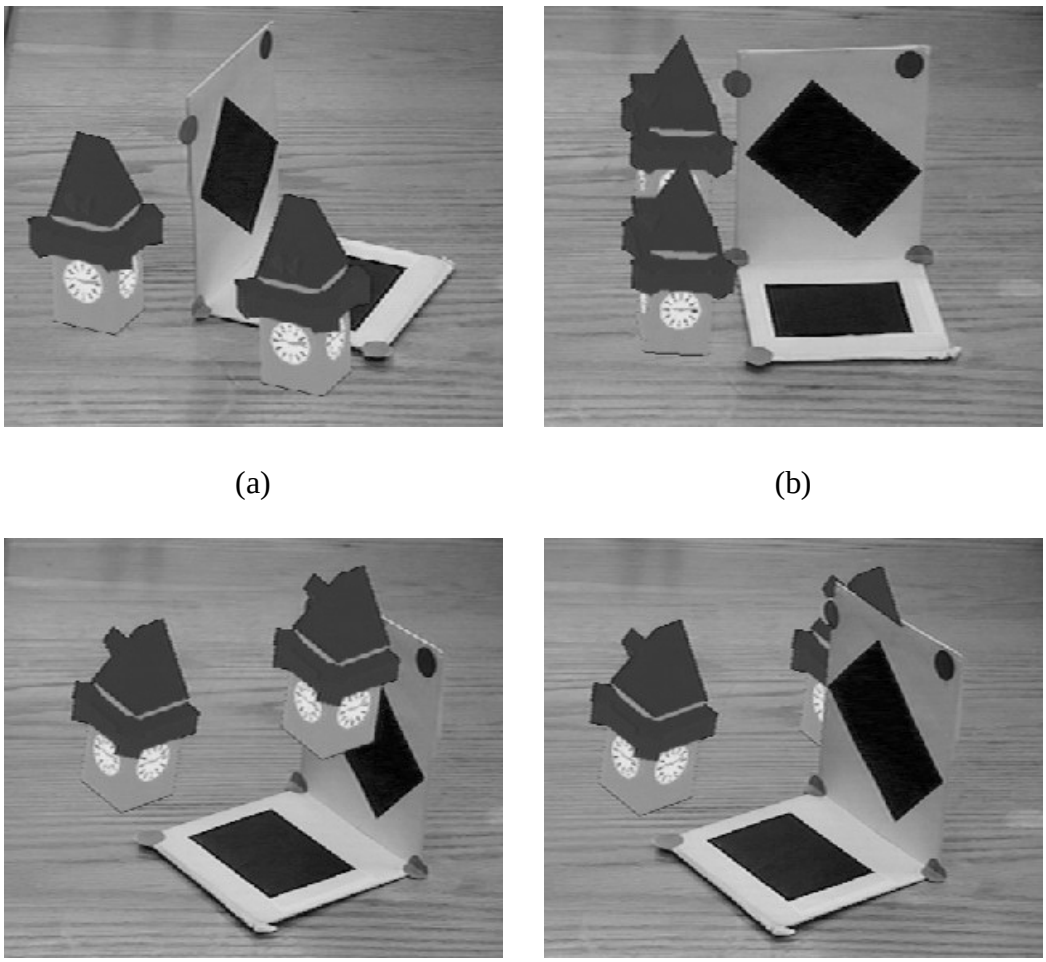


Figure 22 - Transforms for rendering a virtual object

4.1.7 Visual interactions between objects

In an augmented view of the scene visual interactions between real and virtual objects must be considered. The affine projection matrix $\Pi_{4\times4}$ will correctly handle hidden surface elimination within a virtual object (Figure 23a). It will also cause rendering algorithms to correctly occlude virtual objects that are behind other virtual objects (Figure 23b). Next, the visual interaction between real and virtual objects must be considered. At any given pixel in the augmented image the keyer controls whether the graphics image of a virtual object or live video image of the real scene is shown. Using the graphics as the foreground element, or key mask, a luminance keyer will display the graphics image at every pixel above the luminance key value. The live

video image is shown whenever a graphics pixel has a luminance value less than the key value. If the background regions of the virtual graphics image are rendered with a luminance at or below the key value then the live video is shown in all background regions and the virtual objects will occlude the video image in the areas where a virtual object is rendered.

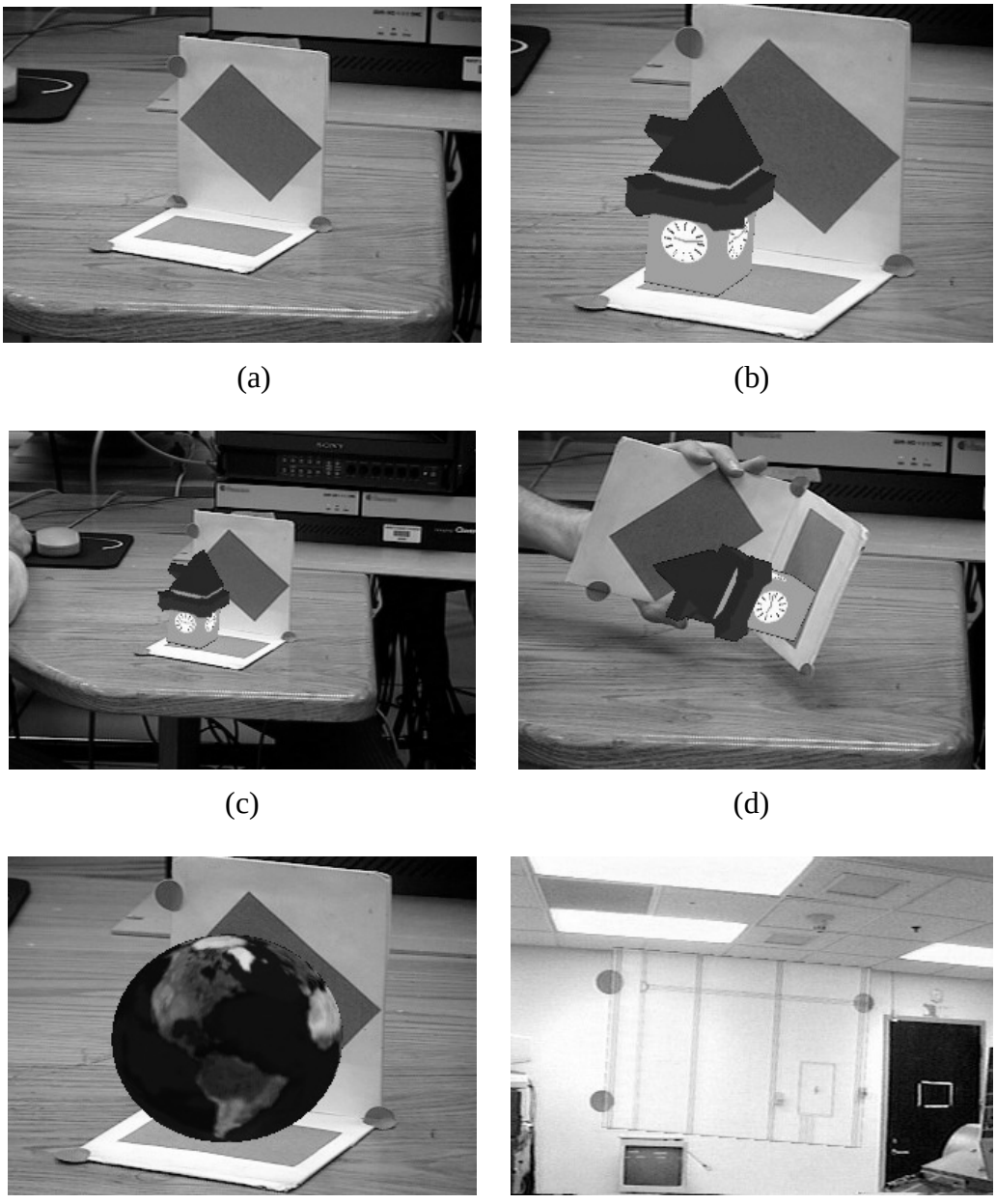


(a) (b)

(c) (d)

Figure 23 - Occlusion in augmented reality scenes

Hidden surface removal, where the hidden surface is on a virtual object does not occur when a real object occludes a virtual one (Figure 23c) because there is no information about the geometric relationship between these objects (Wloka and Anderson 1995). If an affine model of a real object is included as another virtual object and rendered in the background color the occlusions are correctly resolved by the keyer (Figure 23d). By defining the real object as a pseudo-virtual one the



(a) (b) (c) (d) (e) (f)

Figure 24 - Gallery of augmented reality images

geometric relationship to the virtual objects is defined. The correct hidden surface removal is performed by the graphics subsystem in the graphics image and by rendering in the background color the keyer performs the operation correctly also. A method for directly creating an affine model for a real object is described in (Kutulakos and Vallino 1996a).

Figure 24 shows a gallery of augmented reality images. The first image is of the real scene in our lab. We performed many of our experiments using the frame with four green feature markers. The rectangular regions on the vertical and horizontal sides of the frame are there only to provide the viewer with a perspective. Figure 24b shows a virtual clock tower merged into the real scene. Our method seamlessly handles changes in the focal length of the camera viewing the scene as Figure 24c demonstrates. Figure 24d shows a different orientation of the frame. The last two images in the figure show a virtual globe placed in the scene and a two-dimensional version of the system demonstrating an application in building construction or repair.

4.1.8 Animation

A desirable feature for an interactive augmented reality system is the ability to animate the virtual objects that have been added to the real scene. This is needed to show motion of the objects in the scene, whether the motion is independent or created by interactions with the user or other objects. Translations in the scene can be directly represented in the object-to-world transform $\mathbf{O}_{4\times4}$ by a multiplication with a translation matrix. This matrix follows the standard format used in computer graphics homogeneous transforms where the translation component is placed in the fourth column of the matrix. The transform is in the common affine coordinate system and requires that the translation be specified in that coordinate system. Translations represented in the Euclidean world coordinate system must be transformed into the common affine coordinate system. An example of translating a virtual cube in the affine coordinate space is shown in Figure 25.

Figure 25 - Virtual object translations

The standard computer graphics rotation matrix requires that the coordinate system be ortho-normal. This is, in general, not true of the common affine coordinate system used to represent the virtual objects. Rotation operations can not be directly performed via multiplications with standard rotation matrices. A method for computing arbitrary rotations in the affine coordinate system is described in (Kutulakos and Vallino 1996b). This method synthesizes the transform for an arbitrary rotation by first being shown examples of three rotations. This generates three rotation axes and any arbitrary rotation about those three axes can be composed into a single rotation transform. With judicious choice of the three rotation examples the user can compose rotations about meaningful axes in the real scene. The rotations demonstrated for the work in this thesis do not use the technique just described. Instead, we compute the rotations in the Euclidean coordinate system defining the virtual object and then transform the result to the common affine coordinate system by the object-to-world transform, $\mathbf{O}_{4\times4}$. The advantage of this method is that we can apply standard rotation transforms. The disadvantage is that the rotation can only be applied within the Euclidean coordinate system in which the virtual object is defined. We do not perform any arbitrary rotation about an axis in the common affine coordinate system. That requires the more general approach discussed just prior.

4.2 Static Registration

Static registration in an augmented reality image of a real scene indicates how well the system localizes a virtual object in the three-dimensional world when no motion exists in the system. This is a measure of the steady state error in the system at any particular viewpoint. Augmented reality systems must also be concerned with dynamic errors and the causes and compensation for those errors are discussed in Section 4.3. In augmented reality systems that are based on position measurements one of the primary contributors to static registration errors is the position sensor. Magnetic sensors, such as the Polhemus sensor, are known to provide systematic errors (e.g. non-uniform position measurements in many workspaces due to interference from metal objects in the workspace) and statistical errors (e.g. non-repeatable jitter) (Byrson 1992). A system based on affine representations trades this source of static error for errors generated by noise in the localization of feature points and violations of the affine approximation to the camera's perspective projection operation.

We assessed the limitation of the affine approximation empirically with the following experiment. The image projection of seven vertices on a physical object, a small rectangular box, served as the ground truth. We manually identified the location of these projections in images taken from multiple camera positions. Four of these points define the common affine coordinate system and associated affine projection matrix for each image. We computed the affine coordinates for the remaining three

points from their projection in two images using the Affine Reconstruction Property. The predicted projections of these points were computed in each of the other images by means of the Affine Reprojection Property. As the camera distance to the object increased, we increased the focal length of the camera zoom lens in order to keep the object's size constant and the registration errors comparable. The camera followed a roughly circular path around the box in a horizontal plane at distances up to 5 meters from the box. Figure 26 shows the camera positions used for computing the static registration errors. The positions are expressed in units of the common affine coordinate system. The affine coordinates for the test point were computed from two images near the position $\left(-1.5\times10^4, 0.5\times10^4\right)$.

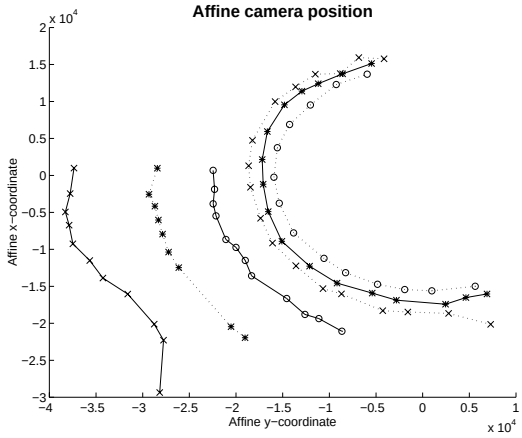


Figure 26 - Positions of the camera in affine space during static registration testing

Over the range of motions considered, the static registration error remains within 15 pixels in the 512×480 images we processed as shown in the plots of errors in Figure 27. The data in an error plot corresponds to the positions in Figure 26 with the same line style. These results show, as expected, that the affine approximation to perspective leads to errors as the distance to the object decreases (Boufama, Weinshall et al. 1994). A projective invariant representation has the potential to reduce the error at the shorter distances to the object (Section 5.1.2).

4.3 Dynamic Registration and System Latency

Latency in an augmented reality system is a major concern. Several research efforts (Azuma and Bishop 1994; Holloway 1997) have addressed the issue. System latency causes the images or haptic feedback to lag behind or "swim around" in the real scene. User motion triggers a transient response in which the motion of the

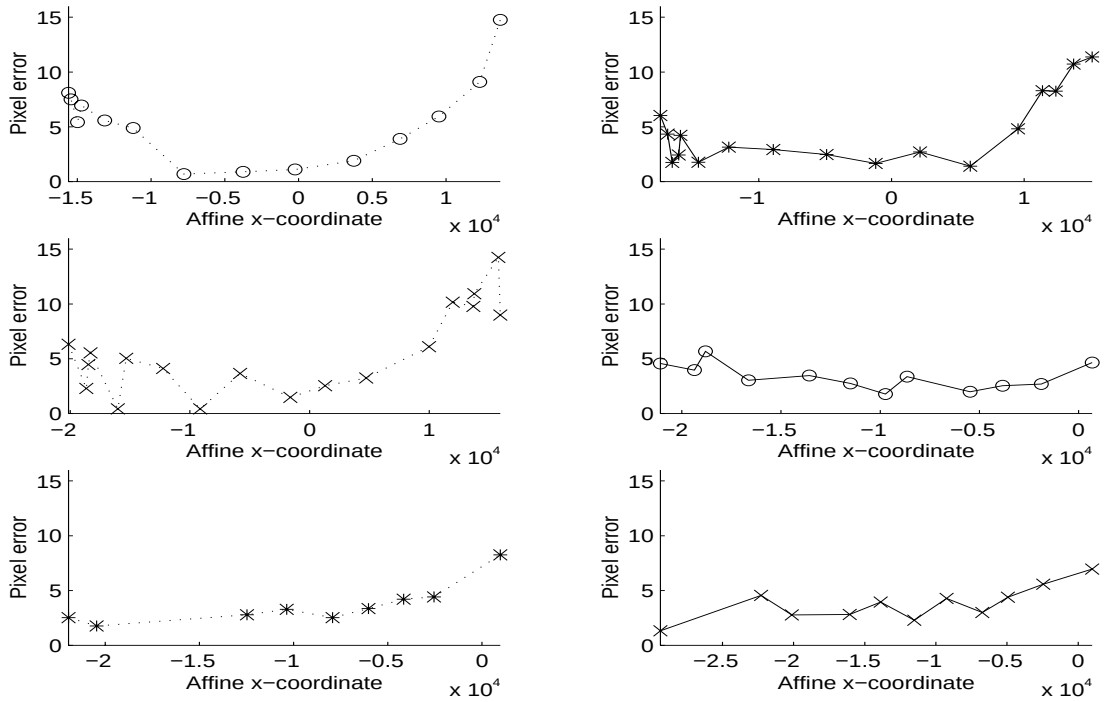


Figure 27 - Static registration errors at varying camera positions

virtual objects lags behind the motion of the real scene. The final steady state error at the new viewpoint is the static registration error for that location. Under continuous motion, the system latency results in the dynamic registration error. Since the problems of system latency are most noticeable during more rapid changes in viewpoint this, unfortunately, is also the time when the user most notices the resulting discrepancies between his visual and proprioceptive inputs. System latency strongly undermines the user's sense of immersion in an environment that is seamlessly composed of real and virtual objects. It is the largest contributor to dynamic registration errors (Holloway 1997). Experiments by Ellis, Bréant et. al. (1997) show that latencies of greater than 50 msec. adversely affect the user's performance of a path following task.

4.3.1 Contributors to system latency

There are several factors that contribute to these dynamic registration errors (Holloway 1997). At the core is whatever processing must be performed to process input viewpoint or position information, generate the new projection matrix and then render the virtual objects. In augmented reality systems that are based on position measurements those sensors (Polhemus Corporation 1996) have traditionally had update rates that were not fast enough for high fidelity virtual environments (Adelstein, Johnston et al. 1992). This is another instance where more expensive

equipment, such as the motion control rigs used to track camera motion for special effects in motion pictures, can be used if the cost containment goal is ignored. The newer and faster position sensors are making this less of a problem. Systems that use video cameras to view the real scene have an inherent one frame (1/30 sec.) delay getting a frame of video into the framebuffer. A monitor-based or video see-through display can also introduce a frame of delay in the keying operation depending on the implementation of the video keyer. The graphics system's pipelines used for rendering the virtual objects are often optimized for throughput and do not attempt to minimize latency (Foley, van Dam et al. 1990). These delay elements are not all serial, in that some of the processing of position information and/or video data will overlap with the video operations. For the system implementation described in Section 4.5 the processing time latencies are in the 70 - 90 msec. range.

4.3.2 Overcoming latency

There are multiple techniques that an augmented reality system uses to counteract the effects of latency. One technique—or non-technique really—is to simply wait for the next generation of processor to speed up whatever part of the operation is processor-bound. In video see-through systems a second approach compensates for the delays by matching the timing in the two video streams being merged to form the augmented image. The stream output by the graphics systems is several video frames behind the video image of the real scene. If a similar number of frames of delay is placed in the real video stream then the augmented image will maintain correct temporal registration. Unfortunately, this results in delaying the entire image the user sees. The haptic system registration must be delayed a similar amount. The last technique that researchers have tried is to model the user motion and predict true position based on this model (Zikan, Curtis et al. 1994; Azuma and Bishop 1995). These prediction algorithms require information about the user's head position. That data is not available in our system based on affine representations (Section 4.1). The only data we use as input to the system is the projections in the image plane of an uncalibrated camera of feature points in the real scene.

Since modeling head motion is not possible, we performed experiments to quantify the registration error and to see if simple predictors applied to the feature point data would bear fruit. Figure 28 shows the apparatus that we used during testing: it consisted of a high contrast white dot against a black background. We initialize the test by determining the projection for the white dot painted on the tip of the nail attached to the frame. We compute the affine coordinates for this point using the Affine Reconstruction Property after the user has manually specified the projection of the point in multiple images. Two images is the minimum needed for calculating the affine coordinates of a point. In these experiments, however, we found that a least squares approximation using the location of the point in up to 8 images reduced the initial static registration error to under 1 pixel. Computing the affine coordinates from

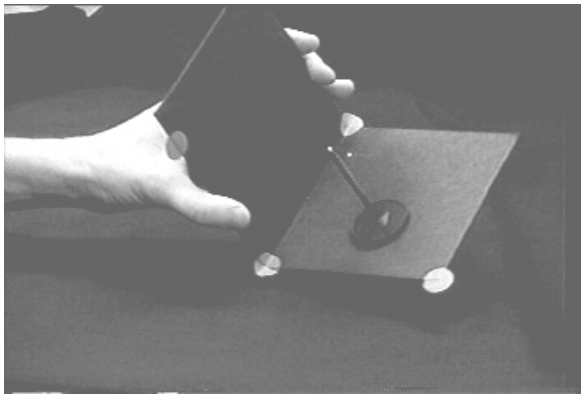


Figure 28 - Dynamic registration error experiment

the minimum two views resulted in errors of 3 to 5 pixels. (This result also has implications for the general method of defining the placement and object-to-affine transform described in Section 4.1.5. Our experiments showed that computing the affine coordinates of the bounding box of the virtual object from the minimum two views often yielded inaccurate rendering of the virtual objects. A more robust computation should be made from multiple views.) In each video frame of the test we projected a virtual marker at the location determined by the test point's affine coordinates and the current affine projection matrix using the affine reprojection property (Section 4.1.3). During the experiments we also tracked the real point in each video frame throughout the sequence. We used simple thresholding followed by blob coloring and centroid computation to track the point in each video frame. Finally, we measured the registration error as the Euclidean distance in the merged image between the virtual marker and the real point.

We recorded a test sequence approximately 60 seconds long and replayed it for all of the experiments. The lag between the real point and its reprojection was clearly seen during test runs. We performed tests with no filtering of the feature points, α-β filtering, and α-β-γ filtering (Kalata 1984; Brown 1994). The two filters are Kalman filters with plant models that make a constant velocity and constant acceleration assumption, respectively. A single parameter called the tracking or maneuverability index controls filter operation. This generalized parameter is proportional to the ratio of the position uncertainty due to the target maneuverability to the uncertainty due to sensor measurement. Larger values of this index indicate a target that is highly maneuverable indicating that the position measurements should be more trusted even when they seem to vary widely. Conversely, smaller values indicate more uncertainty in the measurement of position from the sensors and a higher degree of faith should be placed in the values that the system model generates. The filters were run independently on each feature point. No additional constraint was placed on the data such as the requirement that an affine structure and motion to be maintained throughout the sequence (Carceroni, Harman et al. 1998). Adding this constraint could potentially improve the filtering operation.

During the initial tests, we placed a second marker in the graphic image to assess how well our simple algorithm was tracking the real point in the video frame. This mark indicated the detected location of the test point. We observed that this marker lagged the real test point in the image though not as far as the reprojected virtual point lagged. During the offline data analysis the location of the test point in future frames is available and we used this information when computing the error distance. At each frame in an experimental run, we calculated the error as the Euclidean distance between the location of the reprojected virtual marker and the detected position of the real point at the next time interval. Considering the video hardware that adds at least a one frame delay before a frame of video is available to the tracker to find the real point this is a reasonable adjustment. The lag in our system has been measured to be approximately three frames (Section 4.5.1). We observed the tracked point to qualitatively lag in the augmented image about half the amount of the virtual point. This qualitative observation combined with our measured latency provides further justification for the choice of a one frame adjustment as part of the data analysis.

Figure 29 shows the plots of mean pixel error for two filter combinations.

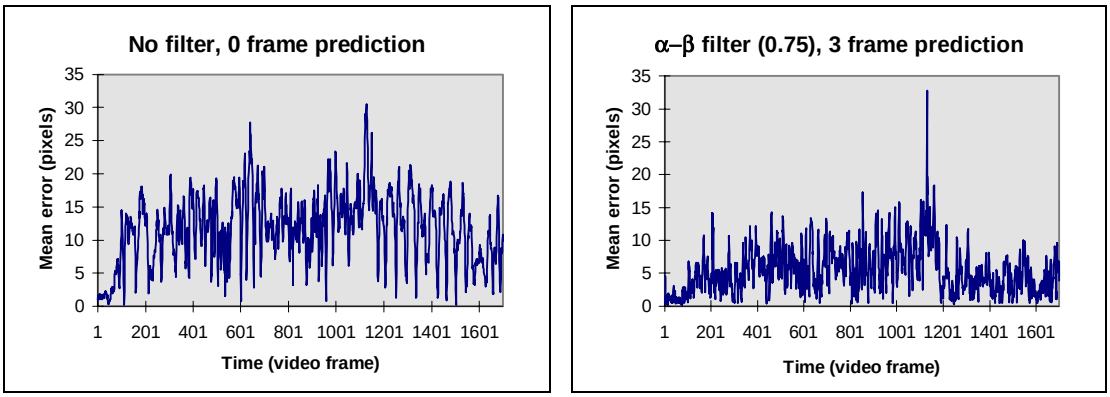


Figure 29 - Sample registration error data

The mean of the Euclidean distances between the position of the target point in the video frame and the reprojection of its affine coordinates for several filter configurations are given in Figure 30. One significant trend the reader should note in the data is that the mean error consistently decreases for forward prediction values of 2 to 3 frames of video. This data confirms the values for latency that we measured in the system. It also matches our experience when running the system which from a visual inspection indicated that 3 frames of prediction gave the best result qualitatively. The specific method of filtering does not have a significant quantitative effect on the registration error compared to the effect of the prediction value selected. This may be due to the plant assumptions of constant velocity (α–β) and constant acceleration (α–β–γ) in that neither of them accurately models the non-linear motion of the real test point through the test sequence. The slightly better results of the α–β–γ filter at

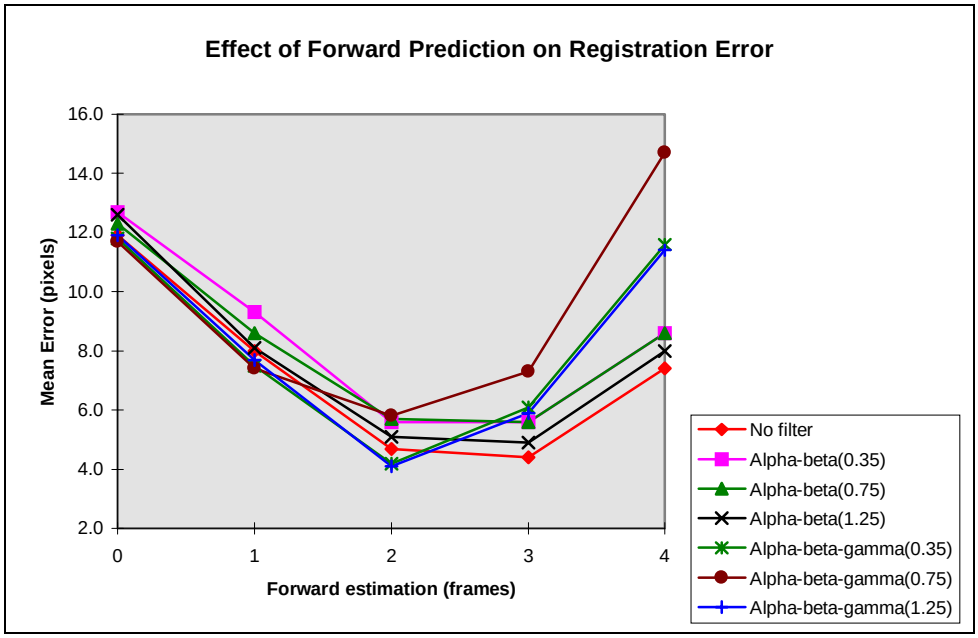


Figure 30 - Effect of forward prediction on Registration Error

smaller prediction values may indicate at better capability for this filter to track the test point during abrupt changes in direction. Because of the greater susceptibility to noise in the input data when computing accelerations, the α–β–γ filter performs much worse at larger prediction values. We also observed during our experiments that qualitatively this filter was less stable and prone to causing large stretches and shears in rendering the virtual objects. With no filtering on the input feature points, the addition of forward prediction qualitatively added an objectionable jitter in the image of the virtual objects. This was due to noise in the localization of the feature points by the tracker. Our addition of filtering the feature point data prior to computing the forward prediction reduced the virtual image jitter. Even at what was qualitatively considered the best combination (α–β, 0.75, 3 frame prediction) a small number of informal observers felt that the smaller registration error albeit with a jerkier motion was not an improvement over the original smoother motion with the larger lags. These observers considered that it was easier to adapt to and interpret the smooth motion of the objects than the jerkiness resulting from the forward prediction.

4.4 Haptic Interface

The Phantom provides the haptic interface for our interactive augmented reality system. This haptic device resembles a small robot arm (Figure 19) that is driven by motors to provide forces at the end effector, which is a small thimble surrounding the user's fingertip. (A stylus end effector is also available but was not

used as part of our thesis work.) Using the supplied GHOST library, we define a haptic scene in a manner analogous to defining the graphic scene used for generating the virtual objects. A tree data structure stores the haptic scene information similar to the way in which OpenInventor maintains the graphic scene information. In fact, the VRML format files that we use to define our virtual objects can also define the haptic scene. Nodes in the haptic scene tree are classified into several categories. The classifications (including the equivalent nodes in a graphics scene graph given in parentheses) are:

- Phantom device node (camera node),
- shape nodes that are haptically rendered (point, sphere, spline shapes),
- effects such as dynamics, buzzing, surface characteristics (material properties, textures).

All these nodes are controlled through the C++ GHOST library provided with the Phantom. The haptic scene is described in the coordinate reference frame of the Phantom.

4.4.1 Haptic-graphic interaction

To operate our augmented reality system in a WYSIWYF (what you see is what you feel) mode, the transform between the haptic coordinate system and the common affine coordinate system in which all of the virtual graphic objects are described must be established. Once we establish that relationship it is possible to exchange position information between the haptic and graphic scene. We need this exchange of position information so that our system can instruct the Phantom to generate haptic feedback to the user that is appropriate for the user's current interactions with the virtual objects.

Our system computes the Phantom to affine transform, $\mathbf{T}_{ap}$, when it initializes Phantom. As part of this computation the user moves the Phantom end effector to four points in the workspace for which affine coordinates are known. In our case, these points are the four feature points that define the common affine coordinate system. At each of the four points, the system records the position in the Phantom coordinate system and after all four positions have been marked solves the following equation the transform, $\mathbf{T}_{ap}$:

$$\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4\end{bmatrix} = \mathbf{T}_{ap}\begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4\end{bmatrix} \tag{10}$$

where $\mathbf{a}_i$ and $\mathbf{p}_i$ are, respectively, the affine coordinates of the points in the workspace and their corresponding haptic coordinates. The system also computes the inverse of this transform.

4.4.2 Haptic demonstrations

We demonstrated several different interactions with virtual objects. The first displays a virtual globe in the augmented image. When the globe is stationary (Figure 31a) the user can move their finger over the surface and clearly feel the shape of the globe as a sphere. The haptic output or display of the Phantom is properly registered with the visual display of the virtual objects. As the active point of the Phantom crosses over a land mass the user feels a vibratory sensation to indicate the land. Over water areas the surface is compliant giving the user a feeling of sinking below the water surface. The user can very easily trace the coastline of land masses by feeling where they exist. Another demonstration spins the globe on its axis. The system tracks the location of the active point of the Phantom on the globe surface so that haptic feedback changes in response to both user motions and the spinning of the globe. The third demonstration with the globe (Figure 31b) allows the user to rotate the globe. Whenever the user is in contact with the surface of the globe that point of contact on the globe surface will stay locked to the user's fingertip. The globe rotates in response to any motion of the finger.

(a) (b)

Figure 31 - Haptic interactions with a globe

A tight coupling exists between the graphics scene and haptic scene as Figure 32 depicts. For the globe demonstrations, a Mercator projection map of the world is divided into a 42×21 grid, haptic textures assigned to each square depending on its land/sea ratio and this haptic texture description is remapped onto the sphere. At each cycle of operation the system obtains the current Phantom position and transforms it into the common affine coordinate system. This affine point is then transformed into the Euclidean coordinate system of the virtual object using an inverse of the object-to-affine transform described in Section 4.1.5. The location of this point in the texture map of the globe surface determines the appropriate haptic response. The system sends the correct commands to the Phantom to generate the feedback at the user's fingertip.

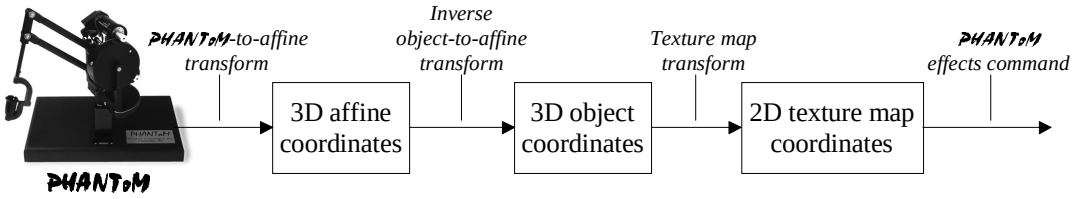
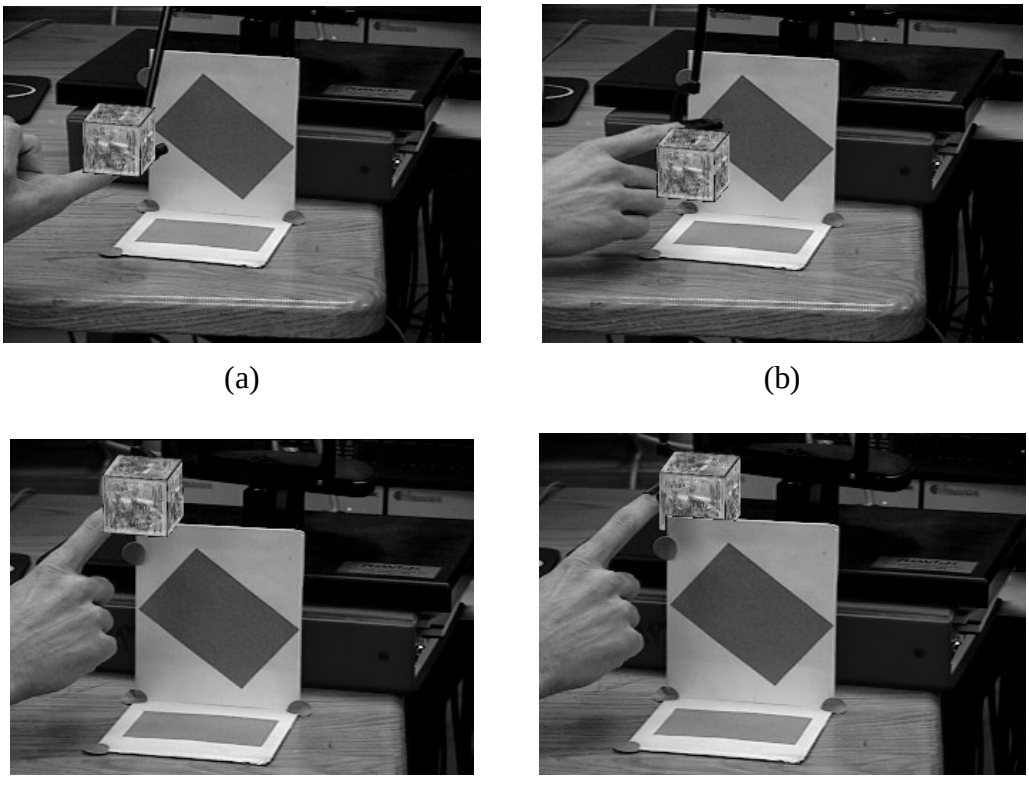


Figure 32 - Phantom-graphics coupling

4.4.3 Haptic interactions between real and virtual objects

The next demonstration is with a cube virtual object. In Figure 33a,b, the user is moving the cube around by lifting it with a finger. In the first case, he applies the lifting force from underneath the cube. The second situation provides the user with a "magnetic" touch active from above the cube. When the user presses the active point of the Phantom down on top of the virtual object a click is felt and from that point



(a) (b)

(c) (d)

Figure 33 - Interactions with a virtual cube

forward the cube is attached to the user's finger. The user releases the cube by placing it on a surface and pressing down slightly. This simulates the push-to-lock, push-to-release latches that are found on some cabinet doors. The vertical part of the frame is defined in the graphics scene so that occlusion between that real object and the virtual objects takes place as described in Section 4.1.7. This object is also described in the haptic scene so that haptic interactions take place between real and virtual objects. In Figure 33c, the user has just rested the cube on top of the vertical wall. It remains suspended in that position until moved by the user. Figure 33d shows the cube disappearing behind the vertical wall after being pushed slightly by the user. It is also possible for the user to play a game of "cube ball" by resting the cube on the table surface and knocking it into the vertical wall. The cube rebounds off the wall back to toward the user's finger. In all these demonstrations with the cube, the system simulates gravity so that when the user lifts or slides the cube the haptic feedback includes the sensation of the weight of the cube.

4.4.4 Foreground detection for improved visual occlusion

The images in Figure 31 and Figure 33 show one strong limitation of haptic interactions in this augmented reality system. This limitation is that the Phantom provides a very compelling sense of touching virtual objects at the user's fingertip but the visual image the user sees in the augmented display is not as compelling. The primary reason for this is that proper visual occlusions are not occurring between the virtual objects and the user's hand. Occlusion is one of the stronger depth cues the human visual system uses. In these examples the virtual objects always occlude the user's hand even when the hand is interacting with a front surface of the object. We noticed this limitation immediately when the haptic interface was added to the system and now a red marker, defined as an object in the virtual image, represents the active point of the Phantom end effector. This "visible virtual fingertip" is defined in the common affine coordinate system and thus exhibits the proper visual interactions with the virtual objects. It facilitates the user's interaction with the virtual objects by being a substitute for the cues that visual occlusion normally provides.

The discussion in Section 4.1.7 describes the method used for proper occlusion between real and virtual objects. This method models the real object as a virtual object rendered below the luminance key value. If this model of the real object is the front most virtual object then that region of the key signal (Section 3.1.1) matches the key criterion and video from the real scene displays in the region. This maintains the correct three-dimensional relationship between the real and virtual objects. To carry this forward with our haptic subsystem requires the Phantom mechanism to be defined as a graphic object with actual joint angles monitored during operation controlling its configuration. However, adding the Phantom as a virtual object still does not provide for occlusion of virtual objects by the user's finger and hand. One could consider making an approximate model of a finger and hand but the problem of determining its

orientation in three-dimensional space remains unsolved since the Phantom end effector does not have instrumentation to report its orientation. The stylus end effector is instrumented for orientation but that adds the need to model the stylus and encoders.

Borrowing a technique used on the Pfinder project at MIT (Wren, Azarbayejani et al. 1997) for detecting humans moving through a room, we experimented with the use of foreground/background segmentation in the 2D video image to identify areas of the image where occlusion by a real object should take place. The technique initially analyzes the workspace scene gathered over a specified number of video frames and gathers mean and covariance statistics on the YUV values of the video signal. We assume that the real scene contains objects that will always be present while the system is operating. The analysis subsamples the input image by a factor of 8 in order to maintain a 30Hz processing rate during operation. In each block of 8×8 pixels a single pixel in the center of the block represents the color information for that entire block. After initialization, when the system is running, the YUV pixel value of the same point in each subsampled block is checked to determine if its color is from the population described by the statistics for the same 8×8 block. If the color value is not from that population the block is marked as a foreground block. The system uses a 90% confidence level for the statistical test and the user has the capability to modify the threshold at runtime. The system processes video frames at runtime at approximately 15 Hz. The user can select one of two different statistical measures. The first gathers mean and covariance statistics on all three individual components in the YUV color space. This method has problems when the overall intensity of an area decreases due to shading. The statistics identify the shadow region as a foreground area. The second statistic method we implemented gathers mean and covariance statistics on only the UV chrominance components of the video signal normalized by the luminance Y. This normalized metric is more robust to these shadow effects. Next, we describe the method we use to obtain the correct visual interactions between our virtual objects and the real objects that generated the foreground blocks.

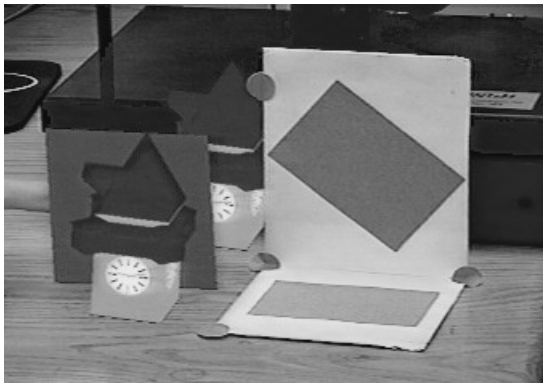
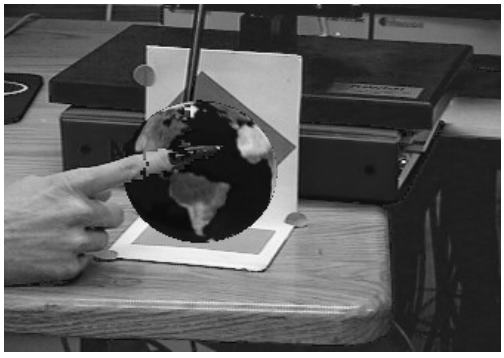


Figure 34 - Foreground mask plane

Earlier, this section described how a virtual fingertip can track the position of the Phantom end effector in the three-dimensional affine coordinate system. If we extend the concept of this fingertip to be a plane large enough to cover the entire scene (shown in Figure 34 only partially) we have a plane that cuts through the three-dimensional scene. The orientation of this "foreground" plane is taken to be perpendicular to the viewing direction (Section 4.1.4). We attach this plane to the active point of the Phantom and combine this plane with the information about the regions that were found to be statistically different than the same region in the image of the workspace scene. We assume that all statistically different regions represent real objects that have entered the workspace and are located in affine space at the depth of the foreground plane. This assumption provides the mechanism to position the statistically different blocks, originally detected in the 2D image plane, in three-dimensional affine space.

To get the proper visual interactions we texture map the foreground plane with the detected foreground information. This texture map exactly matches the blocks and subsampling factor that is used during the initial background statistics generation. Figure 35a shows the foreground plane with detected areas of foreground activity. Processing the entire video image each frame to find the foreground blocks takes time.

(a) (b)



(c)

Figure 35 - Foreground detection for visual occlusion

In order to maintain a 30 Hz processing rate, a subsampling factor of 8 must be selected. The overall goal here is to find areas where the live video image should occlude a virtual graphic object. We only need to make this determination in pixel areas where there are projections of virtual graphic objects. Restricting the foreground checks to these areas speeds up processing and allows a smaller subsampling factor of 4. We use the projection of the bounding box of all virtual graphic objects to define the region in which we perform foreground detection. The foreground mask in the area where graphics are present is shown in Figure 35b.

We want the virtual graphics image to key the live video into the final augmented image in all the 8×8 foreground block regions that are closest to the camera in affine space. Remember that we make the assumption that all detected foreground activity, i.e. a real object that has entered the workspace, is located at the affine depth of the Phantom virtual fingertip. For the virtual image to key the live video in a particular region the luminance of the virtual image in that region must be below the luminance key value. In the texture map applied to the foreground plane we render a detected foreground block as an opaque block below the luminance key value. If the active point of the Phantom—and by connection the foreground plane with its texture mapped foreground blocks—is in front of all virtual objects the opaque foreground blocks will generate a region below the luminance key value in the final virtual image. In contrast, when virtual objects occlude regions of the foreground plane those objects will appear in the final virtual image and not key the live video in the augmented image. In areas where we are not doing foreground detection or have detected no foreground activity we need the foreground plane to have no effect on the final virtual image or be transparent. We use the common computer graphic technique of alpha channel blending (Foley, van Dam et al. 1990) to accomplish this. Alpha, commonly a fourth component—along with red, green and blue—in the specification of a computer graphic color, is the inverse of transparency. An alpha value of 1.0 is completely opaque, i.e. it blocks any objects further in depth, and 0.0 is completely transparent, i.e. it is as if the object was not present in the scene. We set alpha to 1.0 in the foreground blocks and 0.0 in all other regions to get our desired result. The foreground plane is texture mapped as transparent in regions with no foreground activity and opaque below the luminance key value in the detected foreground regions. The final augmented image (Figure 35c) shows the user's finger properly occluding the virtual object that it hides.

Our approach uses a plane in the virtual object space to represent where motion in the scene is taking place. This assumption of all motion occurring in a plane in space is a gross simplification of reality. To avoid making this assumption requires information about the depth along the viewing ray for the object projected to each pixel in the image. There are two augmented reality systems (Wloka and Anderson 1995; Kanade 1996) that use depth maps for resolving visual occlusions. Instead of having a plane as a virtual object representing the real objects that entered the workspace, these two systems create what can be visualized as a flexible sheet in the virtual object space that conforms to the shape and location for all the visible real

objects. Only virtual objects that are rendered closer to the camera than the surface of this sheet are visible in the final augmented image.

4.5 System Implementation

A system that creates an augmented reality interface is a tightly connected set of components operating in real-time. Figure 36 shows a system diagram of our interactive augmented reality system. This section describes the implementation of our

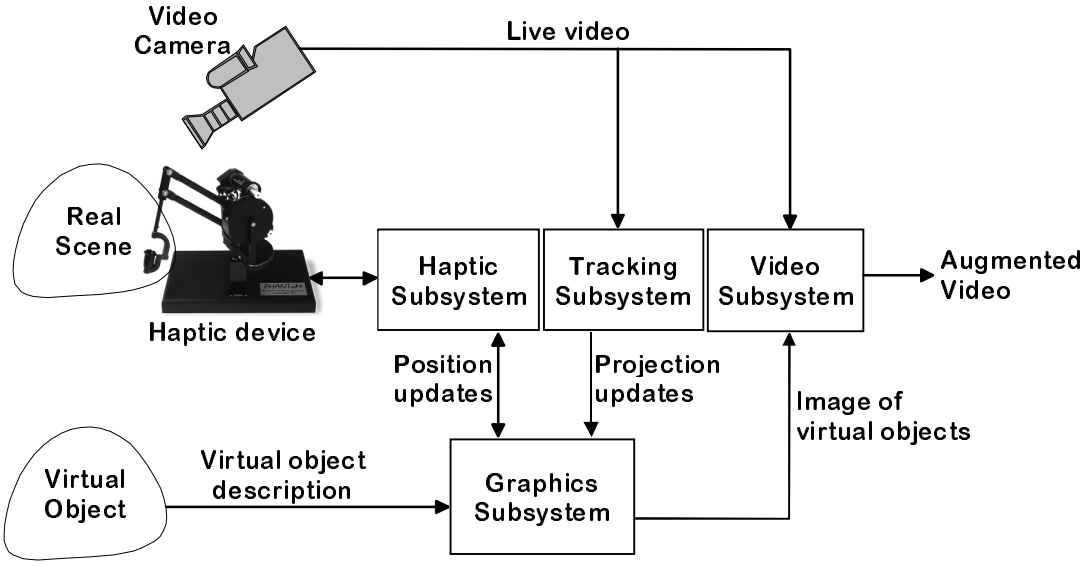


Figure 36 - System components of an interactive augmented reality system

augmented reality system based on the techniques detailed in the previous sections of this chapter. The following sections include details on the hardware we use and a description of the software architecture.

4.5.1 Tracking subsystem

The tracking subsystem performs all of the image processing required to track the feature points that define the common affine coordinate system. The input to the subsystem is the video signal from the camera viewing the real scene. The output is the affine projection matrix updated for the current video frame.

An augmented reality system that is of general utility requires operation in a natural setting. From this, it follows that a system built on our method of affine

representations requires tracking of features in a natural setting. To date we have greatly simplified the tracking problem to minimize the effort required in that regard. If these techniques are to be moved out of a laboratory setting a significant effort is needed to incorporate other more robust feature tracking algorithms. We discuss this in Section 5.1.4.

We implemented the system with two different trackers. The original, a region based tracker, runs on a multiprocessor SPARCstation. It requires two high contrast regions in the scene. The user initializes the tracker by selecting a pixel within each region. A region growing algorithm determines the outline of the region which is known to be a rectangle. Several points on each edge are tracked through the video frames. In each frame, the tracker determines the actual edge by doing a least squares fit of a line to the points for that edge. The "tracked" feature points are the corners of the two rectangles. The tracker determines the corners by intersecting the lines defining each edge. The axes of the common affine coordinate system are computed using the decomposition method of Tomasi and Kanade (1992). This tracker uses more than the minimum four feature points needed to define the common affine coordinate system and thus has some robustness against noise in a least squares sense.

We used our more recent tracker—a color blob based tracker—for most of the experiments in this thesis work. The change to a color tracker allows features to be placed in a somewhat more natural setting by applying colored dots in the work areas. The color tracker is based on a Datacube MV200 with Digicolor digitizer for input that is hosted on a Sun 3 machine. This image processing computer communicates with the rest of the augmented reality system via both Ethernet using UDP datagrams and a serial line connection. Configuration commands are sent to the tracker over the Ethernet connection. Feature data centroids are communicated back via either Ethernet or a serial connection. The choice of method is a compile-time option in the code for both the tracker and graphics subsystems. Because of variations in

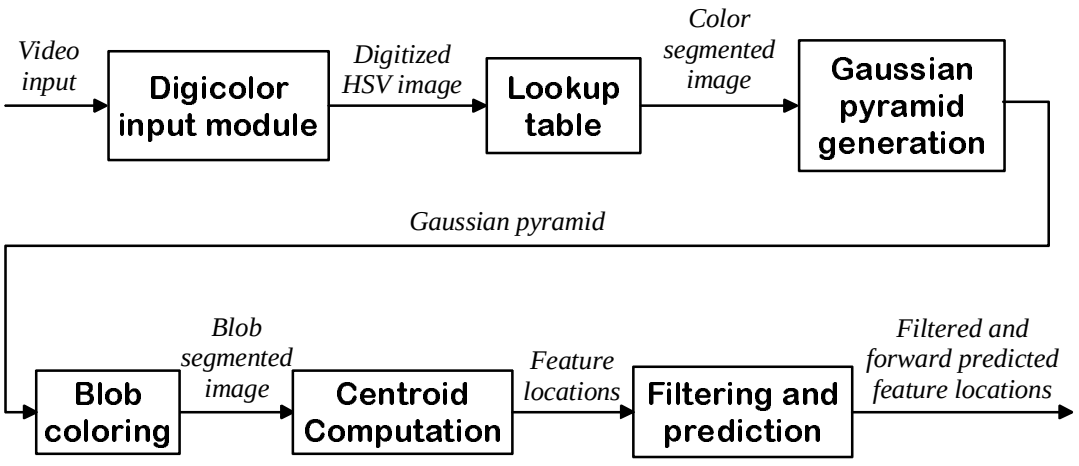


Figure 37 - Color tracker block diagram

congestion on the house Ethernet that the system is on, we found a small improvement in system latency when we switched communication to the serial line connection. Figure 37 is a block diagram for the blob tracking algorithm.

The Digicolor input module digitizes the video signal and codes it into the hue-saturation-value color space. The ability of the Digicolor to create a digitized image in this color space is crucial to frame rate color segmentation. The 8 bit hue and saturation components are the input to a 16 bit lookup table that performs the color segmentation. The tracker decomposes the binary (0 or 255 value) color segmented image into a Gaussian pyramid using a 5×5 convolution kernel. The images are not re-binarized after each convolution. The tracker performs the blob coloring and later stages of the algorithm on the host processor and not in the Datacube system. The host processor must transfer the pixel data from one level of the gaussian pyramid into its memory for further processing. Transfers between system memory and Datacube MV200 memory are relatively slow, averaging less than 200 kbytes/sec. The third level of the pyramid was the highest resolution image that the tracker can completely process while maintaining a 30Hz rate. The tracker uses the blob coloring algorithm from Ballard and Brown (1982). After the tracker performs blob coloring at the third level of the Gaussian pyramid it could compute a more accurate localization of the feature centroid by calculating that centroid in the full resolution color segmented image. Processing at full resolution requires the processor to have extended access to the Datacube memory. The 200 kbyte/sec bandwidth to Datacube memory is not high enough to support centroid calculations on the full resolution image at 30 Hz. Due to this limitation the tracker performs the centroid calculation on the decomposed image and this, in turn, limits the resolution of feature localization. The minimum feature that the tracker can detect is in the 8×8 pixel size range.

Multiple sources contribute to noise in the localization of the feature point centroids. One is from the color segmentation lookup table, which generates a sharp cut-off for color detection. The inputs to the color segmentation look-up table are the 8 bit hue and saturation values. The output is a binary (0 or 255) color thresholded image. The tracker creates the look-up table using an instantaneous transition from unacceptable to acceptable values for hue and saturation. If the edges of this transition were sloped instead, it should reduce the noise, and resulting jitter, associated with color thresholding because a small variation in hue or saturation near the acceptance boundary would not result in large changes in the threshold output. The output of the look-up table is the input of the Gaussian pyramid decomposition which can handle an image that is not strictly binary. We did not perform any experiments to test this potential improvement. A second source of centroid localization error is the spatial quantization caused by the tracker performing centroid calculations on level three of the pyramid. The spatial quantization is directly tied to processor cycles and access to the Datacube memory. A faster host processor and an update to the newer Datacube MV250 system that has significantly increased the bandwidth between processor and Datacube memory would allow centroid localization to be done on the full resolution image. A last source of localization error that we noticed during experimentation is

systematic error from the video camera or the Digicolor digitizer. With at least one video camera we used there is a variation in color segmentation that is frame rate based. Viewing the color segmented blobs from the Datacube system we noticed a frame-to-frame alternation between two distinct segmentation patterns. We did not investigate the exact source for this variation but postulate that it is a systematic variation in the camera or the digitizer.

In our implementation the tracker transmits the feature centroid locations from the Datacube host system to a tracker subsystem component that runs on the graphics workstation. This workstation runs the graphics, haptic and video subsystems (Figure 36). The graphics workstation part performs the interframe correspondence of feature points and the filtering and prediction operations discussed in Section 4.3.2. It would logically make sense for these operations to be performed on the Sun 3 computer hosting the Datacube system. With that arrangement the image processing computer could be considered, in an abstract sense, as an affine camera position tracker and the entire tracker subsystem would run on it. We made the decision to separate these logically connected functions based solely on the capabilities of the available hardware and the desire to maintain a 30Hz tracking rate. The image processing computer does not have sufficient power to handle all the operations of the tracker subsystem. Section 4.5.5 describes the software architecture for the parts of the tracker subsystem that run on the graphics workstation.

We measured the system latencies, discussed in Section 4.3.2, by having the graphics subsystem send a time stamp packet to the tracker subsystem. The graphics subsystem generates this time stamp using its high speed hardware clock. The tracker subsystem returns the time stamp in the next packet of feature data transmitted to the graphics subsystem which then computes the elapsed time. The elapsed time computed does include the time to send the time stamp packet to the tracker subsystem. Strictly speaking, the system latency should only be the time from when a video frame arrives until the video subsystem generates the augmented image based on the video frame. Our computation thus provides an upper bound on the system latency. We measured latencies in the 70 to 90 msec. range.

4.5.2 Graphics subsystem

The current graphics subsystem runs on a Silicon Graphics Indigo2 workstation with High Impact graphics and Impact video. The video hardware component is described in Section 4.5.4. The haptics subsystem (Section 4.5.3) runs on the graphics processor also. The graphics capabilities of the system are not being taxed very hard because of the simple nature of the virtual objects being rendered. The high cycle demands of the haptic subsystem required a reduction in the complexity of the images. The graphics workstation ran out of processor cycles when the haptics subsystem was running and the scene contained more than two simple cube or sphere objects. One example of the reduction in complexity was rendering the globe with

larger tessellation (20 sided sphere) than desired. The functionality of the graphics subsystem which also includes the user interface composes the main software component of the system and is described in Section 4.5.5.

4.5.3 Haptic subsystem

The visible part of the haptic subsystem is the Phantom haptic device shown in Figure 19. This device generates force feedback to the finger of the user. It creates a compelling sense of interaction with virtual objects. Not only can the user feel the surface texture of objects but weight of the objects is also simulated. The hardware amplifiers and software drivers for this run on the SGI computer with the graphics subsystem. Sensable Technologies, the manufacturer of the PHANToM™, provides an interface library named GHOST and we coded to that application program interface. GHOST is a C++ class library that allows the programmer to define a haptic scene graph in a manner analogous to the graphics scene graph created by Open Inventor. Unfortunately, the program must define two separate scene structures. A single one can not be shared for both purposes. The haptic subsystem software is an integral part of the graphics subsystem software because of the tight coupling needed between the two. Position information is constantly exchanged between the graphics scene and haptic scene to keep them aligned with each other. Merging the operations for the two subsystems into a single software process was the best approach to reduce the interprocess communication that occurs. The software architecture is described in Section 4.5.5

To maintain the fidelity of the haptic rendering, GHOST starts a servo loop that runs at high priority and requires 1 kHz service rate. If the servo loop does not get called within 2 msec. (500 Hz rate) a failsafe mechanism shuts off the force drivers. The user must restart the augmented reality program because it is not possible to restore correct Phantom operation after a forced shutdown. The SGI machine running our augmented reality system is a single processor system. Running GHOST and the haptic servo on a single processor—along with all the other parts of the system—places severe limitations on the complexity of the haptic scene the system can render. An optimal design is to dedicate one processor in a multi-processor configuration to handle only haptic operations. These limitations are discussed in Section 5.2.2. Another difficulty working with the GHOST library is that in the Silicon Graphics environment it is not stable. With version 2.1 the system locks up or reboots at random times. Sensable Technologies is aware of this problem and will hopefully solve it in a future release of the software.

4.5.4 Video subsystem

The video subsystem is hardware in the Silicon Graphics computer used by the system. Video inputs and outputs are 4:2:2 525 line digital video that follows the CCIR-601/SMPTE 602 video standard. The 4:2:2 nomenclature refers to the sampling rate for the three components, YUV, of the video signal. The luminance, Y, component is sampled at every pixel while the two chrominance components, U and V, are sampled every other pixel. We use external A/D and D/A converters to process standard NTSC analog video through the system. There are two channels of digital video input and two channels of digital video output.

The subsystem also contains internal hardware for keying and blending two video signals. The inputs to the blender and keyer can be one of the input video channels or a graphics window displayed on the workstation console. The programmer can direct output to the digital video output ports or to a workstation screen window. The programmer can setup the keyer to operate as either a luminance or a chroma keyer. The hardware also supports blending between a video input channel and graphics screen window.

We configure the keyer in the luminance key mode. The virtual image rendered in a window on the workstation monitor is the key signal that keys the live video from one of the digital video input ports. Even though there are two video inputs and outputs to the video subsystem as a whole, the keyer can only process one video signal at a time. We set the luminance key value at black. This is the color of the darkest regions in the background of the virtual graphics image. The keyer keys the live video signal at every pixel in the graphics image whose luminance value is at the luminance key value. Any pixel in the graphics image whose luminance value is above the key value replaces the video pixel at that location in the merged output image. With this keying method a virtual pixel that is above the luminance key value always replaces a real video pixel. Note that a pixel in the middle of a virtual object that meets the keying criterion causes the live video to show through at that spot in the virtual object. This fact is exploited by the methods discussed in Sections 4.1.7 and 4.4.4 that are used to get a real object to visually occlude a virtual object. We experimented using the alpha blending in the video hardware to make smoother transitions and reduce the blocked contour when our foreground detection method keys live video to occlude a virtual object. The blending we did achieve was visually less acceptable than the original blocks. (Note that we do successfully use alpha blending, as discussed in Section 4.4.4, to create a virtual image as the key signal. The resultant key signal allows real objects to occlude virtual objects. The blending performed in that case is between virtual objects in the graphics image. The alpha blending that we could not get to work well is in the keyer itself—between the key signal and the background live video being keyed.)

Our augmented reality system operates most of the time with a monitor-based display system (Figure 15). Using either a video or optical see-through display (Azuma 1997) increases the user's perception of being present in the augmented

reality scene. The technique of augmenting reality with affine representations immediately lends itself to operation with a video see-through display (Figure 16). Since neither position information nor camera calibration parameters are needed, we created a video see-through display by simply attaching cameras to a standard virtual reality head-mounted display (HMD).

We built our "augmented reality goggles" by mounting two Panasonic miniature color CCD cameras with 7.5mm lens on a Virtual Research VR4 HMD. The user must manually adjust the fixtures holding the two cameras to correctly orient them and obtain a properly fused stereo view of the real scene. This is the only additional system initialization step needed. Figure 38 shows a block diagram for our system. Our video hardware supports the augmentation of only one video signal with an overlay of virtual objects. Note that the system block diagram shows the signals from both video cameras being digitized and then rescanned for display in the HMD. Our arrangement of the video path in this way ensures that the two signals forming the stereo display of the real scene in the HMD have equivalent processing delays.

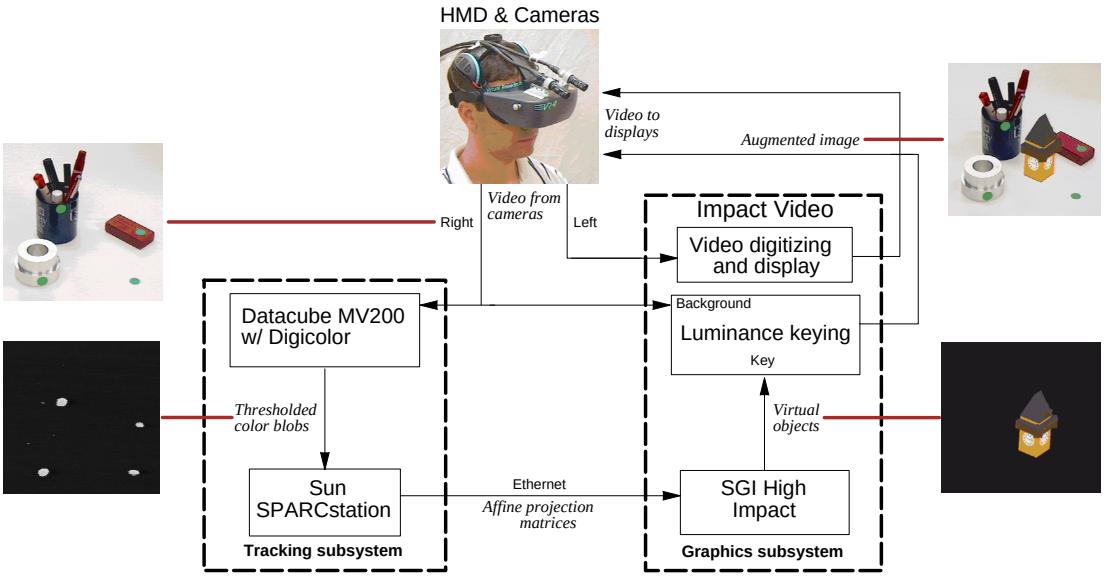


Figure 38 - Block diagram for system incorporating a head-mounted display

As noted above the luminance keyer can only key a single video signal. This means that while the user can see the real scene in the HMD in stereo the video subsystem only creates an augmented image for one eye. We made an interesting observation during informal experiments in the laboratory. Most user's are unaware of the lack of augmentation in one eye when the augmented view is presented in the user's dominant eye. Only when asked to close that eye did the user notice the missing augmentation in the non-dominant eye.

4.5.5 Software Architecture

The software implementation for our augmented reality system is divided into five components: tracker, tracker client, graphic/haptic, haptic servo loop and user interface. Figure 39 is a block diagram of the software architecture. The architecture roughly corresponds to the system component diagram shown Figure 36 although the correspondence between subsystems and software components is not strictly one-to-one. This section describes the high-level design of the software showing the relationships between software components and the flow of control in each of them.

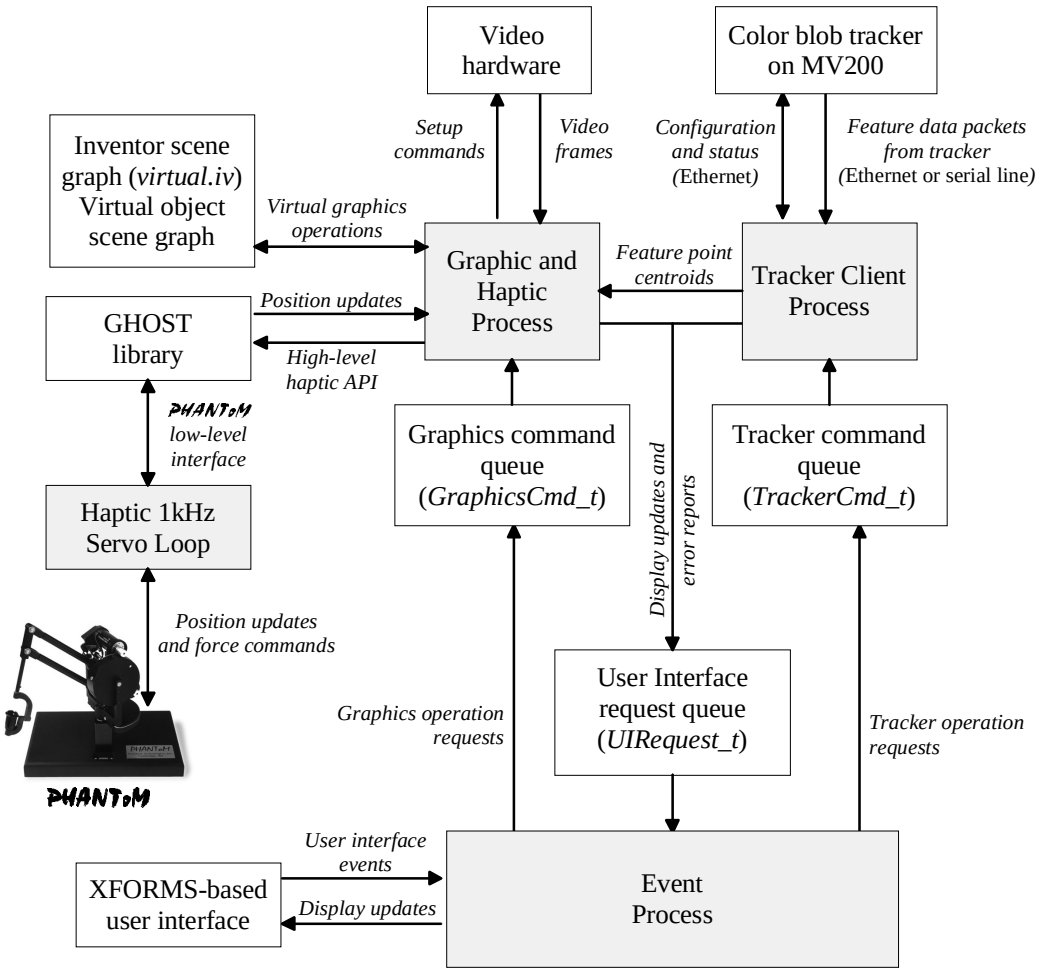


Figure 39 - Software architecture diagram

The gray boxes are the processes which are each spawned using the Silicon Graphics IRIX™ system *sproc* function. This allows all the processes to share a

common address space and facilitates the exchange of information between processes. The command and request queues shown are the main interface between processes. A request is initiated through the queue. Whenever a command or request is placed in a queue SIGCONT is raised on the receiving process. This will wake up the suspended process so that it can process the request. During processing of the request auxiliary data may be exchanged via shared memory. The code is written in C++ but the top level structure of the program is not object-oriented. Object classes were created when deemed appropriate. Otherwise, the program follows a procedural design with a main loop in each process. The flow of control in each of those loops is described in the following sections.

4.5.5.1 User interface (Event process)

The user interface is based on the Xforms design package—a public domain X Windows interface design system written by the Physics Department at the University of Wisconsin - Madison. All the user interaction is through this graphical user interface. The interface defines callback functions for all of the interface button and menu elements. The code associated with this process is mainly contained in the file Event.cc. The user interface event callback functions are defined in virtualUI_cb.cc and virtualUI.h.

The main loop of the Event process is handled by the Xforms library. It loops waiting for user actions in the interface windows. When an event occurs the associated callback function is activated. An idle callback (CheckUIRequests) is specified during initialization of the user interface. During idle periods this function is called. If there is a request in the user interface request queue the request is removed and processed via a large case statement switching on the request type in the UIRequest_t structure. The process is exited when the user presses the exit button.

4.5.5.2 Graphic/Haptic process

The graphic/haptic process is the main process that starts the system running. It is responsible for spawning the other processes and initializing the graphics and video hardware. Once initialization is performed the process enters a loop executing the operations shown in Figure 40.

The tight coupling between graphic and Phantom processing that was discussed in Section 4.4.2 can be seen in the main loop processing. The virtual graphics will be rendered whenever there is a new affine projection matrix computed. Since there is also processing that must be done with the Phantom the system tries to anticipate when the next projection matrix will be available and perform the haptic interaction processing prior to that time. The time between graphic updates is measured and the process sets an alarm to raise SIGALRM and wake itself up prior to that time. This ensures that the haptic update has been performed as near in time to

```
If graphics are running
        Wait up to 50 msec for an X event in the virtual graphics window
            or new video frame to arrive
        If an X event arrived
                Handle the event.  Mainly used to get button clicks on features in
                    the virtual graphic window to define locations in the augmented
                    display
        If a video frame has arrived
                Process the video frame to compute foreground mask if feature
                    is enabled
Else when graphics is not running
        Suspend waiting for signal
If new tracker feature point data has arrived
        Process the data to compute new affine projection matrix
        Handle timing delay data to determine processing latency
        Set alarm for next Phantom update
If Phantom processing is due
        Process haptic scene and update the Phantom
If render operation is needed
        Compute time to next Phantom update
        Perform virtual scene render
If graphic/haptic command or request is in queue
        Process command or request
```

Figure 40 - Graphic/haptic process main loop

the graphic update as possible. It should be done before the graphic render update because the current active haptic position may have an effect on the virtual objects.

The source code files that are associated the graphic/haptic process are:

File name	Description
virtual.cc	Main graphic program. Starts other processes and initializes graphic and video hardware. Cleanup routines and signal handlers
classify_background.cc statistics.cc, statistics2.cc Mask.cc	Compute background color statistics and generate the foreground mask.
GLX.cc	Low level interface to graphic hardware using OpenGL. Needed to augment the capabilities of OpenInventor.
Graphics.cc	Initialize OpenInventor, loading of scene graph and object graph, bounding box operations, computation of object-to-affine transform
HapticScene.cc	Define a blank haptic scene. Set up haptic scene based on haptic id extracted from object scene file.

Phantom.cc	Interfacing to GHOST library, processing Phantom commands, computation of Phantom-to-affine transform
RegError.cc	Perform dynamic registration error tests.
TrackedPoints.cc	Control display of feature centroid and affine basis, load affine projection matrix into scene graph
Align.cc	Compute graphics-to-video transform
Video.cc	Setup, takedown and all interactions with the video hardware

4.5.5.3 Tracker Client process

The tracker client process handles all of the interaction with the color blob tracker. Its main responsibilities are sending configuration and status requests to the tracker, processing the responses to those requests, and receiving the feature point data. Requests made to the tracker client include: start, stop and exit; send options and color segmentation masks; perform alignment operation; initialize the affine basis; and start/stop filtering or set the filter type. Operation of the main loop in this process is detailed in Figure 41. This figure shows the operation when the tracker transmits feature data on the serial channel.

```
If tracker is on
        Wait on data arriving or receiving a signal
        If serial channel data available
                Process incoming feature data
        If socket data is available
                While there is socket data
                        If a receive error
                                Signal tracker death
                        Else
                                Process a reply from the tracker
                                (alignment, rate update, start confirmation)
Else
        Suspend waiting for SIGCONT
If tracker request is in queue
        Execute tracker function for the received command
```

Figure 41 - Tracker client process main loop

In processing the feature point data this process performs the inter-frame tracking operation using a simple nearest neighbor algorithm from the current position to the previous position. There is opportunity here to improve on the inter-frame correspondence. No cross-check is performed from the previous position to the new positions. The tracker client also does the filtering and predictive estimation on the

feature points if those options are active. The values reported to the graphic/haptic process are the filtered values.

The source files associated with the tracker client process are:

File names	Description
TrackerClient.cc	Main loop, functions for processing tracker request, receiving data from tracker and performing inter-frame correspondence
AB.cc, ABC.cc	α–β, α–β–γ filter code
ABFiltering.cc, ABCFiltering.cc	Glue between tracker client and filters
VSDFclass.cc	Variable state dimension filter class
VSDF.c	Glue to Horatio library implementation of VSDF

4.5.5.4 PHANToM™ servo loop

The Phantom servo loop performs the low-level interfacing to the haptic device. This program is provided as part of the Phantom software distribution. It is started during initialization of the GHOST library. This program runs as a high priority process under IRIX. It requests service at a 1 kHz rate and must be serviced at least at a 500 Hz rate or else a failsafe mechanism shuts down the motor drivers and terminates all force generation. As of the writing of this thesis, this process will randomly hang the processor and/or reboot the machine. We were not able to determine the exact cause of this problem. The manufacturers of the Phantom are aware of it and hopefully will resolve it with a future release of their software system.

4.5.5.5 Tracker

The color blob tracker is an ImageFlow program running partly on the Datacube MV200 system and on the host workstation. The operation of this program is discussed in Section 4.5.1. This process is spawned on the remote machine whenever the user requests the system to start the tracker. The tracker client process, running on the local machine, handles all the interactions with the tracker. The two systems communicate via Ethernet sockets using UDP datagrams. A compile time choice allows the tracker to transmit feature point data on a serial line connection. We implemented the serial line connection because our experience showed that there was significant variability in transit times on the house Ethernet. The measured delay using Ethernet transmission, at times, was longer than on the slower serial line connection.

The Datacube program is locked to the video frame rate and is capable of processing color blob data at 30Hz. Every time through the main loop the tracker processes a frame of video and checks the Ethernet socket for a request sent by the

tracker client. Part of processing the video frame is transmitting the packet of new feature point data to the tracker client process over either Ethernet or the serial connection.

The source files that are associated with the color blob detection program are:

File name	Description
Tracker.cc	Main function. Initializes Datacube system and processing commands from the tracker client
TrackerAlign.cc	Perform alignment operation.
TrackerColorMask.cc	Manage the color mask setting for color segmentation.
coltr30.c	Builds multiple pipes and the pipe altering threads (PAT's) that control them.
conv.c	Sets the convolution kernel for the Gaussian pyramid.
blobs.c	Blob coloring algorithm
mylib.c	Miscellaneous functions

5 Conclusions

5.1 Affine Representations: an Attractive Alternative

Augmenting reality using affine representations is an attractive alternative to the previous methods used by augmented reality systems for registering the real and virtual scenes. These previous methods (Feiner, MacIntyre et al. 1993b; Janin, Mizell et al. 1993; Rastogi, Milgram et al. 1995) mostly relied on explicit measurement of the position of the viewer in the workspace. Methods (Grimson, Lozano-Perez et al. 1994; Mellor 1995a; Neumann and Cho 1996) that did not require position sensing did require knowledge about the intrinsic parameters of the video camera viewing the scene or locations of fiducials placed in the image. The simplicity of the affine representations method is very appealing, though the technique comes with a set of new requirements and limitations, namely, the requirement for real-time tracking of features, the weak perspective approximation, and the use of a non-Euclidean reference frame. This section discusses these requirements and limitations along with suggestions for future work to address them.

5.1.1 Tracking as a source of registration errors

Our augmented reality system has two main sources for static and dynamic registration errors. The entire technique rests on the ability to track multiple feature points over time in video images. The accuracy of the localization of the feature points in each video frame directly effects the definition of the common affine reference frame and in turn the placement of the virtual objects in the scene. This is particularly true in our system which is run using the minimum number of feature points. Some immunity to inaccurate feature point localization can be obtained at the expense of tracking more points in the scene. The tracker can influence both the static and dynamic registration. As has been mentioned in previous sections errors in the dynamic registration of virtual objects will give them an appearance of moving around in the scene. Virtual objects lagging behind the rest of the scene motion is a common problem in all augmented reality systems. Latencies, starting with the trackers, and then later added on at any point in the system are a problem. Using 30Hz video input to the trackers

there is an immediate one-frame delay waiting for the video image to process. Executing the tracking algorithm adds an additional delay. In our system the color tracker adds approximately one additional frame delay.

Tracking has been studied in computer vision research for a long time. Many tracking algorithms are available that can be considered in place of the color blob tracking algorithm used in our system. Concentrating on the current system, there are areas that can be investigated for improvements. During experiments in the lab we observed that the localization of a blob centroid varies ±1 pixel even with no motion in the scene. This is caused by jitter in the color segmentation. One area that should be checked to improve color segmentation is the shape of the window used in the segmentation lookup table in the Datacube. Currently, this window has a sharp cutoff which can lead to jitter on the edges of the color field. Reworking this table so that it represents a window with sloped sides may help.

To improve the latency in the tracker subsystem tracking on fields at 60Hz should be tried. The Digicolor module used for color acquisition on the Datacube system unfortunately can not be programmed to do field acquires. Datacube Corporation was never able to provide information on how field rate acquisition could be performed. The assumption has been that it is not possible. An alternative that can be considered is to switch to using the Cognachrome Vision System for color blob detection. The Cognachrome, manufactured by Newton Labs, is capable of tracking color features of sizes similar to that used in our system at a 60Hz rate. This rate can be maintained with at least 5 blobs in the scene. Tracking a larger number of blobs will result in decreased performance. The serial communication channel on the Cognachrome can be interfaced using the serial input mode of the TrackerClient code. An additional benefit of the Cognachrome vision system is its $2500 cost which is a factor of ten lower than the cost of a Datacube system.

5.1.2 The affine approximation as a source of registration errors

The affine approximation is a known limitation in our system. Approximating the projective camera as an affine camera introduces errors. The severity of the errors can be limited by operating within guidelines of not having feature points too far off axis, keeping the focal length short compared to the average distance to the feature points in three-dimensional space and keeping the difference in depth between the nearest and farthest feature points short compared to the focal length of the lens. We also assume a perfect lens is in the video camera. A distortion-free lens is rarely the case with most lens, at the least, exhibiting radial distortion to some degree. Lens distortion incorrectly projects the feature points into the image plane and will produce errors in the registration of virtual objects.

One benefit of the affine camera assumption is the simplicity of the computation of the projection matrix. If the simple affine formulation is kept then the limits of the affine assumption will be present. One direction to pursue is using a

common projective representation instead (Faugeras 1992). A projective representation requires tracking a minimum of five feature points instead of the four required for the affine representation. The affine approximation has been shown to yield more accurate results in structure-from-motion computations than a projective representation (Wiles and Brady 1996) when the large object-to-camera distance requirement is not violated. This result warrants future investigation of Wiles and Brady's *combined appropriateness* idea where the most appropriate camera model is selected for use. Lens distortion can be compensated for by calculating an appropriate image warp (Mohr, Boufams et al. 1993) or estimating the distortion coefficients in conjunction with the tracking subsystem using a recursive variable state dimension filter (VSDF) (McLauchlan and Murray 1996).

5.1.3 Consequences of a non-Euclidean reference frame

The global affine coordinate system defines relative relationships between the camera, real scene and virtual objects. This is useful for the purpose of registering virtual objects in the real scene but does have drawbacks that some researchers consider significant (Azuma 1997; Koller, Klinker et al. 1997). Since we have only relative relationship information generated at runtime there is no absolute reference frame that is known apriori. No metric information about the real 3D world is obtained, which eliminates the possibility of placing virtual objects based on measurements in the real scene. The affine coordinates for locations in three-dimensional space are not known until the common reference frame is defined at runtime. The virtual objects must be placed interactively after the global coordinate system has been determined via methods described in Section 4.1.5**.** An alternative is to place the virtual objects in the real scene using standard measurements and then at runtime calculate the transform of the coordinate system that defines the real scene to the newly defined common affine reference frame. This is the technique employed by the "one-step cheat" used by our current system. This "cheat" assumes that the bounding box for the virtual object is the affine coordinate system and thus the affine coordinates are know without computation via the Affine Reconstruction Property (Section 4.1.3). For this to work, the feature points that define the affine basis points are placed so that they define a cube in three-dimensional real space.

The global affine coordinate system, in general, is not an orthonormal frame in space. Operating in a non-orthonormal coordinate system has ramifications for some computations that must be performed. On the graphics subsystem our work has demonstrated that projection computations, z-buffer determination of visible surface and texture mapping can be performed within this affine representation. Other rendering algorithms, such as lighting computations, that require metric information in the form of measurement of angles, can not be performed in a straightforward manner directly in affine space. For some augmented reality systems, such as those addressing applications in the maintenance and military domains, this limitation on the rendering

of virtual objects is not a major issue. Additionally, how to perform the computation of dynamic behavior for the virtual objects in an affine coordinate system is an open question. The technique used in this system was to compute the dynamics in the Euclidean reference frame of the Phantom and then transform the virtual objects into the global affine coordinate system for rendering. Likewise, real objects that interact with any virtual objects were defined in the global coordinate system and then transformed into the Euclidean Phantom coordinate system to determine the haptic interactions

5.1.4 Feature tracking: the coign of the method

Augmenting reality with affine representations is limited by the accuracy, response and abilities of the tracking system. To date we have simplified our tracking problem by limiting it to tracking high-contrast objects and easily segmented color blobs. For each video image the tracking subsystem must compute a consistent global coordinate system. Tracking in a completely natural setting requires better tracking algorithms or continued engineering of the scene to simplify the problem. We did not want our work to mutate into a thesis on better trackers. Instead the features were engineered to make the tracking process easily tractable. More capable tracking algorithms are available in the literature and can be used within this system.

No results are presented in our work for the situation when the feature tracker detects more or less than the required number of feature points. The current system will not generate an affine projection matrix if the tracker does not detect the correct number of features in an image. The system freezes the image of the virtual object at its last rendering if the correct number of feature points are not available. Additional research work is needed to determine a method to generate a stable affine coordinate system when there are a variable number of feature points from one video frame to the next. This variation may be due to:

- feature points being occluded for one or more frames,
- a current feature point permanently disappearing, or
- a new feature point appearing in the image.

One promising direction to look for a method to handle this feature point variation is the Variable State Dimension Filter (VSDF) (McLauchlan and Murray 1995). The formulation for this recursive Kalman filter allows a variable number of features. It also enforces the affine structure by considering the feature points to be noisy observations of the system state defined to be the affine projection matrix. Future work in this area should also consider the research of Rodrigo Carceroni who has looked at the most efficient methods to use the VSDF (Carceroni, Harman et al. 1998). The results of his work shows that more accurate filtering of data points can be obtained if very approximate camera calibration is available. The image feature

points are pre-multiplied by the inverse of the approximate camera calibration transform. This yields points defined in the 3D camera coordinate system. The filtering is applied to those points rather than the 2D image points that are filtered in our system. Factoring out the camera projection operation, even if only approximated, yields more accurate determination of the camera pose.

5.1.5 Class of applications best suited for affine augmented reality

Applications where there is little apriori knowledge about the scene will benefit most from our method of augmenting reality using affine representations. Using no scene knowledge we define the global coordinate system simply by extracting feature points from the video images. The user interactively places virtual objects with respect to the global coordinate system at runtime. Also, because of the lack of metric information and limitations on the algorithms that can be directly used while rendering the virtual scene ideal applications are those that do not require highly photorealistic virtual objects to augment the real scene. For example, many applications from the military domain would match well with this technique. The scenario given for the Video Surveillance and Monitoring project (Section 2.3.3) is one good application. Geometric information about the battlefield scene will rarely be available. The ability to transfer position information from a reconnaissance aircraft to ground forces through the correspondence of feature points is a very attractive solution.

In applications where apriori knowledge of the structure of the scene is available a technique that uses the scene information to directly compute a Euclidean camera position (Neumann and Cho 1996; State, Hirota et al. 1996) may be more advantageous. The additional metric information eliminates the need to position virtual objects and create the global affine definition of real objects at runtime. In addition, lighting calculations can be performed in a straightforward fashion. One flexibility that is lost, however, is the ability to vary the focal length of the video camera that is viewing the real scene.

5.2 An Exercise in Software Engineering

An augmented reality system is a complex collection of multiple cooperating components. To build such a system requires a careful analysis of the system requirements. We built our augmented reality system using common real-time design techniques. Often these software engineering considerations are ignored as part of a research project. One aspect of our work was to demonstrate that with careful engineering it is possible to build a robust and reliable interactive augmented reality system. It is a system that is modular in design and can have new components added to it without requiring a redesign of major components. We added the haptic

subsystem toward the end of the development cycle and we accomplished initial integration in the relatively short period of two days because of the modular design.

5.2.1 Designing for minimum latency

The system implementation description in Section 4.5 highlighted the result of many of the design choices we made during the building of our augmented reality system. Latency (Section 4.3) has been seen to be a major concern for generating a high-fidelity augmented reality display. We designed the architecture around several standard techniques used for building complex software systems. The techniques used in the architecture of the system that helped maximize its performance include:

- *Multi-threaded architecture*. There are five process threads in this system: feature tracker, tracker client, graphic/haptic application, haptic servo and user interface. This architecture was decided upon to improve response time for the individual operations. Commands and requests are passed between threaded processes through queues. The alternative would have been a single monolithic loop architecture. Because there were several activities of the program that required waiting for data a loop architecture would not give adequate service to all of the system components. This comment is particularly true for the user interface functionality. With a single threaded system processing of tracker data and the updating of the affine projection matrix would be halted whenever user interface activity was taking place. Or on the other side of the coin, the user interface would not be able to respond at all to user actions if other components of the system were using all of the available processor cycles. The multi-threaded design prevents any one component from using all the processor cycles. The individual threads are full weight Unix processes sharing a common address space. We performed an experiment to determine if improved performance could be obtained using lightweight POSIX compliant *pthreads*. We abandoned this direction was abandoned because the current IRIX implementation of *pthreads* is not functioning correctly. When a new release of the OS is made the potential benefits of this change can be reconsidered.

- *Event driven programs.* When the program was originally built, we observed CPU utilization close to 95%. This was in an early stage of the design before all of the graphics or any of the haptic operations were in place. The user interface was very sluggish in response and adding more functionality seemed unlikely. These processes spent most of their time spinning on input. We adopted an event driven architecture to minimize idle spiining. The tracker client, graphic/haptic and user interface were all

changed to trigger on events. One event that all three respond to is a new message in their respective command or request queues. The tracker client also waits on a select for new data from either the serial channel or network socket that connects it to the feature tracker. With this design CPU load normally runs in the mid-20% range with the graphics objects typically used and the haptic system not running. When the haptic servo loop is active the utilization is almost 90%. Not spending resources on waiting processes provides the necessary processor cycles for all of the operations especially the haptic servo loop. Only if the system was running on a multi-processor machine could a non-event driven architecture be considered.

- *Shared-memory data transfers*. All processes running on the Silicon Graphics machine operate in a single address space having been created using the IRIX *sproc* system function. This permits easy sharing of data between the threads. The architecture does use a limited number of global variables preferring instead to exchange information between threads using the command and request queues. This provides a clean interface and, for the most part, a single interface point and method for all operations. It made adding new features an easy task. There was a standard set of coding operations for adding a new command or request. This eliminated most need for new design work to work new features into the multi-threaded architecture.

5.2.2 Incorporating haptic technology

The choices for haptic technology for an augmented or virtual reality system are rather limited. Large scale body-suit exoskeletons can provide gross level force sensations to the user. To handle finer manipulation tasks we choose a smaller scale haptic system. The haptic servo loop is a large processor hog. It executes at a 1 kHz rate and will shut down if it is not serviced at a 500 Hz minimum rate. The GHOST library has the nice feature that it can read a VRML file as input. This could provide the ability to have the graphics and haptic scenes defined from the same source. Unfortunately, the R4000 processor available for our work can not render more than a very simplistic haptic scene. The standard clock tower that we often use as a virtual object can not be haptically rendered without drastic trimming of its VRML file down to a set of no more than a handful of objects. Four or five cubes, spheres or other simple shapes was the limit of the processor. Operation in a multi-processor machine can almost be considered a requirement for using the Phantom. One processor alone would be dedicated to running the haptic servo loop and other haptic functions.

The Phantom provides a very compelling sense of touching virtual objects especially when there is no conflict with the visual cues. In an augmented reality

environment conflicts do occur when proper occlusions between the real Phantom and the virtual objects do not occur. The method we investigated with color background detection and masking of the video has promise. Improvement of this technique should be considered for future work. There are many single block errors in the mask. Single block errors in the middle of the mask or along the edges could be handled by erosion and dilation morphological operations. We did not experiment with applying morphology as part of our work. Alternatively, a deformable snake defined using nurbs—a graphic object with hardware rendering support—could be used to track the detected foreground area. This would take care of both interior holes and jagged contours. Another direction we did not pursue is modeling the Phantom in the graphics system and generating the occlusions from the graphics side instead of at the image plane pixel level. Modeling the Phantom device has the disadvantage of not handling occlusions generated by the user's finger and hand. A combination of both techniques should be investigated.

5.3 A System that Works Surprisingly Well

The overall operation of our augmented reality system is very good—considering the simplicity of the underlying technique. This is despite some very obvious flaws. The two flaws that seem the most glaring are: noisy feature tracking and a very rough method for computing the relationship between the haptic coordinate system and the common affine coordinate system. As we noted in Section 5.1.4, we simplified the low-level tracking problem by selecting features that could be easily detected in the video images. The color feature segmentation is visibly noisy and one to two pixel jitter in the localization of features can be seen when the system is running. Any improvements made in feature tracking will directly help improve registration accuracy. The system is also operating with the minimum number of feature points needed to define the common affine coordinate system. Using more than the minimum number of points would provide some robustness to individual errors in feature point localization. This would also allow the system to use the benefits of the VSDF formulation mentioned in Section 5.1.4.

Computing the transform between haptic and affine coordinate systems is performed by having the user move the end effector of the Phantom to the location of the four feature points defining the affine coordinate system. This is a very rough alignment operation. The active point of the end effector is in the middle of the user's finger. A compensation for this offset is added to the end effector location during the alignment operation but it is not a precise procedure. Despite this the registration of the graphic and haptic scene was never a source of problems for the user of the system during informal testing in the laboratory.

5.3.1 Qualitative comments

The generated augmented reality display is very convincing to the user. This is particularly true when using the HMD setup (Figure 38). A small number of users did experience problems generating a fused image of the real scene and for them the monitor-based system was better. Improvements in HMD technology and changing the particular color cameras we used to newer models should improve that situation. Also, the method for fusing the video image of the real scene was to do a rotation of the two viewing cameras. This will generate a fused image at only a single working distance. Some users had trouble maintaining the stereo image because of changes in depth to the scene. They could not, for example, tilt their head up and view a more distant lab area in stereo without adjusting the cameras. The more correct method to create a fused stereo image is to adjust the cameras to match interocular distance of the users pupils. This would require a bit more complex fixture to be used for mounting the video cameras to the HMD. Current technology for the miniature "lipstick" video cameras does not provide for auto-focus. This will remain a limitation for operating in an environment with widely varying distances in the work envelope.

The extent of realism of the display even with the monitor-based system can be gauged by a common question asked by visitors viewing a lab demonstration of the system. Upon seeing the augmented image and getting a short description of the technique we use they will ask how the system can continue to operate when some of the feature points are being occluded by the virtual objects. The display is convincing enough that they begin to think that the video camera is seeing the augmented scene. This highlights how important occlusion is as a visual cue. Knowing that this is an important cue it is not surprising that the realism of the augmented image decreases significantly when proper visual occlusions between virtual and real objects is not performed. This is particularly true with Phantom operation. Without the small virtual fingertip (Section 4.4.4) to orient the user to the active position of the Phantom end effector in the scene using the haptic subsystem would be more difficult.

The relative simplicity of the haptic scene that can currently be rendered is a limitation to performing real world tasks with our system. Internally the graphics and haptic scenes are separate data structures. Defining the interaction between these objects is a limitation of the system. The GHOST library provides an interface to the Phantom that is above the raw device level. Even with this, each one of the haptic demonstrations that we created required hard coded sections of code to initialize the demonstration and perform the interactions between the graphics image and the Phantom. A higher abstraction level is needed to increase the productivity of the programmer of these systems. This comment is not specific to an augmented reality system. It also applies to virtual reality systems that use this haptic subsystem.

5.3.2 Quantitative registration error studies

Our quantitative work centered around the primary challenge for augmented reality, namely, improving the registration error. Using unfiltered feature point localizations, the virtual objects do not remain visually fixed in the three-dimensional space. Small object jitter is present even when there is no motion in the system. This is due to noise in the color segmentation of the feature trackers. Section 4.5.1 discusses potential sources for this noise. When there is motion the objects appear to be moving from their fixed location in the scene. This motion is a very smooth motion and appears as a lag in that the object is behind its correct position as it moves. Filtering the feature points helps with the object jitter at the slight expense of response time. Usually making an assumption of constant velocity motion is not very effective for the non-linear motions that are present in this system. It was surprising that the assumption did work as well as it did and cut the registration error by a factor of 2. The reduction in error is offset by an increase in jerkiness of the virtual object motion due to sensitivity of the first derivative operation to noise in the input. Depending on the application it may actually be preferred to use unfiltered operation, if the smooth lags can be tolerated better than the more jerky object motion obtained with filtering.

Even with the best filtering and predication that could be applied, registration errors in our system are still easily detected by the viewer. The overall error is composed of both static errors, due to tracking error and violation of the affine approximation, and dynamic errors, due mainly to system latencies. The affine camera approximation places an inherent lower bound on the registration error that can be achieved for a particular position in the workspace. To go beyond this the affine approximation will need to be eliminated. Despite that, the method of augmenting reality using affine representations that we demonstrated is good enough for less critical applications. It will require more work if it is to be applied to any application in which misregistration can cause critical errors. Those applications, which would include most medical applications, can consider using affine representations only after more research has been done to improve the method's overall accuracy.

5.4 The Elusive Augmented Reality Interface

The registration problem in augmented reality still has many areas for continuing research activities. This problem is a difficult one to solve and is likely to remain that way. Our work presented a novel approach for performing the coordinate system registration but it has not solved the problem because of several limitations. Future work is needed to extend this method so it can more closely match the performance goals stated in Section 2.4. Particular shortcomings with respect to these goals and possible directions to find solutions include:

Desired Goal	**Area requiring work**
• operate in a completely natural environment	➔ improved feature tracking
• reduce latencies	➔ better filtering, faster hardware
• handle a variable number of feature points	➔ trackers robust to occlusions
• not be limited by the affine approximation	➔ projective representation

The fact that the system interacts with the real world adds an extra level of difficulty to the augmented reality problem. The number of researchers working in augmented reality is small compared to, for example, the number of researchers working in the closely associated virtual reality area. Considering the crucial registration problem there is no recent quantitative work on errors. The work of Holloway (Holloway 1997) still provides the only careful assessment of registration errors in an augmented reality system. Research activity is slowly moving from the lab into the real world, as seen in, the manufacturing application at Boeing and the image guided surgery at MIT and Brigham and Woman's Hospital. Progress in that direction will continue as current research activities bear fruit. There still is lots of fertile ground for exciting research work in augmented reality.

6 Summary, Contributions and Future Work

The work we discussed in this thesis is in the area of augmented reality, which is a relatively new field within the wider virtual reality research area. We discussed the performance goals for an ideal augmented reality system. These goals place additional demands on an augmented reality system compared to a fully immersive virtual reality system. Our thesis work provides a new solution for the registration problem which is faced by all augmented reality researchers and also shows that within that framework many of the other performance goals can be achieved. This chapter summarizes the demonstrated results and contributions of this work.

6.1 Demonstrated Results

Our work demonstrated that it is possible to create an augmented reality system, composed of multiple subsystems, that operates in real-time. Our system correctly registers the virtual objects with the view of the real three-dimensional world using an affine representation for all objects. This registration is effected without any knowledge of the intrinsic or extrinsic parameters of the video camera that is viewing the scene. In addition, the system does not use any form of explicit position tracking to determine user motion within the workspace. We demonstrated the full integration of a haptic subsystem with the other components in our augmented reality system. This haptic subsystem provides the user with the ability to touch virtual objects and feel forces that accurately represent a natural interaction with the virtual object. Our work also demonstrated that within this framework visual and dynamic interactions between virtual objects and real objects are correctly performed.

6.2 Contributions of this Thesis

6.2.1 Affine representations for augmented reality

The primary challenge for an augmented reality system is to determine the proper rendering and registration of the virtual objects that will be merged with the view of the real scene. This requires computation of the relationships between multiple coordinate systems. Most augmented reality systems use methods that transform these coordinate systems into a common 3D Euclidean frame relying on position sensing, camera calibration and knowing the 3D locations of fiducial markers in the scene. By determining the relationship of the multiple coordinate systems to a common non-Euclidean affine reference frame this system is able to determine the camera pose with neither an explicit position measurement nor calibration of the video camera viewing the scene. The registration problem is reduced to:

- real-time tracking of a set of four or more points that define a global affine coordinate system,
- representing the virtual objects in the common affine coordinate system, and
- computing the projections of virtual points in each video image as linear combinations of the projections of the affine basis points.

6.2.2 Realistic interactions with virtual objects

Using an affine representation to solve the registration problem, our work also showed that an augmented reality system can provide realistic interactions with the virtual objects that augment the user's view of the real scene. These interactions include correct visible occlusion of real objects by virtual ones and virtual objects by real ones. It is not completely seamless, however, in that the dynamic interactions are not computed in the affine coordinate system. Instead the interactions are performed in a Euclidean coordinate system and then transformed into the global affine coordinate system for registration with the virtual graphic objects. Using the Phantom as a force output device, the realistic interactions with the virtual object included the ability to:

- sense touching a virtual object in the correct location in space for its placement in the three-dimensional real scene,
- feel the "weight" of a virtual object when it is lifted,
- feel surface texture on a virtual object, and

- feel the force of collisions of virtual objects with real objects.

6.2.3 Analysis of static and dynamic registration errors

We performed an analysis of the static and dynamic registration errors that result from using the method of affine representations in an augmented reality system. In regions in which the affine approximation was not violated, i.e. large camera-to-object distances, the static reprojection errors were less than 3 pixels in a experiment with camera distances ranging up to approximately 5 meters using manually tracked feature and test points (Figure 26, Figure 27). The errors increased to a maximum of 15 pixels for closer distances as expected from the affine camera approximation. We measured dynamic registration errors for our method. Absolute quantitative measures are highly dependent on the motion used during the experiment. We recorded pixel errors in excess of 30 pixels for some rapid motions. In the one test sequence we used for our experiments, the mean pixel error for the base system is 11.9 pixels. Simple predictive filtering with forward estimates of three frames improves the dynamic performance of the system by a factor of 2 to 3 (Figure 30). We see this improvement for the unfiltered case and when we apply α–β or α–β–γ filtering. This validated our qualitative assessment of the system and the quantitative timing measurements that show a 90 msec. system latency. The improvement in performance does come with a loss in smoothness of virtual object motion, however.

6.2.4 System architecture

We have shown a system architecture demonstrating that an interactive augmented reality system can be built to run in real time. Our system is capable of correctly registering the real and virtual images and giving the user the capability to interact with the virtual objects in a natural manner. Our architecture is a contribution in that it can be used as a framework upon which to build future research activities in augmented reality.

6.3 Future Work

There are many opportunities for continued research activity in augmented reality. Many problems still remain unsolved. These opportunities exist in the topic area in general and also more specifically in continuing work started by this thesis. Some of these future projects potentially include:

- **Trackers** - The tracker that we use can be improved in a number of ways. Work on the color segmentation would improve the jitter in localization of feature points. Switching to a faster tracker, such as the Cognachrome Vision System our department purchased, would improve system latencies. To move into a more natural setting will require the use of trackers that can detect and track natural features. In addition, robustness to feature point occlusions will remove some of the restrictions that currently exist in our system. Handling feature point occlusions will most likely require operation with more than the minimum number of points needed to define the basis. Having more features will improve the noise immunity due to the overdetermined nature of the system.

- **Affine representations** - There are a number of limitations to the use of affine representations. Switching to a projective representation will allow for improved operation in some regions of the workspace. A better approach might be a hierarchy of representations along the lines of Wiles and Brady's (1996) *combined appropriateness.* For some applications, it is worth investigating some recent results (Li, Brady et al. 1996; Carceroni, Harman et al. 1998). These results indicate that better estimation of motion parameters can be obtained when some information, even if only approximate, is available for the intrinsic parameters of the camera and the scene structure.

- **Haptic Interface** - No pun intended, the surface has barely been scratched on the use of a haptic interface in augmented reality. There are many more areas for investigating interactions between virtual objects and real objects. How can this technology be incorporated into the application domains mentioned in Chapter 2? Can the dynamics be performed directly in the common affine coordinate system rather than have to be converted from a Euclidean coordinate system? The foreground detection used to resolve occlusions can be improved. Currently, there are many single block holes in the center of detected regions of motion. The contour of a foreground region is also jagged. Morphological operations would clean up these errors in foreground detection or perhaps the contour can be defined by a deformable snake.

We hope that some of these ideas for future work will spark a succession of researchers to continue the work in interactive augmented reality that was started with this thesis.

Bibliography

Adelstein, B. D., E. R. Johnston and S. R. Ellis (1992), "A Testbed for Characterizing Dynamic Response of Virtual Environment Spatial Sensors," In *Proceedings of 5th Annual Symposium on User Interface Software and Technology*, pages 15-21, 1992.

Ahlers, K., D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker and D. Greer (1994), "An Augmented Vision System for Industrial Applications," Technical Report ECRC-94-39, European Computer Industry Research Center (ECRC) 1994.

Ahlers, K. H., A. Kramer, D. E. Breen, P.-Y. Chevalier, C. Crampton, E. Rose, M. Tuceryan, R. T. Whitacker and D. Greer (1995), "Distributed Augmented Reality for Collaborative Design Applications," Technical Report ECRC-95-05, European Computer Industry Research Center 1995.

Aukstakalnis, S. and D. Blatner (1992), *Silicon Mirage - The Art and Science of Virtual Reality*, Berkeley, CA: Peachpit Press.

Azuma, R. (1993), "Tracking Requirements for Augmented Reality," *Communications of the ACM*, 36(7):50-51.

Azuma, R. (1995), *SIGGRAPH '95 Course Notes: A Survey of Augmented Reality*, Los Angeles: Association for Computing Machinery.

Azuma, R. and G. Bishop (1994), "Improving static and dynamic registration in an optical see-through hmd," In *Proceedings SIGGRAPH '94*, pages 197-204, 1994.

Azuma, R. and G. Bishop (1995), "A frequency-domain analysis of head-motion prediction," In *Proceedings of SIGGRAPH 1995*, pages 401-408, 1995.

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

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

Balcisoy, S. and D. Thalman (1997), "Interaction between Real and Virtual Humans in Augmented Reality," In *Proceedings of Computer Animation '97*, pages 31-38, 5-6 June 1997.

Ballard, D. H. and C. M. Brown (1982), *Computer Vision*, Englewood Cliffs: Prentice-Hall, Inc.

Barrett, E. B., M. H. Brill, N. N. Haag and P. M. Payton (1992), "Invariant Linear Methods in Photogrammetry and Model-Matching," In J. L. Mundy and A. Zisserman, editors, *Geometric Invariance in Computer Vision*, 277-292. Cambridge, MA: The MIT Press.

Betting, F., J. Feldmar, N. Ayache and F. Devernay (1995), "A New Framework for Fusing Stereo Images with Volumetric Medical Images," In *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95*, pages 30-39, 1995.

Boufama, B., D. Weinshall and M. Werman (1994), "Shape from motion algorithms: a comparative analysis of scaled orthography and perspective," In *Proceedings of the European Conference on Computer Vision*, pages 199-204, 1994.

Breen, D. (1995), Augmented Reality for Mechanical Maintenance and Repair. http://www.ecrc.de/research/uiandv/gsp/ECRCToday/mechrep.html (Current as of January 1998).

Brooks, F. P., M. Ouh-Young, J. J. Batter and P. J. Kilpatrick (1990), "Project GROPE - Haptic Displays for Scientific Visualization," *Computer Graphics*, 24(4):177-185.

Brown, C. M. (1994), "Tutorial on Filtering, Restoration, and State Estimation," Technical Report TR534, Department of Computer Science, University of Rochester, September 1994.

Byrson, S. (1992), "Measurement and calibration of static distortion of position data from 3D trackers," In *Proceedings SPIE Vol. 1669: Stereoscopic Displays and Applications III*, pages 244-255, 1992.

Capin, T. K., H. Noser, D. Thalmann, I. S. Pandzic and N. M. Thalmann (1997), "Virtual Human Representation and Communication in VLNet," *IEEE Computer Graphics and Applications*, 17(2):42-53.

Carceroni, R. L., C. Harman, C. K. Eveland and C. M. Brown (1998), "Design and Evaluation of a System for Vision-Based Vehicle Convoying," Technical Report TR 678, Department of Computer Science, University of Rochester, January 1998.

Caudell, T. P. (1994), "Introduction to Augmented Reality," In *SPIE Proceedings Vol. 2351: Telemanipulator and Telepresence Technologies*, pages 272-281, Novmeber 1994.

Cavallaro, R. (1997), "The FoxTrax Hockey Puck Tracking System," *IEEE Computer Graphics and Applications*, 17(2):6-12.

Corby, N. R., Jr. and C. A. Nafis (1994), "Augmented reality telemanipulation system for nuclear reactor inspection," In *Proceedings SPIE Vol. 2351: Telemanipulation and Telepresence Technologies*, pages 360-365, 31 October -1 November 1994.

Dinsmore, M., N. Langrana, G. Burdea and J. Ladeji (1997), "Virtual Reality Training Simulation for Palpation of Subsurface Tumors," In *Proceedings of the IEEE 1997 Virtual Reality Annual International Symposium*, pages 54-60, 1-5 March 1997.

Drascic, D., J. J. Grodski, P. Milgram, K. Ruffo, P. Wong and S. Zhai (1993), "ARGOS: A Display System for Augmenting Reality," In *Proceedings of InterCHI 93 Conference on Human Factors in Computing Systems*, page 521, 1993.

Durlach, N. I. and A. S. Mavor, Eds. (1995), *Virtual Reality: Scientific and Techological Challenges*. Washington, D.C.: National Academy Press.

Ellis, S. R., F. Bréant and B. Menges (1997), "Factors Influencing Operator Interaction with Virtual Objects Viewed via Head-mounted See-through Displays: viewing conditions and rendering latency," In *Proceedings of the IEEE 1997 Virtual Reality Annual International Symposium*, pages 138-145, 1-5 March 1997.

Ettinger, G., E. Grimson, T. Kapur, R. Kikinis, M. Leventon, T. Lozano-Perez, J. P. Mellor, S. Wells and S. White (1998), Image Guided Surgery Home Page. http://www.ai.mit.edu/projects/vision-surgery/surgery_home_page.html (Current as of January 1998).

Fabiani, L., G. Burdea, N. Langrana and D. Gomez (1996), "Human interface using the Rutgers Master II force feedback interface," In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 54-59, March 1996.

Faugeras, O. D. (1992), "What can be seen in three dimensions with an uncalibrated stereo rig?," In *Proceedings of Second Euopean Conference on Computer Vision*, pages 563-578, 1992.

Feiner, S., B. MacIntyre, M. Haupt and E. Solomon (1993a), "Windows on the World: 2D Windows for 3D Augmented Reality," In *Proceedings of ACM Symposium on User Interface Software and Technology*, pages 145-155, 1993a.

Feiner, S., B. MacIntyre and D. Seligmann (1993b), "Knowledge-Based Augmented Reality," *Communications of the ACM*, 36(7):53-62.

Feiner, S., B. MacIntyre and D. Seligmann (1995), KARMA. http://www.cs.columbia.edu/graphics/projects/karma (Current as of January 1998).

Foley, J. D., A. van Dam, S. K. Feiner and J. F. Hughes (1990), *Computer Graphics Principles and Practice*, Reading, MA: Addison-Wesley Publishing Co.

Fraunhofer Institute for Computer Graphics (1997), IGD A4 - Augmented Reality. http://www.ecrc.de/staff/gudrun/ar/london/Lond.html (Current as of January 1998).

Gibbs, S. (1995), GMD Digital Media Lab: Virtual Studio. http://viswiz.gmd.de/DML/vst/vst.html (Current as of January 1998).

Grimson, W. E. L., G. J. Ettinger, S. J. White, P. L. Gleason, T. Lozano-Perez, W. M. W. III and R. Kikinis (1995), "Evaluating and Validating an Automated Registration

System for Enhanced Reality Visualization in Surgery," In *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95*, pages 3-12, 1995.

Grimson, W. E. L., T. Lozano-Perez, W. M. W. III, G. J. Ettinger, S. J. White and R. Kikinis (1994), "An Automated Registration Method for Frameless Stereotaxy, Image Guided Surgery, and Enhanced Reality Visulatization," In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 430-436, 1994.

Hoff, W. A., K. Nguyen and T. Lyon (1996), "Computer Vision-Based Registration Techniques for Augmented Reality," In *Proceedings SPIE Vol. 2904: Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling*, pages 538-548, November 1996.

Holloway, R. L. (1997), "Registration Error Analysis for Augmented Reality," *Presence*, 6(4):413-432.

Janin, A. L., D. W. Mizell and T. P. Caudell (1993), "Calibration of head-mounted displays for augmented reality applications," In *Proceedings IEEE Virtual Reality Annual International Symposium '93*, pages 246-255, 1993.

Jannin, P., A. Bouliou, J. M. Scarabin, C. Barillot and J. Luber (1997), "Visual matching between real and virtual images in image guided neurosurgery," In *Proceedings SPIE Vol. 3031: Medical Imaging 1997: Image Display*, pages 518-526, 23-25 February 1997.

Jebara, T., C. Eyster, J. Weaver, T. Starner and A. Pentland (1997), "Stochasticks: Augmenting the Billiards Experience with Probabilistic Vision and Wearable Computers," Technical Report TR-439, MIT Media Lab, October 1997.

Kalata, P. R. (1984), "The Tracking Index: A Generalized Parameter for α–β and α–β–γ Target Trackers," *IEEE Transactions on Aerospace and Electronic Systems*, AES-20(2):174-182.

Kanade, T. (1996), Z-Key: A New Method for Creating Virtual Reality. http://www.cs.cmu.edu/afs/cs/project/stereo-machine/www/z-key.html (Current as of April 1996).

Kim, W. S., P. S. Schenker, A. K. Bajczy, S. Leake and S. Ollendorf (1993), "An Advanced Operator Interface Design with Preview/Predictive Displays for Ground-Controlled Space Telerobotic Servicing," In *Proceedings of SPIE Vol. 2057: Telemanipulator Technology and Space Telerobotics*, pages 96-107, 1993.

Koenderink, J. J. and A. J. van Doorn (1991), "Affine Structure from Motion," *Journal of the Optical Society of America A*, 8(2):377-385.

Koller, D., G. Klinker, E. Rose, D. Breen, R. Whitaker and M. Tuceryan (1997), "Real-time Vision-Based Camera Tracking for Augmented Reality Applications," In *Proceedings of the Symposium on Virtual Reality Software and Technology*, pages 87-94, September 1997.

Kutulakos, K. N. and J. R. Vallino (1996a), "Affine Object Representations for Calibration-Free Augmented Reality," In *Proceedings of 1996 IEEE Virtual Reality Annual International Symposium*, pages 25-36, 1996a.

Kutulakos, K. N. and J. R. Vallino (1996b), "Non-Euclidean Object Representations for Calibration-free Video Overlay," In *Proceedings Object Representation in Computer Vision II. ECCV '96 International Workshop.*, pages 381-401, April 1996b.

Li, F., M. Brady and C. Wiles (1996), "Fast Computation of the Fundamental Matrix for an Active Stereo Vision System," In *Proceedings of the Fourth European Conference on Computer Vision*, pages 157-166, 1996.

Lorensen, W., H. Cline, C. Nafis, R. Kikinis, D. Altobelli and L. Gleason (1993), "Enhancing Reality in the Operating Room," In *Proceedings of the 1993 IEEE Visualization Conference*, pages 410-415, 1993.

Manhart, P. K., R. J. Malcolm and J. G. Frazee (1993), "'Augeye': A Compact, Solid Schmidt Relay for Helmet Mounted Displays," In *Proceedings 1993 IEEE Virtual Reality Annual International Symposium*, pages 234-245, 1993.

Mark, W. R., S. C. Randolph, M. Finch, J. M. V. Verth and R. M. T. II (1996), "Adding Force Feedback to Graphics Systems: Issues and Solutions," In *Proceedings of the 1996 Symposium on Interactive 3D Graphics*, pages 447-452, 1996.

McLauchlan, P. F. and D. W. Murray (1995), "A unifying framework for structure and motion recovery from image sequences," In *Proceedings of the 5th IEEE International Conference on Computer Vision*, pages 314-320, 1995.

McLauchlan, P. F. and D. W. Murray (1996), “Active Camera Calibration for a Head-Eye Platform Using the Variable State-Dimension Filter,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):15-21.

Mellor, J. P. (1995a), *Enhanced Reality Visualization in a Surgical Environment*, Masters thesis, AI Lab, Massachusetts Institute of Technology.

Mellor, J. P. (1995b), "Realtime Camera Calibration for Enhanced Reality," In *Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95)*, pages 471-475, 1995b.

Metzger, P. J. (1993), "Adding Reality to the Virtual," In *Proceedings of the IEEE 1993 Virtual Reality Annual International Symposium*, pages 7-13, 1993.

Milgram, P. and F. Kishino (1994), “A Taxonomy of Mixed Reality Visual Displays,” *IEICE Transactions on Information Systems*, E77-D(12):1321-1329.

Milgram, P., H. Takemura, A. Utsumi and F. Kishino (1994), "Augmented Reality: A Class of Displays on the Reality-Virutality Continuum," In *Proceedings SPIE Vol. 2351: Telemanipulator and Telepresence Technologies*, pages 282-292, 1994.

Milgram, P., S. Zhai, D. Drascic and J. J. Grodski (1993), "Applications of Augmented Reality for Human-Robot Communications," In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1467-1476, 1993.

Mine, M. R., F. P. Brooks and C. H. Sequin (1997), "Moving objects in space: exploiting proprioception in virtual-environment interaction," In *Proceedings of 24th International Conference on Computer Graphics and Interactive Techniques*, pages 19-26, August 1997.

Mohr, R., B. Boufams and P. Brand (1993), "Accurate projective reconstruction," In J. L. Mundy, A. Zisserman and D. Forsyth, editors, *Applications of Invariance in Computer Vision*, 257-276: Springer-Verlag.

National Association of Broadcasters (1994), "Princeton Electronic Billboard Develops "Virtual Arena" Technology for Television,", National Association of Broadcasters, April 18, 1994 1994.

Neumann, U. and Y. Cho (1996), "A Self-Tracking Augmented Reality System," In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 109-115, July 1996.

Neumann, U. and Y. Cho (1997), Video See-through Self-tracking Augmented Reality (STAR). http://tracker.usc.edu/~ykcho/star.html (Current as of January 1998).

Newman, W. and R. Sproull (1973), *Principles of Interactive Computer Graphics*, New York: McGraw-Hill.

Patel, V. V., M. W. Vannier, J. L. Marsh and L.-J. Lo (1996), "Assessing craniofacial surgical simulation," *IEEE Computer Graphics and Applications*, 16(1):46-54.

Polhemus Corporation (1996), Polhemus Trackers. http://www.together.com /trackres/ourprod.html (Current as of 21 April 1996).

Pyros, G. G. and B. N. Goren (1995), "Desktop Motion Image Processing for Episodic Television," *Advanced Imaging*, 10(7):30-34.

Rastogi, A., P. Milgram and J. J. Grodski (1995), Augmented Telerobotic Control: a visual interface for unstructured environments. http://vered.rose.utoronto.ca/people /anu_dir/papers/atc/atcDND.html (Current as of January 1998).

Roland, J. P., R. L. Holloway and H. Fuchs (1994), "A comparision of optical and video see-through head-mounted displays," In *Proceedings SPIE Vol. 2351: Telemanipulator and Telepresence Technologies*, pages 293-307, 1994.

Rosen, J. M., D. R. Laub, S. D. Pieper, A. M. Mecinski, H. Soltanian, M. A. McKenna, D. Chen, S. L. Delp, J. P. Loan and C. Basdogan (1996), "Virtual Reality and Medicine: From Training Systems to Performing Machines," In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 5-13, 1996.

Shapiro, L., A. Zisserman and M. Brady (1995), “3D Motion Recovery via Affine Epipolar Geometry,” *International Journal of Computer Vision*, 16(2):147-182.

Shashua, A. (1993), "Projective Depth: A Geometric Invariant for 3D Reconstruction from Two Perspective/Orthographic Views and For Visual Recognition," In *Proceedings 1993 IEEE International Conference on Computer Vision*, pages 583-590, 1993.

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

Srinivasan, M. A. and C. Basdogan (1997), “Haptics in virtual environments: taxonomy, research status, and challenges,” *Computers & Graphics*, 21(4):393-404.

State, A., D. T. Chen, C. Tector, A. Brandt, H. Chen, R. Ohbuchi, M. Bajura and H. Fuchs (1994), "Case Study: Observing a Volume Rendered Fetus within a Pregnant Patient," In *Proceedings of the 1994 IEEE Visualization Conference*, pages 364-368, 1994.

State, A., G. Hirota, D. T. Chen, W. F. Garrett and M. A. Livingston (1996), "Superior augmented reality registration by integrating landmark tracking and magnetic tracking," In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 429-438, 1996.

Sutherland, I. E. (1963), "Sketchpad: A Man-Machine Graphical Communication System," In *Proceedings of Spring Joint Computer Conference* 1963.

Taylor, R. H., J. Funda, L. Joskowicz, A. D. Kalvin, S. H. Gomory, A. P. Gueziec and L. M. G. Brown (1996), “An overview of computer integrated surgery at the IBM Thomas J. Watson Research Center,” *IBM Journal of Research and Development*, 40(2):163-183.

Thompson, W. B. and J. L. Mundy (1987), "Three-dimensional model matching from an unconstrained viewpoint," In *Proceedings of the IEEE 1987 Robotics and Automation Conference*, pages 208-220, 1987.

Tomasi, C. and T. Kanade (1992), “Shape and motion from image streams under orthography: A factorization method,” *International Journal of Computer Vision*, 9(2):137-154.

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 Transactions of Robotics and Automation*, RA-3(4):323-344.

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

Uenohara, M. and T. Kanade (1995), "Vision-Based Object Registration for Real-Time Image Overlay," In N. Ayache, editor, *Computer Vision, Virtual Reality and Robotics in Medicine: CVRMed '95*, 14-22. Berlin: Springer-Verlag.

Ullman, S. and R. Basri (1991), "Recognition by Linear Combinations of Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992-1006.

UNC - Chapel Hill (1995), Augmented Reality. http://www.cs.unc.edu /Research/graphics/graphics.html (Current as of January 1998).

UNC - Chapel Hill (1997), UNC Ultrasound Research. http://www.cs.unc.edu/~us/ (Current as of January 1998).

Urban, E. C. (1995), "The Information Warrior," *IEEE Spectrum*, 32(11):66-70.

Ware, C., K. Arthur and K. S. Booth (1993), "Fish Tank Virtual Reality," In *Proceedings of InterCHI 93 Conference on Human Factors in Computing Systems*, pages 37-42, 1993.

Welch, R. B. (1978), *Perceptual Modification: Adapting to Altered Sensory Environments*, New York: Academic Press.

Wiles, C. and M. Brady (1996), "On the Appropriateness of Camera Models," In *Proceedings of the Fourth European Conference on Computer Vision*, pages 228-237, 1996.

Wloka, M. M. and B. G. Anderson (1995), "Resolving Occlusion in Augmented Reality," In *Proceedings 1995 Symposium on Interactive 3D Graphics*, pages 5-12, 1995.

Wren, C. R., A. Azarbayejani, T. Darrell and A. P. Pentland (1997), "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780-785.

Yokokohji, Y., R. L. Hollis and T. Kanade (1996), ""What You can See Is What You can Feel" - Development of a Visual/Haptic Interface to Virtual Environments," In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 46-53, 30 March - 3 April 1996.

Ziegler, R. (1996), "Haptic Displays - How we can feel Virtual Environments," In *Proceedings SPIE Vol. 2949: Imaging Sciences and Display Technologies*, pages 221-232, 7-10 October 1996.

Ziegler, R., C. Brandt, C. Kunstmann, W. Müller and H. Werkhäuser (1997), "Haptic Display of the VR Arthroscopy Training Simulator," In *Proceedings SPIE Vol. 3012: Stereoscopic Displays and Virtual Reality Systems IV*, pages 479-486, 11-14 February 1997.

Zikan, K., W. D. Curtis, H. A. Sowizral and A. L. Janin (1994), "A note on dynamics of human head motions and on predictive filtering of head-set orientations," In *Proceedings of SPIE Vol. 2351: Telemanipulator and Telepresence Technologies*, pages 328-336, 1994.

Zorpette, G. (1994), “An eye-popping summer,” *IEEE Spectrum*, 31(10):19-22.