INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Designing Real-Time Vision Based Augmented Reality Environments for Mobile Applications

by

Peiran Liu, B.E.

A Master's thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master's of Applied Science
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Abstract

Augmented Reality (AR) is a variation of Virtual Reality (VR). It allows the user to see computer generated virtual objects aligned with or superimposed upon objects in the real world through the use of some kind of see-through head-mounted display. Human users of such system can interact with the virtual world and perceive information, such as character descriptions of physical objects and instructions for performing physical tasks, in form of annotations, speech instructions, images and 3D models, from input devices such as screens, microphones, or haptic devices.

This thesis describes our work of building a wireless collaborative augmented reality system, which supports vision-based tracking, video-based 3D graphics, and a keyboard interface over wearable computers to interact with virtual objects. Our goal is to have all the mobile users who work in the same real world space share the predefined virtual space and interact with the virtual world in real time. Multi-player Games in AR domain and Collaborative Conferences are two typical applications.

One of the most critical requirements for augmented reality applications is to recognize and locate real world objects with respect to the person's head position and orientation. In this thesis, we propose a new technique for identifying real world objects. The method utilizes a Binary Square Marker, which can identify a great number of real world objects with markers tagged on them by using computer vision techniques. The marker is a link between the real world and the virtual world. It can be detected by a CCD camera attached to a mobile device. The marker works to determine the 3D position and orientation of the object with the marker tagged on it, as well as its unique ID number. Our technique is based on the observation that given four vertices of a square marker region, the coordinates of the projection of every point on the marker can be computed as a linear combination of the coordinates of the projections of four vertices under the assumption of affine transformation.

Acknowledgments

Thanks are in order for my co-supervisors, research group members, committee, parents, professors, friends, and fellow students of S.I.T.E. Without their support, this thesis would never have been completed.

First of all, I would like to thank Prof. Nicolas D. Georganas, who has been my teacher and mentor through out the project, not only for guiding and encouraging me, but also for going out of his way to ensure the resources for the project and making accommodations to the schedule. Without his push and support this thesis would not have been completed so smoothly.

Thanks are also due to Dr. Pierre Boulanger, then with the National Research Council and currently with the University of Alberta, who has introduced me to AR, suggested the problem and assisted me in the completion of this thesis. Pierre was gracious enough to take the time and explain the related computer vision technologies. His expertise was also helpful in allowing real time video output be implemented in OpenGL.

Special thanks go to Xiaowei Zhong, who is presently a Master's student at the University of Ottawa and also a key member of the augmented reality research group. These research results have been made possible with her help in the design and implementation of the prototype architecture, in solving some problems in implementation, and providing materials and suggestions.

I would like to give my special thanks to my parents for having supported me throughout the project.

I am also grateful to all my friends and fellow students of the MCRLab team for many helpful discussions and for their constant encouragement and support through the course of this work.
The financial support of the National Capital Institute of Telecommunications (NCIT), given to the University of Ottawa under the Distributed Virtual Environments with Training Applications (DIVERTIONS) project is also gratefully acknowledged.

Table of Contents

List of Figures

List of Tables

Abbreviations

AR	Augmented Reality
AV	Augmented Virtuality
CCD	Charged-Coupled Device
CRT	Cathode-Ray Tube
DOF	Degrees of Freedom
DR	Diminished Reality
FOV	Field of View
GUI	Graphic User Interface
HMD	Head Mounted Display
IPD	Inter Pupillary Distance
LCD	Liquid Crystal Display
MFC	Microsoft Foundation Classes
MR	Mixed Reality
OST	Optical See-Through
SDK	Software Development Kit
VGA	Video Graphics Adaptor
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
VST	Video See-Through

Chapter 1

1 Introduction

1.1 Thesis Background & Motivation

From the human's perspective, the real world is composed of the physical materials that people can feel by their own senses. The information people get from their senses is very limited in some circumstances and extra information augmenting the real world could overcome the limitations. For example, when a repairman wants to fix a broken pipeline in the wall, a virtual map of the pipeline overlaid on the real scene of the wall, will save the repairman time to find the broken pipeline and open the wall. Another example: without a tour guide in an art museum, information such as the background information of the artist or the music of his/her time will help the visitors to understand the work of art.

Augmented Reality (AR) is used to describe a system, which enhances the real world by superimposing computer-generated information on top of it. AR is a variation of Virtual Reality (VR). VR technologies immerse a user completely inside a synthetic or virtual environment. When immersed, the user cannot see the real world around him/her. In contrast, AR technologies allow the user to see the real world with computer-generated information or context information of the objects in real space superimposed upon or composed with the real world. Hence, AR is a supplement to reality, rather than a complete replacement of it. Combining augmented information with the real world in 3-D space is useful in that it enhances a user's perception of and interaction with the real world. In addition, the virtual information, such as annotations, speech instructions, images, video, and 3D models, helps the user perform real world tasks.

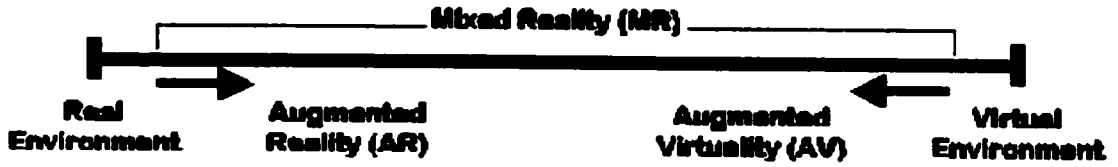


Figure 1.1: Milgram's Reality-Virtuality Continuum
(adapted from [1])

Milgram [2][1] describes a taxonomy that identifies how Augmented Reality (AR) and Virtual Reality (VR) works are related. He defined a Reality-Virtuality continuum as shown in figure 1.1. A real world and an entirely virtual world are at the two ends of this continuum with the middle region called Mixed Reality (MR). Augmented reality lies close to the real world end of the continuum. The predominant perception of AR is the real world being augmented with computer-generated data. Augmented Virtuality (AV) as a term identifies systems in which the principle elements are synthetic with some real world perceptions added, such as texture mapping of video images onto virtual objects. This is a derivative that will fade as the technology improves and the virtual objects in the scene become less and less distinguishable from the real objects. Diminished Reality (DR) is a branch of AR technology. DR technologies require removing real objects from the real world. For example, Billboards or advertisement posters at the sides of busy roadways and highways are annoying and sometimes even dangerous to drivers. One feasible solution is to wear special protective eyeglasses to help us see better by diminishing some reality, which means filtering out dangerously distracting advertisements and replacing them with useful and undisturbed subject matter. This makes the wearer of the special eyeglasses feel undistracted and pay more attention to the road [3].

The research field of Augmented Reality has existed for more than one decade. The progress made in the past few years has been remarkable. Azuma first defined this research field in 1997 in his survey [4]. In addition to introducing the developments of the field, he also gave a detailed description and analysis of many problems related to AR technologies by that time. The field has been growing rapidly in recent years. Since the late 1990's, some conferences and workshops specializing in this area were started. They are the IEEE and ACM International Symposium on

Augmented Reality (ISAR), the International Symposium on Mixed Reality (ISMR), the International Workshop and Symposium on Augmented Reality (IWSAR), and the Designing Augmented Reality Environment Workshop (DARE). Some well-known interdisciplinary consortia were formed that focused on AR. They are the Mixed Reality System Laboratory in Japan [5], Project ARVIKA in Germany [6], the Human Interface Technology Laboratory (HITLab) at the University of Washington [7], MIT Media Lab [8], and Sony Computer Science Laboratories [9].

1.2 Existing Problem

In general, an AR system requires that the registration process between the real and virtual images and the interaction between users and virtual world be **real time** (at least 10Hz) [4]. Two basic technologies are available to accomplish the combining of real and virtual. One is using see-through head-mounted display (HMD) equipment adapted to a wearable computer. The other is overlaying computer-generated graphic images upon the camera-captured video using the monitor.

The advantage of using a HMD is its mobility. When a user wearing a HMD and a wearable computer is moving around in the real world, the pose of the user's head is tracked in real time by the camera of the HMD. The object recognition and the graphic image rendering tasks are accomplished in the wearable computer. The wearable computing technology is most useful in outdoor applications where the use of a computer graphic workstation is not practical. For AR applications, the reasonable image refresh rate must be at least 10 frames per second. According to a research by Durlach [10], delays between head motion and virtual feedback greater than 60ms impair the adaptation and the illusion of presence. **The processing speed and graphic capability** of the existing wearable computer products on the market are very limited.

Accurate dynamic registration is a key issue for AR systems. The registration process can be described as identifying objects in the real world, tracking their location, and aligning virtual objects accurately onto the real object in the scene. Magnetic and ultrasonic trackers used in VR today are constrained by their inaccuracy and limited volume. Therefore, they cannot provide

necessary accuracy and portability for registration in AR. Vision-based AR systems are those using video images captured by the camera to track the camera's position and orientation relative to the objects in the real world. Since the camera is mounted on the user's head in those systems, the computed camera's pose is the pose of the user's head relative to the object in real space. The camera tracking that has the potential to provide the accurate registration data needed by AR systems has been extensively investigated in the field of computer vision. In computer vision technologies, there are several ways for identifying objects in scenes. (1) Edge detection, which is applicable to non-real-time applications due to the complexity of the algorithms; (2) Reconstruction of the 3D coordinates of a number of points in a scene given several images obtained by cameras of known relative positions and orientations, which is applicable to non-mobile applications in restricted environments; (3) Reconstruction of 3D coordinates of objects by detecting fiducial points or landmarks in the image obtained by one camera from a single view, which is applicable for AR systems due to its simplicity and low cost.

Besides working on the essential technologies required to build a compelling AR environment, such as calibration, display, real time tracking, and accurate registration, researchers are solving the problems of how to **interact and control the virtual information** augmented on to the real world, and how to present this information in **an AR display**. These include designing GUI and augmented information databases for AR applications, and solving occlusion problems. The development of AR interfaces for collaborative applications is also a major issue to consider.

Five years ago, more than six classes of potential AR applications had been explored: medical visualization, maintenance and repair, annotation, robot path planning, entertainment, and military aircraft navigation and targeting [4]. **Developing new applications** has been an attractive research topic in recent years. Advances in tracking technologies and displaying devices, and increasingly cheap and powerful computing capability, are making our research easy to go forward. The new applications can be categorized into three areas: outdoor and mobile AR, collaborative AR, and commercial AR [2].

1.3 Thesis Objective & Contribution

The goal of the project is using basic technologies, such as image processing, camera calibration, pattern recognition, error-control coding, geometry invariant, image compression, data communication, multimedia database and computer graphic technologies, to develop an intuitive wearable augmented reality interface for users, as they walk through the real world. This project involved two Master's students, Xiaowei Zhong and myself. In the research, we designed two Augmented Reality prototypes for industrial training applications, which are the focus and emphasis of our research.

The first prototype will permit a "trainee" to see 3D graphic substitutes of microchips overlaid on a circuit board that the trainee has to provide assembly or repair to. This overlaying ("augmented reality") will greatly facilitate those functions. While the trainee is working locally, he/she can also see real-time video, captured at a remote location by a camera, displayed on top of a marker in the real world. This video display will show the trainer's activities, such as installing or removing a chip from the circuit board, which can help the trainee to do the real world task correctly.

The second prototype is for collaborative industrial tele-training, based on distributed augmented reality. Distribution enables users on remote sites to collaborate on the training tasks by sharing the view of the local user equipped with a wearable computer. In this prototype, the users can interactively manipulate virtual objects that substitute for real objects, try out and discuss the training tasks. Imagine a trainer and some trainees are in different locations. A trainee T wearing a wearable computer operates a real circuit board. The other trainees working on remote workstations share the view of the trainee T through the network. A trainer working on a remote workstation also shares the view of trainee T. The trainer manipulates the 3D models of the virtual chips through the keyboard. All of the trainees watch the changed view simultaneously. In this way, the trainees are trained to install chips on a circuit board. All of them are equipped with a microphone and are able to talk through live audio.

Figure 1.2 (a) shows the architecture of the first prototype. Figure 1.2 (b) illustrates the architecture of the second prototype. Xiaowei Zhong implemented most functions in the second prototype. My work is focused on developing the new tracking algorithm and implementing the first prototype. Please refer to figure 1.2 for the workload distribution.

The objective of this work is constructing a new kind of vision based computing environment and developing prototype applications for industrial training, tele-operation, and collaborative design. The hardware platform of the project is a combination of wearable computing, sensing, and networking in a body-integrated design. The software platform combines user interface elements, image processing tools, registration algorithm, windows socket, and multimedia information database on the Window NT operating system.

In this thesis, we propose a Binary Square Marker recognition technique, which adds the error code detection and correction, and marker orientation functions into the 2-D matrix code. This technique makes the code of the new marker being recognized with higher probability. A comparison among known marker techniques those of which are involved in AR applications is made.

Although the AR technology has been developed since 1993, and many related researches have been done in highly-controllable confines of an AR laboratory, only a few of commercial products using AR technology appear in the market today, as for example, the Instructional System for the Boeing Company [11], and the telecommunications services product for Bell Canada field-service technicians [12][13]. In this thesis, we introduce a real-time vision-based AR prototype using the enriched Binary Square Marker. We also propose several potential applications using AR and marker recognition techniques. Those applications could be used in industry, education and entertainment.

Publications resulting from this thesis:

1 P. Liu, N.D. Georganas and P.Boulanger "Designing Real-Time Vision Based Augmented Reality Environments for 3D Collaborative Applications", Proc. Can. Conf. on Elec. and Comp. Eng., Winnipeg, May 2002

2 X.Zhong, P.Liu, N.D.Georganas and P.Boulanger, "Designing a Vision Based Collaborative Augmented Reality Application for Industrial Training", subm. to ACM Multimedia 2002, Juan LePins, France, Dec. 2002

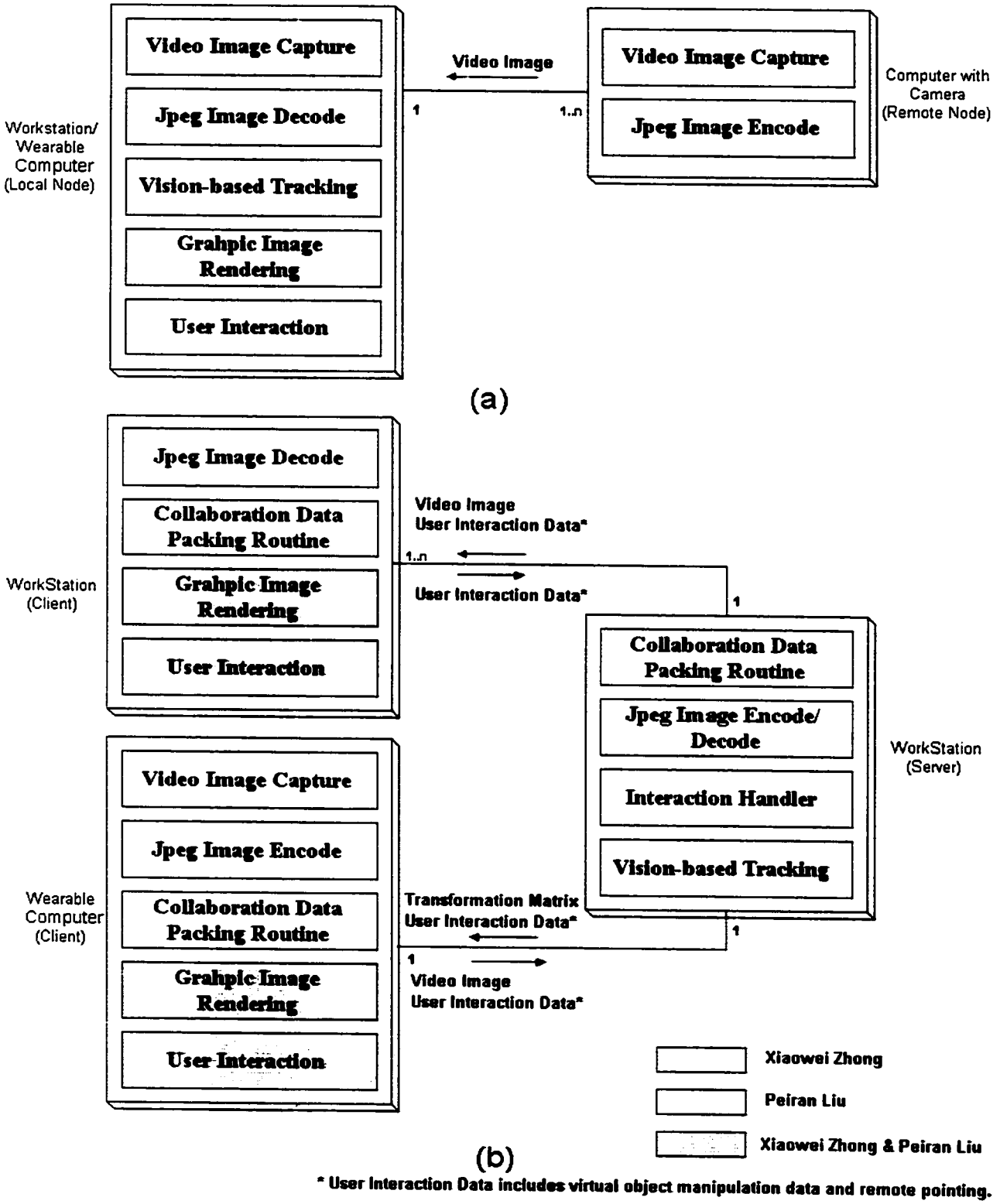


Figure 1.2: (a) The architecture of the first prototype. (b) The architecture of the second prototype.

1.4 Thesis Organization

This thesis describes our work of building a mobile industrial training system and a wireless collaborative system for tele-training, which support video-based 3D graphics and a keyboard interface over wearable computers to interact with virtual objects. Our goal is to build a vision based augmented reality environment in which the mobile users who work in the real world immerse partially into the predefined virtual space and interact with the virtual world in real time.

Chapter 1 introduces the background and motivation of augmented reality technology. Some existing problems of AR technologies are illustrated. This thesis's objective and three main contributions are described.

In chapter 2, some basic concepts and notations about AR technologies and applications are given. Some applicable solutions and their limitations are introduced.

In chapter 3, we propose a vision-based method for object recognition. The vision-based tracking and registration algorithms used in our system are introduced. Then a comparison between our solution and other methods is made. After that, some testing result of the object recognition method we propose is given. Some potential augmented reality applications, such as 3D tracker, collaborative architecture design in a shared space and mobile tour guide system are proposed.

Chapter 4 characterizes how the augmented reality prototype was implemented and discusses the system requirements.

Performance issues are discussed in chapter 5.

The problems we encountered in our research work are illustrated in chapter 6. A conclusion of this work is made and some future works are explained.

Chapter 2

2 Related Work / Literature Review

Before 1998, all of real-time vision-based AR systems implemented registration based on landmark detection techniques, which include image binarizing, 2D color segmentation, boundary detection, watershed, labeling, and template matching techniques. Landmarks could be solid or concentric circular fiducials, ping-pong balls, LED, or the corners of a big rectangle [14][15][16]. Those landmarks might facilitate light-weighed fiducial detection and accurate 3D coordinate reconstruction, but what kind of information, or more precisely, what 3D model, annotation or texture image should be superimposed on the reconstructed object in the real world? In all those systems, the computers know in advance what the object with reference to the landmarks is and which 3D virtual model or texture image will be registered with the object seamlessly or what annotation will be displayed on the screen. This constraint hindered the vision-based AR technology to be used in more practical and larger scale applications, in which the environment around the user changes dynamically and more than one objects in the user's view need to be detected and augmented with objects related information. This raises a question. How to distinguish one real object from another based on the landmark detection techniques? In 1998, Rekimoto proposed a method using a 2-D matrix code marker as landmark in his vision-based system [17]. The markers are 2-D barcodes, which can identify a large number of objects by unique marker IDs. The complexity of detecting the 2-D barcode and extracting bar code ID depends on how the 2-D matrix code marker is defined. The following section takes a look at some of the augmented reality applications that have been developed or are under research.

2.1 Applicable Solutions

2.1.1 Vision Based Augmented Reality with Fiducial Targets

1. Maintenance / repair instruction

The project developed by the Colorado School of Mines [14] tracks and locates a set of pre-placed passive fiducial targets tagged on the real-world objects. The system calculates the pose of the objects and renders graphics overlays to a see-through HMD. Figure 2.1 illustrates a PC maintenance application. The scenario is that a set of fiducial targets are affixed to the cover of a PC and also to the interior frame of the chassis. When the fiducial targets on PC run into the user's view, the system displays directions to remove cover of the PC to the viewer and assist in identifying parts within the PC.

Figure 2.1: (a) Person wearing AR system. (b) View through HMD, showing overlays indicating direction to remove cover of PC. (Courtesy William A. Hoff [14])

2. Visualization

Due to speed limitations in realistic real time rendering, the use of augmented reality technology in visualization applications has been restricted. A Calibration-Free Augmented Reality System [18] was designed and developed by Kutulakos and Vallino. Since the imaging process is represented by affine transformations in the system, the object's projection can be computed without knowing the information about the camera's position or calibration. However, only the properties of virtual objects preserved under affine transformation can be captured. The system generates accurate and fast augmented displays using live video of uncalibrated camcorders from

Figure 2.2: The green fiducials define an affine coordinate frame within which all world points can be represented. (Courtesy Kiriakos N. Kutulakos [18])

one or two views as the only input. At least four fiducial points (Figure 2.2) whose world coordinates are unknown are specified by the user during system initialization. These fiducial points are required to be tracked across frames

2.1.2 Vision Based Augmented Reality with Markers

In designing augmented reality systems, it is often essential to implement a marking (ID) system to make a link between physical and digital spaces. Some examples are barcodes, radio-frequency tags, resonant tags, and infrared LEDs arranged in prescribed patterns. Location information based on GPS can also be regarded as a kind of ID. Vision based AR systems focus on markers that can be attached to objects or environments.

1. CyberCode

The CyberCode [17][19] is a visual tagging system based on a 2D-barcode technology. The CyberCode tags can be recognized by low cost CMOS or CCD cameras. As in other systems using 2D-barcode markers, information being encoded in a two-dimensional pattern can be recognized optically from image data. The position and orientation of a tag in 3D space relative to the camera can be calculated. In the CyberCode system, the digital and physical spaces are combined together. For example, when a CyberCode tag is attached to a document, a user can view or even hear a digital version without referring to a file name or a URL. It is also feasible and simple to assign a unique code to each document automatically just by printing the CyberCode out on the document. When a device with a camera recognizes these tags' IDs, the

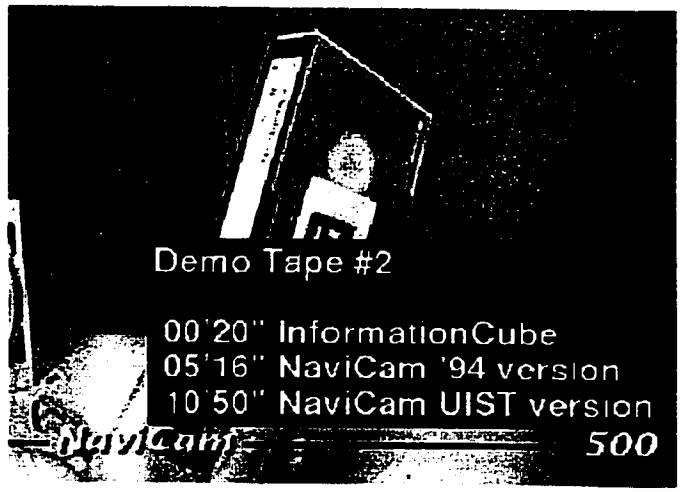


Figure 2.3: Physical embedded links to the digital content. (Courtesy J. Rekimoto [19])

predefined actions, such as opening a specific web page, launching an application, or starting a movie, are activated automatically (Figure 2.3).

2. ARToolkit

A freely available software toolkit (the ARToolkit) for rapidly building AR applications was developed by Billinghurst and Kato [7]. An AR conferencing system implemented using ARToolkit was built. The live video of remote collaborators is represented on the markers of which can be freely positioned around in a user's real space (Figure 2.4). Video images of remote collaborators are mapped on the surface of objects (markers) in the users real environment. The AR conferencing interface relies heavily on computer vision techniques for marker pattern recognition and user head position and pose determination. In this system, the

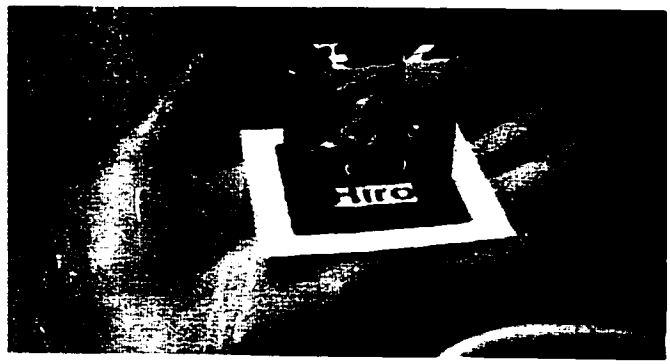


Figure 2.4: Remote user representation in the MR interface[7]

pattern of the marker region is compared with templates, which were given to the system in advance. The pattern template could be user names, characters or an image with limited complexity.

2.2 Basic Methods and Notations

2.2.1 Virtual Reality (VR)

The founder of VPL Research, Jaron Lanier, initially coined the term 'Virtual Reality' in 1989. Other related terms include 'Artificial Reality' (Myron Krueger, 1970s), 'Cyberspace' (William Gibson, 1984), and, more recently, 'Virtual Worlds' and 'Virtual Environments' (1990s).

Although 'Virtual Reality' is used in a great variety of ways, it is a term often confusing and misleading. Originally, the term referred to 'Immersive Virtual Reality' in which the user becomes totally immersed in an artificial and three-dimensional world that is completely generated by computers. Besides, the term 'Virtual Reality' is also used for applications that are not completely immersive. Today, not only the fully immersed VR, but also other variations of VR technologies are becoming more and more important. For example, 3D mouse controlled navigation through a 3D environment with the graphic images displayed on a graphics monitor, stereo viewing of a 3D world from a monitor via stereo glasses worn by the viewer, stereo projection systems, and others. Another example is Apple's QuickTime VR. It uses photographs to model a three-dimensional world and provides faked look-around and walk-through capabilities on a graphics monitor.

2.2.2 Augmented Reality (AR)

Augmented Reality is a derivation of VR technology used to align computer-generated objects or information with the real world in a user's view. AR supplements reality, instead of totally replacing it. The augmented view enriches the user's perception and understanding of the natural environment, often provides information that he/she cannot detect by his/her own senses or gives instructions to do a complex operation. Operating in the real world, AR systems are required to provide real-time interaction with the user and surrounding real environment.

Recently, many publications broaden the definition of Augmented Reality to Mixed Reality and Mediated Reality (Diminished Reality). The essential properties of the original AR systems in the following list [2] are to be shared by those extended technologies.

Blends real and virtual, in a real environment;
Real-time interactive;
Registered in 3D.

This definition of AR does not restrict the user to perceive the virtual world from vision only using particular display technologies, such as Head-Mounted Display (HMD). Actually, AR can potentially apply to all senses, including visual sense, sense of touch, hearing, etc.

2.2.3 Head-mounted display (HMD)

Head-mounted display (HMD) was the first equipment providing its wearer with immersive experiences. As early as in 1965, Evans and Sutherland demonstrated a head-mounted stereo display. A typical HMD has two miniature display screens (LCD or CRT) and an optical system that passes the images from the screens to the eyes and, thereafter, presents a stereo view of a virtual world to the wearer. A head mounted motion tracker continuously measures the position and orientation of the user's head and asks the graphic image generation computer to keep adjusting the scene to the user's current view. As a result, the user can look around and walk through the surrounding virtual world.

To overcome wearers' uncomfortable intrusive feeling of head-mounted displays, alternative concepts (e.g., BOOM and CAVE) [20] for immersed viewing of virtual environments were developed.

2.2.4 See-through HMD

See-through HMD is a device used to combine real and virtual. There are two ways to implement the combination. One is Optical See-Through (OST). Another is Video See-Through (VST). Figure 2.5 is a diagram of an optical see-through HMD. An optical combiner is placed in front of

the user's eyes. Since it is partially transmissible, the viewer can directly look through them to see the real world. The combiner is also partially reflective, which makes it work as a mirror to bounced off graphic images from the head mounted monitor to the viewer's eyes. Video see-through HMDs work by combining a closed-view HMD (do not allow any direct view of the real world) with one or two head mounted video cameras. The video cameras provide the user's view of the real world. Video images come from these cameras are combined with the graphic images created by the scene generator, blending the real and virtual. The result is sent to the monitors in front of the user's eyes in the closed-view HMD. The video composition can be done by some special technology, such as chroma-keying and doing composition with depth information. In VST, the video cameras can also work for vision-based tracking. In OST, additional tracking devices have to be mounted to the helmet for head motion tracking. Table 2.1 lists the advantages and disadvantages of the two approaches.

	Optical	Video
Simplicity	Only one "stream" of video: graphic image.	Deals with separate video streams for the real and virtual image.
Resolution	Shows the graphic image at the resolution of user's display device; User's view of the real world is not degraded.	Limits the real and virtual to the resolution of the display device.
Safety	Safe	If the power is cut off, the user is effectively blind.
Eye offset	No.	Offset between the cameras and the real eyes. The distance between two cameras may not be the same as the user's IPD.
Flexibility in composition strategies	Virtual objects do not completely obscure the real world object.	The compositors can take the real, or the virtual, or some blend between the two and simulate transparency on a pixel-by-pixel basis.
Registration information	The tracker provides the information of the user's head location.	The tracker & the digitized image of the real scene is involved, which can employ addition registration strategies

Table 2.1: Optical see-through vs. Video see-through

Display screen is a key issue related to HMDs that affects AR and VE systems performance and prevents AR technology from reaching end users. Most HMDs use a Liquid Crystal Display

(LCD) to display. However, LCD display devices could not be made with adequate resolution and small enough physical size to suit a HMD very well until very recently. Other HMDs use a Cathode-Ray Tube (CRT) to display, which is equal to a standard television screen in technology. Although such devices can have sufficiently high resolution in a small unit, they are not sufficiently light-weighted to hang on a user's head. Some systems using CRT HMDs have been built and put to limited use already. The standard design of HMDs for both VE and AR has tended to locate optical elements between the display screens and the user's eyes. This design makes the display appear to the user's eyes as if it occupies a larger field of view.

2.2.5 Wearable Computing

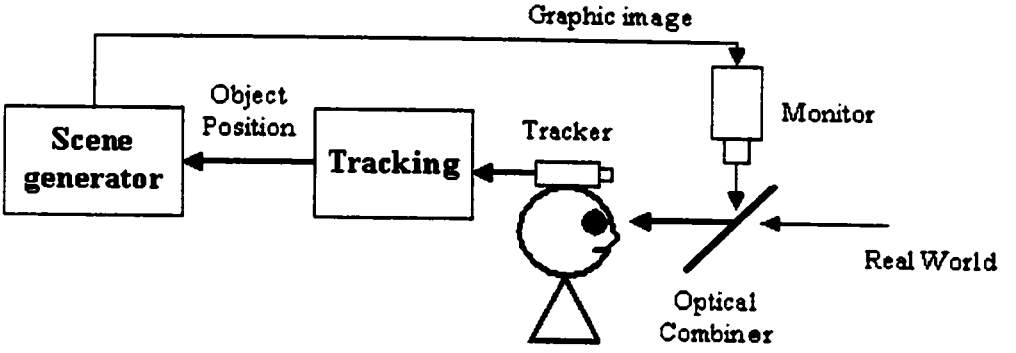


Figure 2.5: Optical see-through HMD conceptual diagram

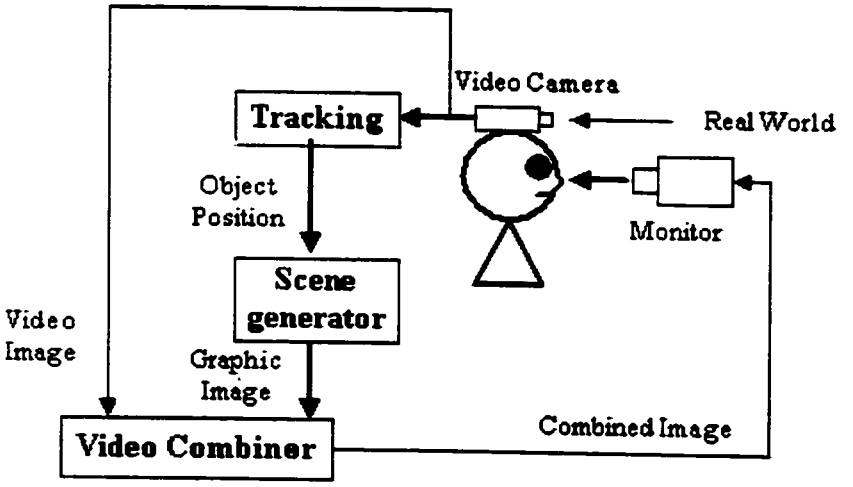


Figure 2.6: Video see-through HMD conceptual diagram

Figure 2.7: Xybernaut MA-IV devices

To date, personal computers have not lived up to their name. Most computers sit on top of the desk and interact with users for only a small part of the day. Smaller and faster lap-top computers have made mobility less of an issue, but the users still couldn't exploit the capability of the computer at anytime anywhere, like the computer is a part of the human body. Wearable computing hopes to break this style of how a computer should be used. A person's computer should be worn, just like eyeglasses or clothing are worn, and interact with the user according to the context of the situation. With HMDs, easy-to-handle input devices, personal wireless LANs, and a host of other context sensing and communication tools, the wearable computer can act as an intelligent assistant, whether it be through a Remembrance Agent, augmented reality, or intellectual collectives [8]. Wearable Computers are used mainly in mobile service applications as of today. Some examples of applications using wearable computing techniques are as follows.

- Assembly and installation inspection
- Troubleshooting and fault isolation
- Maintenance and repair
- Operations and controls
- Geographic information systems
- Training

Wearable PCs, like the Xybernaut MA-IV devices shown in Figure 2.7, are used mostly in industrial settings where hands-free computer use can justify a system's cost significantly.

Maintaining an accurate parts inventory in real time can be done on a wearable PC running Legacy database software [21]. The software is written for standard operating systems Windows or Linux, using a touch-sensitive screen for input. On an assembly line, where using a desktop or laptop PC would be awkward, a wearable PC is used to consult equipment diagrams on a head-mounted display.

Figure 2.8: The IBM Wearable PC prototype packs the computing power of a Thinkpad 560 laptop computer into a package the size of a personal stereo. The configuration includes an IBM's head-mounted display, a video output adapter, a central processing hand-held mouse-pointing device, and a keyboard attached to user's arm.

2.2.6 Field of View (FOV)

For video-see through HMD, the field of view acquired from the cameras defines the field of view used for rendering the synthetic images. There are currently no designs capable of acquiring a wide field view for VST. Most HMDs achieve a field of view that covers only a fraction of the normal human field of view. Typical values for a human's total field of view are 180° to 200° horizontally (150° in each eye with around 120° overlap) and 125° vertically. HMDs for VE systems have a field of view ranging from 20° to 140° horizontally and 20° to 40° vertically. This decrease in the field of view makes the user feel as if s/he is wearing a blinder. This can be very disturbing when the user is moving about in an AR environment.

2.2.7 Camera Calibration

Camera calibration is required to model the real camera and match the virtual camera for accurate registration using pose computation algorithms. It is the process of estimating the intrinsic and extrinsic parameters of a camera, which are used to determine the relationship between what appears on the image plane and where it is located in the 3D world. The intrinsic

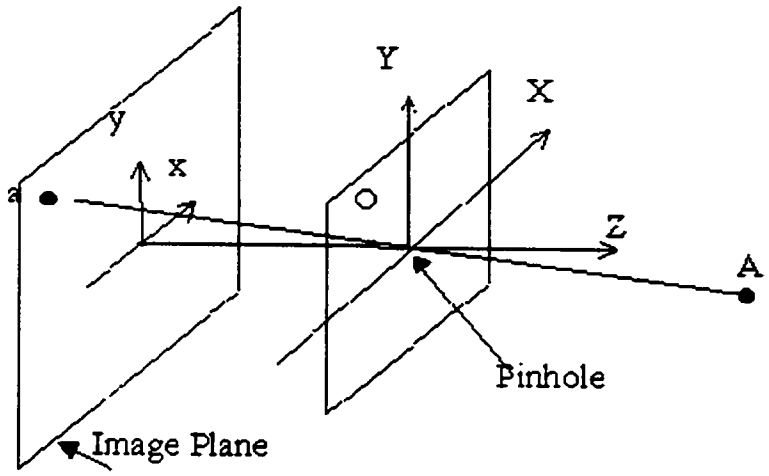


Figure 2.9: The pinhole camera model

parameters are related to the physical characters of a camera, such as focal length, pixel width, and pixel height. The extrinsic parameters are six degree of freedom, three for the orientation and three for the translation of the camera.

Camera calibration methods are based upon solving a mathematical model to create a mapping from the camera projective image plane to the 3D world. The calibration differs depending on what camera model is used for interpreting information from the image plane. The pinhole camera model [22] is used as the basis of most works in computer vision because it provides simple linear equations for the perspective projection of the 3D world onto the image plane. Perspective projection requires a 4 x 4 projection matrix to map from the 3D world to a 2D image plane. The pinhole model was chosen for this work. More complex models are in preference to incorporate lens distortions. The main reason for this choice is because the computational expense of more complex models outweighs the increase in accuracy. The improvement in registration error due to a complex model is negligible due to the relative magnitude of pose computation errors.

A pinhole camera projects a point A from the 3-D projective space onto a point a of the 2-D projective plane. This projection can be written as a 3×4 homogeneous matrix P of rank equal to 3:

$$a \cong PA \tag{1}$$

where $\cong$ is the equality up to a scale factor. If we restrict the 3-D projective space to the Euclidean space, then it is well known that P can be written as:

$$P = (KR, Kt) \tag{2}$$

R and t are the rotation and translation that link the camera frame to the 3-D Euclidean one. The most general form for the matrix of intrinsic parameters K is:

$$\begin{pmatrix} b & rb & u_0 \\ 0 & kb & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

where b is the horizontal scale factor, k is the ratio between the vertical and horizontal scale factors, r is the image skew and u_0 and v_0 are the image coordinates of the center of projection.

2.2.8 Tracking for AR

Accurate positioning and registration of virtual objects in the real environment requires accurate tracking of the user's head and sensing the locations of other objects in the environment. To build effective AR systems, accurate, long-range sensors and trackers that report the locations of the user and the surrounding objects in the environment are required.

Methods used for tracking in AR include magnetic tracking, inertial tracking using accelerometers and gyroscopes, ultrasonic target based tracking, and computer vision based

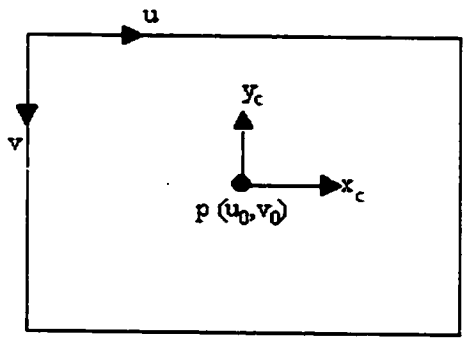


Figure 2.10: Camera Image coordinate system

tracking. It has been shown that electromagnetic trackers, popular in VR applications, are not accurate enough for augmented reality [4]. This is due to the human ability to detect visual disparities while they would not detect differences between haptic and visual senses. Inertial tracking and ultrasonic target tracking have been demonstrated as accurate methods of registration for augmented reality systems in constrained environments and for monitor based AR [4]. Magnetic tracking devices could be used only in those AR applications within certain constrained environments due to limited range of their magnetic field, interference with ferromagnetic objects in the environment, and their lack of portability. Magnetic tracking also requires more initial calibration. Computer vision based registration has developed to overcome problems of low accuracy and noisy tracking in partially constrained environments. Using simple image processing algorithms, fixed targets can be tracked in most applications running in real time. Unfortunately, past computer vision based solutions to the registration problem have generally lacked robustness and been computationally expensive.

2.2.9 Vision-based Tracking

$$\begin{pmatrix} u \\ v \\ h \end{pmatrix} = P_{3\times4} \times C_{4\times4} \times O_{4\times4} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \tag{4}$$

Vision based tracking uses image processing and computer vision techniques to aid registration. Video images of the user's view are available for tracking the viewer's position and orientation. 3D information is recovered from 2D images based on detecting and tracking certain features in images. Since a video-based AR system has a digitized image of the real environments, it may be possible to detect features in the environment and use those to realize registration. A video-based AR system essentially has two cameras, a real one, which generates video of the real environment, and a virtual one, which generates the 3D graphics to be merged with the video images. Both cameras must have the same internal and external parameters in order for the real and virtual objects to be properly aligned (Figure 2.11). To achieve this, an initial calibration of the real camera and a dynamic update of its external parameters are required. In homogeneous coordinates, the accurate projection of a virtual object is described by equation 4, where $[x\ y\ z\ w]^T$ is a point on the virtual object, $[u\ v\ h]^T$ is its projection on the virtual camera image plane,

O_{4x4} and C_{4x4} are the matrices corresponding to the object-to-world and world-to-camera homogeneous transformations, respectively. P_{3x4} is the matrix modeling the object's projection onto the camera image plane.

In some AR applications it is acceptable to place fiducials in the environment. These fiducials may be LEDs or special markers. Ultrasound experiments at UNC Chapel Hill [16] used colored dots as fiducials. The patterns of the fiducials are assumed to be known. Image processing algorithm detects the locations of the fiducials, then those locations are used to make adjustments that enforce accurate registration. We propose a registration system based solely on computer vision that relies on simple algorithms that run robustly in real time.

2.2.10 Registration

In computer vision, the 3-D coordinates of a number of points in a scene, reconstructed by giving several images obtained by cameras of known relative position and orientations, are the result of the registration procedure. In an augmented reality system, registration is the process of properly aligning the objects in the real and virtual worlds with respect to each other, or generating the illusion that the two worlds coexist.

The lack of proper registration or the alignment between the real and virtual objects on the image plane is a major technical problem that so far has limited acceptance of AR applications by customers. There are many components of an AR system that significantly affect the accuracy of registration of the combined image (Figure 2.12). One of the most important components is the tracking subsystem. The input of the subsystem is the tracking data. The output of this subsystem is the transformation matrix from the 3D world coordinate system camera image plane. The output is passed to the graphic image subsystem to generate a view of the virtual objects that matches the real objects in the user's view of the real world. The tracking data and the computed transformation matrix must be accurate so as to make the real and synthetic objects been aligned (spatially registered) properly. In addition, these data must be timely (temporally) generated, so that the virtual models appeared in the user's view would not "swim" back and forth about the real world objects.

Generally, registration in AR is accomplished by having a precise model of the virtual object, knowing the location of the virtual objects in the real world, and knowing the precise pose of the viewpoint, which could be the user's eyes in OST or the camera viewpoint in VST. The rendering engine can then rasterize the object using the viewpoint, on top of the video background image provided by the video camera. This requires an accurate but lengthy calibration procedure, which remains error-prone. Therefore, some recent works have attempted to apply computer vision techniques and projective geometry to solve the problem of aligning synthetic graphic images with their counterpart in the real world.

Calibration is very important for registration. This includes a variety of discontinuous measurements of transformation parameters, such as transformation parameters from one coordinate system to another and camera projection parameters. These measurements are frequently lengthy and trivial, computationally expensive, and inherently difficult to perform precisely. Here are some of the coordinate systems that may be involved in the calibration process.

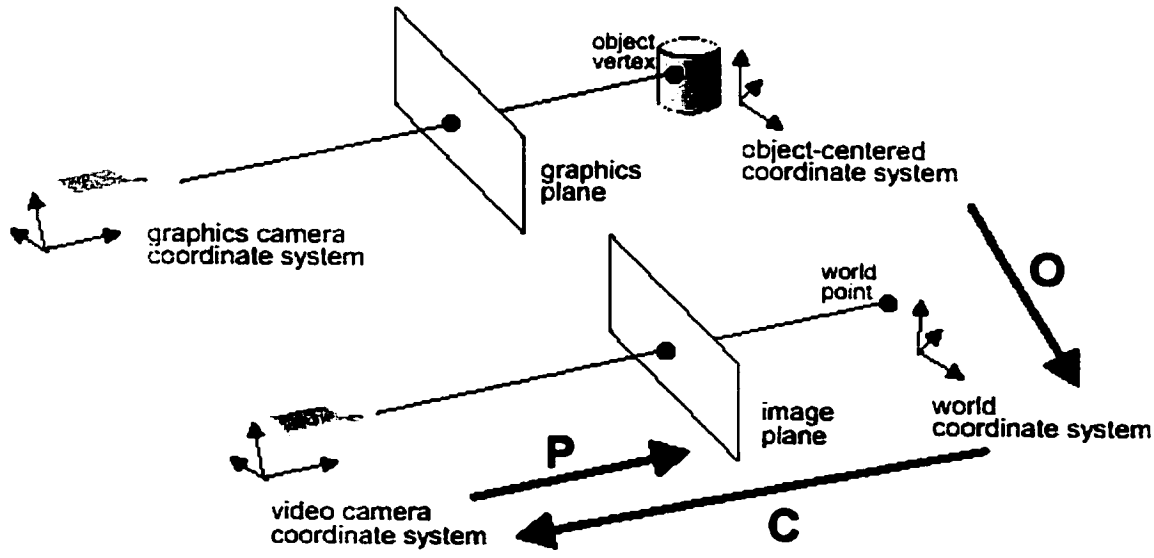


Figure 2.11: Coordinate systems for augmented reality (adapted from [18])

1. The coordinate systems of the tracking subsystem reference and sensor(s)
2. The coordinate systems in which the virtual objects are defined
3. The coordinate systems of the eyes or head mounted cameras

In an OST system, the last coordinate system (3) is an offset between the user's two eyes. In a VST system, the coordinate system (3) is an offset between the two cameras. Also, the parameters that control the projection of the real world objects onto the user's retina (OST) or camera image plane (VST) must be measured and used for generating the graphic images [20].

With the tedious nature by which most of these measurements are acquired and inherent difficulty to perform precisely, calibration is an area that requires further research.

Vision-based tracking systems have been applied to automatic and real time calibration because of the non-intrusive nature of their acquisition, but frequently non-intrusive implies tracking natural features, a difficult and computationally intensive task. Alternate image-based techniques for achieving proper registration don't explicitly compute the viewpoint and view direction with a tracking system. A principle of projective geometry states that for a set of four or more non-coplanar points in 3D, the projection of all of the points in the set can be computed as a linear combination of the projections of just four of the points. This implies that by simply tracking four fiducial points on an object in the video image, the entire object can be registered to the real image. This would avoid having to know the position and orientation of the object relative to some fixed coordinate basis, in which the camera is also tracked. The camera-to-object transformation is effectively computed directly, but not explicitly. It is only determined how it projects into the final (augmented) image.

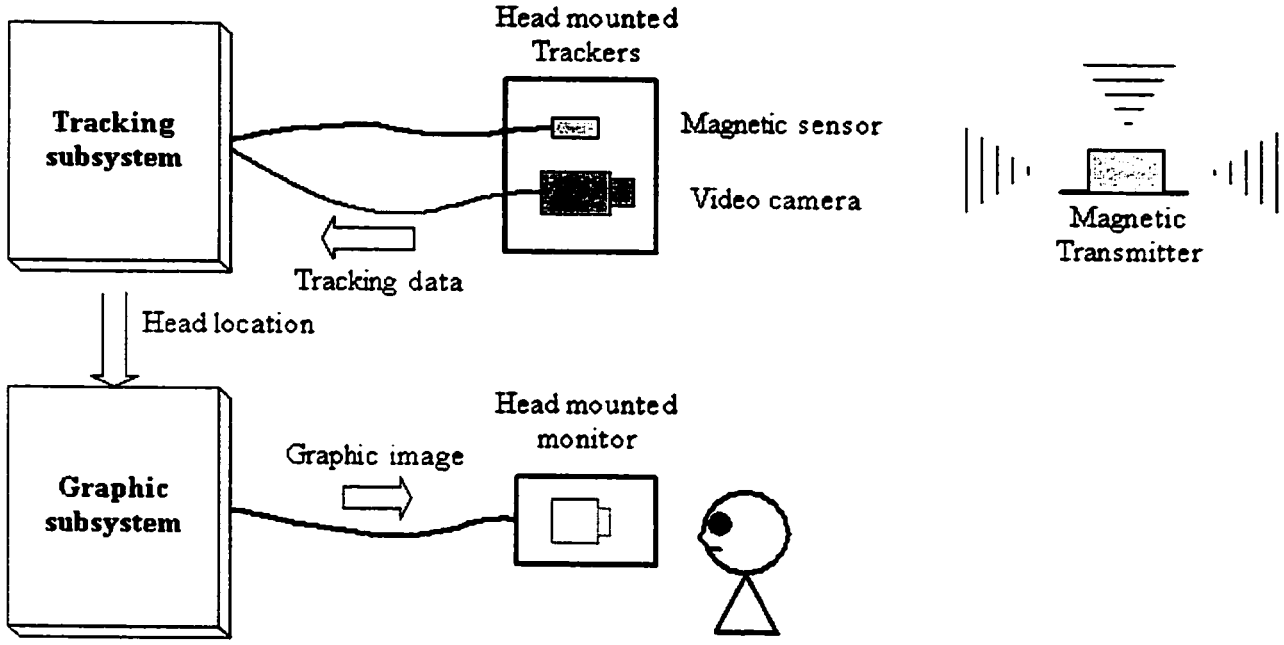


Figure2.12: AR registration by integrating vision based tracking and magnetic tracking

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

2.2.11 Occlusion

Occlusion is a major technical problem that most AR systems must overcome. It can be stated as: Portions of real objects that are behind portions of synthetic objects in the merged world must be obscured by those synthetic objects in the final image. Synthetic objects that are behind real objects in the merged world must similarly be obscured by those real objects [20].

Not all AR applications require occlusion resolution to accomplish their goals. For example, the annotation display in an AR repair and design application wants to block out the real imagery, so that it is clearly visible and legible to the user. In order to correctly represent the occlusion relationships between real and virtual objects, an AR application needs to consider the depth of

the real imagery in the rendering of the graphic images. Graphic image generation for AR is the same as for VE. Of particular interest in this discussion is the depth buffer, or z-buffer. This is a standard way that occlusion relationships are established for objects in VE. The way that AR should differ is that the AR system needs to measure depth for the real world object and compare these data with the computed depth for the virtual object. The AR system can then display exactly the graphic image that is closer to the user's view than the real world object image, and not paint over the real object image that is closer. To implement the occlusion effect, the rendering engine must assign a value in the depth buffer to each pixel corresponding to the real objects in the scene.

The most common method of resolving occlusion in AR is to simply render models of real world objects into the depth buffer. These objects must either be static in the world or tracked in real time. Assuming the model is accurate, the rendering engine can rasterize depth and use these values in the z-buffer computations that are standard in computer graphics architectures. Another approach to resolving occlusion is to compute a mask based on the occluding contours of the real objects. This requires computing the outline of the virtual objects, finding any contours in the real image within this outline, labeling the contour points as behind or in front of the virtual objects, then building the mask from the contour points that are in front.

2.2.12 Block Sum Checking

The theoretical limitations of coding are placed by the results of information theory. These results are frustrating in that they offer little clue as to how the coding should be performed. Error detection coding is designed to permit the detection of errors. Once detected, the receiver may ask for a re-transmission of the erroneous bits, or it may simply inform the recipient that the transmission was corrupted. In a binary channel, error-checking codes are called **parity check** codes.

Practical codes are normally block codes. A block code converts a fixed length of K data bits to a fixed length N codeword, where $N > K$. The rate of the code is the ratio K/N, and the redundancy

of the code is ***1-K/N***. The ability to detect errors depends on the rate. A low rate has a high detection probability, but a high redundancy.

The receiver will assign to the received codeword the pre-assigned codeword that minimizes the Hamming distance between the two words. If we wish to identify any pattern of ***n*** or less errors, the Hamming distance between the pre-assigned codewords must be ***n + 1*** or greater.

A very common code is the **single parity check code**. This code appends to each ***K*** data bits an additional bit whose value is taken to make the ***K + 1*** word even (or odd). Such a choice is said to have even (odd) parity. With even (odd) parity, a single bit error will make the received word odd (even). The pre-assigned code words are always even (odd), and hence are separated by a Hamming distance of 2 or more. The addition of a parity bit has reduced the uncorrected error rate exponentially.

Single parity bits are common in asynchronous, character-oriented transmission. Where synchronous transmission is used, additional parity symbols are added that checks not only the parity of each 8-bits row, but also the parity of each 8-bits column. The column is formed by listing each successive 8-bits words one beneath the other. This type of parity checking is called **block sum checking**, and it can correct any single 2-bits error in the transmitted block of rows and columns. However, there are some combinations of errors that will go undetected.

Parity checking in this way provides good protection against single and multiple bit errors when the probability of the error is independent. However, in many circumstances, errors occur in groups, or bursts. Parity checking of the kind just described then provides little protection. In these circumstances, a polynomial code is used.

2.2.13 Hamming Code Checking

The fundamental principal embraced by Hamming codes is parity. Hamming codes are capable of correcting one error or detecting two errors but not capable of doing both simultaneously. You may choose to use Hamming codes as an error detection mechanism to catch both single and

double bit errors or to correct single bit errors. Hamming codes employ modulo 2 arithmetic. Addition in modulo 2 arithmetic is replaced by the exclusive OR (often written XOR or EOR) logic operation. The number of parity or error check bits required is given by the Hamming rule, and is a function of the number of bits of information transmitted. The Hamming rule is expressed by the following inequality:

$$d + p + 1 <= 2^P \quad (5)$$

Where d is the number of data bits and p is the number of parity bits. The result of appending the computed parity bits to the data bits is called the Hamming code word. The size of the code word c is obviously d+p, and a Hamming code word is described by the ordered set (c,d).

An example of a (11, 7) code is given. Consider a Hamming code to detect and correct for single-bit errors assuming each codeword contains a seven-bit data field, e.g. an ASCII character. Such a coding scheme requires four check bits, since with this scheme the check bits occupy all bit positions that are powers of 2. Such a code is thus known as a (11, 7) block code with a rate of 7/11 and a redundancy of 1-7/11. For example, the bit positions of the value 1001101 are:

Bit Position	11	10	9	8	7	6	5	4	3	2	1
Bit value	1	0	0	x	1	1	0	x	1	x	x

The four bit positions marked with x are used for the check bits, which are derived as follows. The four-bit binary numbers corresponding to those bit positions having a binary 1 are added together using modulo 2 arithmetic and the four check bits are then the four-bit sum:

```
11 = 1011
 7 = 0111
 6 = 0110
 3 = 0011
     1001
```

The transmitted codeword is thus:

Bit Position	11	10	9	8	7	6	5	4	3	2	1
Bit value	1	0	0	1	1	1	0	0	1	0	1

Similarly, at the receiver, the four-bit binary numbers corresponding to those bit positions having a binary 1, including the check bits, are again added together and, if no errors have occurred, the modulo 2 sum should be zero:

11 = 1011
8 = 1000
7 = 0111
6 = 0110
3 = 0011
1 = 0001
0000

Now consider a single-bit error; say bit 11 is corrupted from 1 to 0. The new modulo 2 sum would now be:

8 = 1100
7 = 0111
6 = 0110
3 = 0011
1 = 0001
1011

Firstly, the sum is non-zero, which indicates an error, and secondly the modulo 2 sum, equivalent to decimal 11, indicates that bit 11 is the erroneous bit. The latter would therefore be inverted to obtain the corrected codeword and hence data bits.

2.2.14 CCD Camera

A CCD (charged-coupled device) camera uses a small, rectangular piece of silicon rather than a piece of film to receive incoming light. This is a special piece of silicon called a charge-coupled device (CCD). This silicon wafer is a solid-state electronic component, which has been micro-manufactured and segmented into an array of individual light-sensitive cells called "photosites." Each photosite is one element of the whole picture that is formed, thus it is called a picture element, or "pixel." More common CCDs found in camcorders and other retail devices have a pixel array that is a few hundred photosites high by a few hundred photosites wide (e.g., 500x300, or 320x200), yielding tens of thousands of pixels. Since most CCDs are only about 1/4" or 1/3" square, each of the many thousands of pixels are only about 10 millionths of a meter (about 4 ten-thousandths of an inch) wide!

Chapter 3

3 A Vision-based Method for Object Recognition

3.1 Coding Pattern Studies

In designing augmented reality systems, it is often essential to implement a marking (ID) system to make a link between physical and digital spaces. Some examples are barcodes, radio-frequency tags, resonant tags, and infrared LEDs arranged in prescribed patterns. Location information based on GPS can also be regarded as a kind of ID. Vision based AR systems focus on markers that can be attached to objects or environments. The ID number of each marker is determined by the design of coding patterns inside the marker squares. In this part I will introduce two vision-based methods for object recognition which have been used in many augmented reality applications

3.1.1 Template matching pattern

3.1.1.1 Introduction

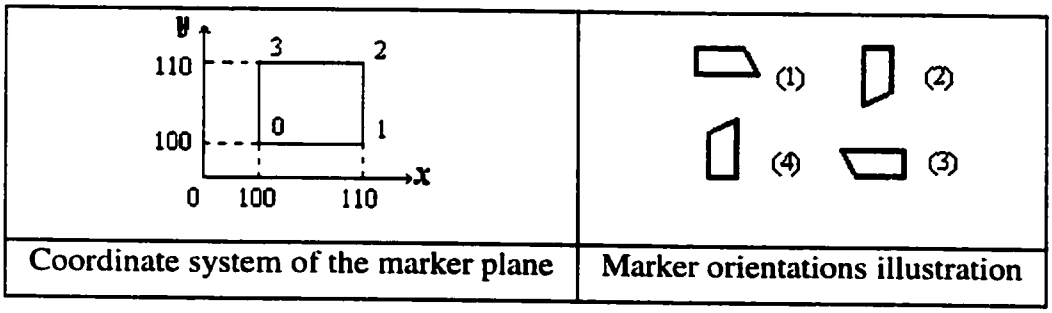
Template matching pattern is a coding pattern designed for vision-based object recognition by H.Kato and M.Billinghurst [23] in 1999. The algorithm is implemented in C language and embedded in the ARToolKit software library [7]. An object in the real world is recognized by matching the pattern inside the square marker tagged on the object to a predefined template. The templates are designed and stored in computers as files. These template files are simply a set of sample images of the desired pattern. Every template has a corresponding ID number. It is a

digital link between the template and the corresponding context information of the object tagged with a marker which has the template pattern inside.

3.1.1.2 Design and Implementation

To create a new template pattern, first print out a black square with an empty white square in the middle. Then create a black and white or color image of the desired pattern that fits in the middle of this square and print it out. Attach the new pattern to the center of the blank square. The best patterns are those that are asymmetric and do not have fine detail on them. Figure 3.1 shows some possible sample patterns. Once the new pattern has been made, a program is run to scan the pattern through the camera captured video image and store the pattern characteristic parameters in a file. These pattern characteristic parameters are retrieved in several steps.

1. Define the coordinates of the four corners of the marker in 3D space with Z-axis value of 0 (x_i, y_i, 0).



Coordinate system of the marker plane	Marker orientations illustration

Detect the coordinates of the four corners of the marker in 2D camera projective image plane (X_i,Y_i).

2. Calculate the 3x3 matrix for transformation

$$\begin{pmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & 1 \end{pmatrix}$$

from marker plane in 3D space to camera image plane in 2D.

Given four pairs of (x_i, y_i, 0) and (X_i, Y_i), transformation matrix parameters can be determined by solving:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -X_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -X_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -X_3x_3 & -X_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -X_4x_4 & -X_4y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1x_1 & -Y_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2x_2 & -Y_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -Y_3x_3 & -Y_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -Y_4x_4 & -Y_4y_4 \end{pmatrix}^{-1} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} \quad (6)$$

3. The marker pattern area in the 3D space is divided to a 16x16 lattice with 16 points (4x4) in each grid. The coordinates of projective points in the image plane corresponding to the points in the marker area in the 3D space are calculated. The color values (BGR) of a grid are determined by the average color values of the 16 pixels in the grid.

4. The BGR values of the 16x16 grids of a template pattern are saved to a template file.

Step1 to step 4 are repeated for 4 times. The BGR values of the marker with different orientations in the camera image plane are calculated and saved at each time.

The algorithm for marker pattern recognition is as follows.

1. Loading template pattern. 4 float-point numbers P1, P2, P3, and P4 are calculated from a template file for 4 different orientations. In equations 7 and 8, the value of i is from 1 to 4 representing 4 orientations of a marker pattern, Cj is the BGR value of the 16x16 grids in the pattern.

$$Pi = \sqrt{\sum_{j=1}^{16\times16\times3} (C_j - \bar{C})^2} \quad (7)$$

$$\bar{C} = \frac{\sum_{j=1}^{16\times16\times3} (255 - C_j)}{16\times16\times3} \quad (8)$$

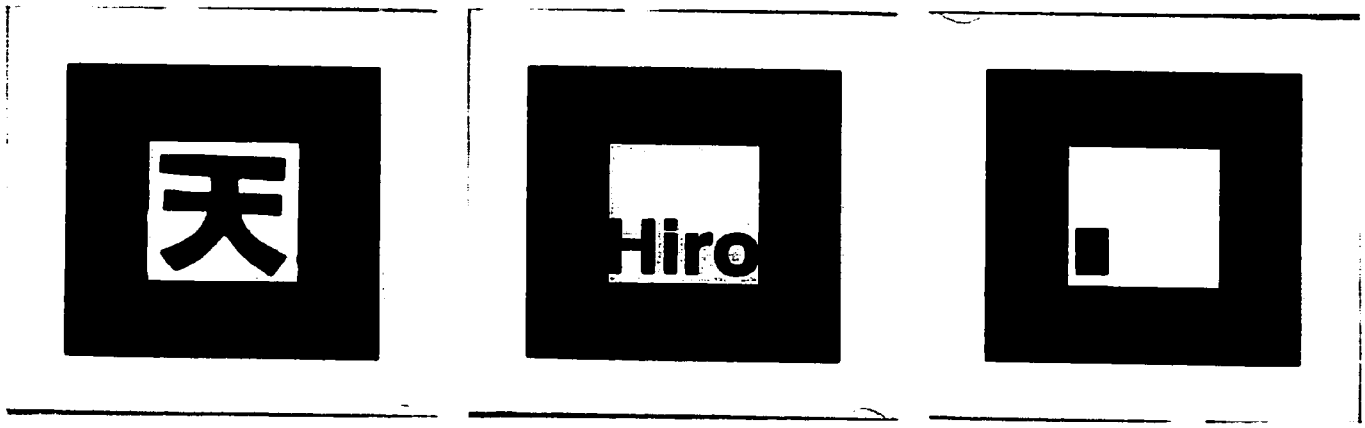


Figure 3.1 Possible Sample Patterns (adapted from [7])

2. Binarizing the image. In this step a simple binarization using the fixed threshold value is used very well because the used temple pattern has a high enough contrast.

3. Selecting the connected regions of pixels the value of which are above the threshold. Searching for the regions in square shape. These regions become candidates for the marker regions. Calculating the $P^{'}$ value of the candidate marker region using the equations listed in step 1. Then comparing it with the loaded P values of all kinds of patterns. The pattern and orientation of the marker captured by the camera are determined by the loaded P value which is approximated by the $P^{'}$ value.

4. Estimation of the transformation matrix.

   Known-size square markers are used as a base of the coordinates frame in which the pattern is represented (Figure 3.2). The transformation matrices from these marker coordinates to the camera coordinates (T_{cm}) are estimated by image analysis.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_x \\ V_{21} & V_{22} & V_{23} & W_y \\ V_{31} & V_{32} & V_{33} & W_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \begin{bmatrix} V_{3\times3} & W_{3\times1} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_{cm} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \qquad (9)$$

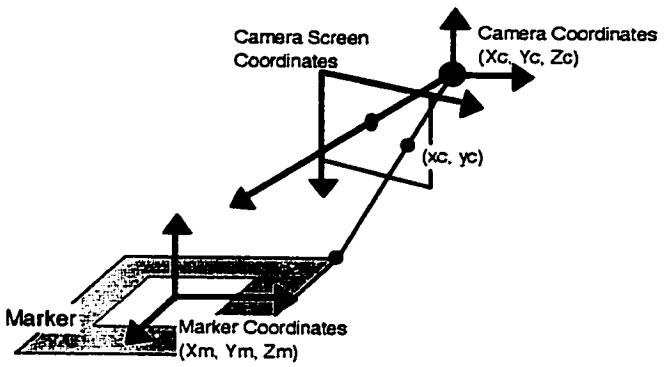


Figure 3.2: The relationship between marker coordinates and the camera coordinates is estimated by image analysis (adapted from [23]).

3.1.1.3 Observation and Evaluation

In order to evaluate accuracy of the marker detection, detected position and pose were recorded while the square marker with 80 [mm] of side length was moved in depth direction with some slants. Figure 3.3 shows the detected error of slant. This result shows that accuracy decreases the further the cards are from the camera.

There are some limitations to purely computer vision based AR systems. Naturally the virtual objects will only appear when the tracking marks are in view. This may limit the size or movement of the virtual objects. It also means that if users cover up part of the pattern with their hands or other objects the virtual object will disappear. There are also range issues. The larger the physical pattern, the further away the pattern can be detected and so the greater the volume the user can be tracked in. This range is also affected by pattern complexity. Patterns with large black and white regions (i.e. low frequency patterns) are the most effective. According to the algorithm introduced in section 3.1.1.2, whether the pattern inside the marker and the marker orientation can be recognized correctly depends highly on the P value. If the P value of the marker in the camera's view matches a P value in the set of loaded P values of all patterns, the marker pattern is recognized. However, it is possible to design two or more patterns which have the same P value. In this case, even though the P values match, the template patterns do not match. This is the weakness inherent inside the algorithm that could result in incorrect recognition. The template patterns are designed at random. They could be simple drawings,

English characters or even Chinese characters. The problem is they do not have a systematic approach to design a template pattern. For this reason, there is no way to study how many template patterns under certain conditions can be recognized correctly. In addition, if the P value of the marker pattern is not calculated precisely under certain lighting conditions, the system does not provide error correction functionality.

Tracking is also affected by the marker orientation relative to the camera. As the markers become more tilted and horizontal, less and less the center patterns are visible and so the recognition becomes more unreliable.

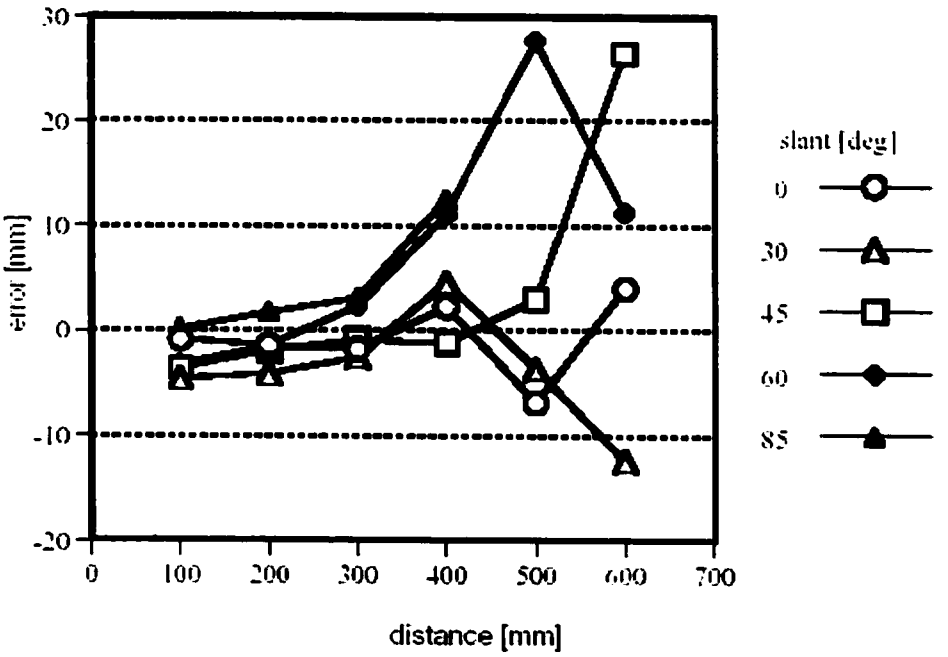


Figure 3.3 Errors of position (adapted from [23]).

3.1.2 2D matrix coding pattern

3.1.2.1 Introduction

The CyberCode is a visual tagging system based on a 2D matrix coding technology and provides some features not provided by other tagging systems [19].

As in other systems using 2D-barcode tags, information is encoded in a two-dimensional pattern, and can be optically recognized from image data. The CyberCode markers can be recognized by low-cost CMOS or CCD cameras. To enable the marker IDs to be read by inexpensive low-resolution CCD cameras, CyberCode encodes fewer bits (24 or 48 bits), than other 2D-barcode systems do. A CCD camera (either monochrome or color) that has 100,000 or fewer pixels can therefore be used as a marker reader. Such a camera works well under normal ambient lighting, and with normal optics. The marker shape is designed to compensate for distortion of the visual image. Therefore the markers are recognized, even if they are not perfectly placed in front of the camera.

The CyberCode markers can also be used to determine the 3D position of the tagged objects as well as their ID numbers. Using a camera as a marker reader, the 3D position and orientation of a marker relative to the camera can be estimated, consequently the camera position and orientation relative to the marker can be determined. This kind of information can be obtained by measuring, feature points of the marker on the camera image plane. In addition, the CyberCode markers can be easily printed by a normal printer, and can be attached to almost any physical objects. These features often make it a good starting point for rapid prototyping.

The simplest use of a CyberCode marker in augmented reality is as an embedded link of real world objects to digital information. For example, when a camera-equipped mobile device recognizes a unique ID of a CyberCode marker attached to a document, a predefined action – such as opening a specific web page or starting a movie is activated automatically.

3.1.2.2 Design and Implementation

The system seeks out the CyberCode marker on incoming video images. It then identifies their code value and estimates the camera position and orientation in relation to the attached position of the marker. The entire image processing is performed by the software. The CyberCode marker is designed in a square shaped region besides a guide bar. The 4 corners in black color are used for detecting the marker region. The guide bar is used to determine the orientation of the square shaped region. The marker ID is recognized in the following four steps (Figure 3.4). The

recognized code frame is used for estimating the position and orientation of the camera. From four known points (four corners of the CyberCode marker) on the image plane, it is possible to calculate a matrix representing the translation and rotation of the camera in a real-world coordinate system [19].

1. Binarizing the image. In this step an adaptive thresholding method is used. However a simple binarization using the fixed threshold value also works quite well because the used matrix code has a high enough contrast.
2. Selecting the connected regions of binary-1 (black) pixels that have a specific second order moment. These regions become candidate guide bars for the marker.
3. Searching for the four corners of the marker region using positions and orientations of guide bars found in step 2. When the guide bar and the four corners are found, the system decodes the bitmap pattern in the marker. The transformation parameters are calculated based on the positions of the 4 corners of the code pattern area. Let $(x_i, y_i, 0)$ be a point on the plane of the matrix code and (X_i, Y_i) be a corresponding point on the image plane of the camera. There is a relation between the two:

$$X_i = \frac{a_1 x_i + a_2 y_i + a_3}{a_7 x_i + a_8 y_i + 1} \tag{10}$$

$$Y_i = \frac{a_4 x_i + a_5 y_i + a_6}{a_7 x_i + a_8 y_i + 1} \tag{11}$$

where $a_1,\ldots,a_8$ are parameters to be solved, representing the camera's intrinsic and extrinsic (motion and rotation) parameters. Giving four pairs of $(x_i, y_i, 0)$ and (X_i, Y_i), these parameters can be determined by solving:

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -X_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -X_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -X_3x_3 & -X_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -X_4x_4 & -X_4y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1x_1 & -Y_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2x_2 & -Y_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -Y_3x_3 & -Y_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -Y_4x_4 & -Y_4y_4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix} \tag{12}$$

This set of parameters restores the effect of rotation, motion and perspective transformation of the camera and maps a distorted code image on the camera plane to the normalized matrix code space.

4. Decoding and error check.

Using the obtained parameters, the system projects the corresponding image region to the code rectangle space. The number of black and white pixels that are projected within the area of each cell determines each cell bit. Finally, the CRC-error checking is applied on the decoded bits, and the certificated cell value is regarded as a recognized code ID. After checking for the error bits, the system determines whether or not the image contains a correct CyberCode marker.

3.1.2.3 Observation and Evaluation

Based on the above algorithm, a workstation class computer (such as the SGI-O2) using the CyberCode object recognition algorithm can recognize the code in real time, a update rate making feasible several of the 3D composition applications. The processing rate is about 15 frames/sec on the SONY VAIO-C1 with a 200MHz Mobile Pentium MMX, and about 5 frames/sec on the Mitsubishi AMITY pen computer with a 50MHz Intel 486.

The 2D matrix coding pattern is based on 1D barcode technology. The marker pattern not only provides the marker identification information, but also provides the marker pose information. The marker ID reader is a normal camera originally designed for taking still pictures or making movies. This is an advantage for a tagging system using mobile devices. However, although the CRC error checking is applied in the CyberCode Marker, there is no error correction functionality. Once an error bit is detected, the system will discard the current image and process the next captured image. When error bits are detected in a sequence of images, those images will be discarded. The jitter generated by discontinued video stream will impair the user's perception. The guide bar besides the code pattern area is used to determine the orientation of the marker. The guide bar recognition increases the complexity of the algorithm. In addition, the guide bar enlarges the marker side as a whole, which limited the size and shape of the object the marker could tag on. According to the algorithm, the number of black or white pixels in each

cell determines the value of the cell. In order to retrieve the value of every cell, the coordinates of each projected point in the marker region are calculated. In the following section, we will introduce the proposed Binary Square marker object recognition algorithm, in which the coordinates of a limited number of feature points in the projected image plane are calculated. This makes the new algorithm simpler than the CyberCode.

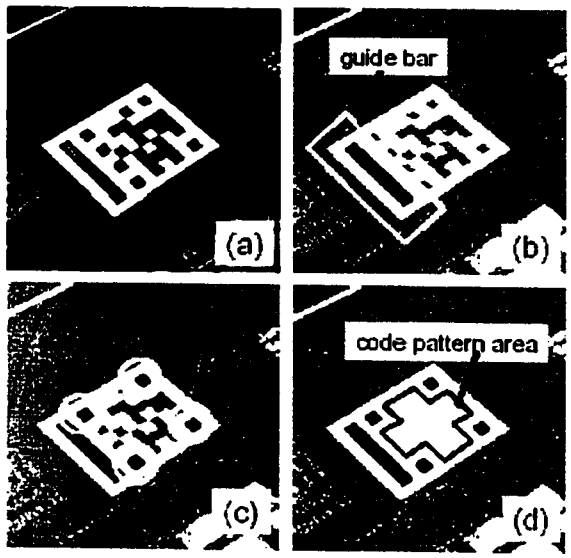


Figure 3.4: The visual tag recognition steps (adapted from [19]).
(a) Video image (b) Guide bar detection
(c) Marker corners detection (d) Marker pattern recognition

3.2 Proposed Method For Object Recognition

Vision-based tracking approaches are based on detecting and tracking certain features in images. These features could be lines, corners, or circle points, which can be easily and unambiguously detected in the images and can be associated with objects in the 3-D world. Our approach uses the corners of the Binary Square Markers attached to moving objects for tracking. Although image processing algorithms can be used to detect the four corners of the square marker, how to determine the square shape region detected in the 2D video image is the specific marker we want, rather than something else which has the square shaped silhouette seen from a certain view of the user occasionally? Furthermore, when more than one objects tagged with a marker appear in the user's view, how to distinguish one object from the others, how to recognize the identification of the object by the detected marker are problems to be solved. Even though the system can extract the exact object and its pose relative to the camera using the object

recognition and pose estimation algorithms, what information about the object the system should be presented to aid the user to do work in the real world, from where this information can be taken, and how to present it to the user? To answer all the questions and build a robust vision based augmented realty system, we proposed a new vision based object recognition algorithm called "Binary Square Marker Recognition Algorithm".

Our AR interface relies heavily on vision-based marker ID recognition and user head pose estimation. The overall system workflow is illustrated in Figure 3.5, which consists of four parts: (1) image processing, (2) pattern recognition, (3) camera pose estimation, and (4) augmenting virtual environment and context information into reality. The first three parts belong to the tracking subsystem. The fourth part belongs to the graphic subsystem. The algorithms are introduced in this chapter. Please refer to *Chapter 4 System architecture* for software design and implementation issues. Some testing results are given in section 3.4.

3.2.1 Image Preprocessing

(1) Binarizing the image.

The program uses a predefined threshold to binarize the video image. Binary images contain only the important information, and often can be processed very fast.

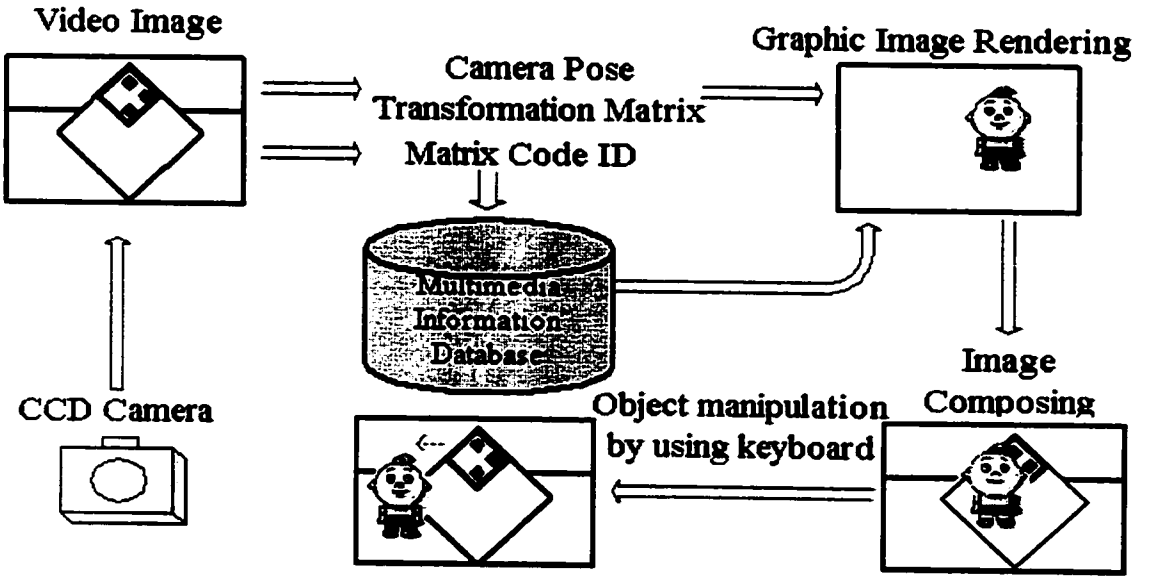


Figure 3.5: Augmented reality system workflow
(tracking subsystem and graphic subsystem)

(2) Selecting the quadrilateral regions. These regions become candidates for the square marker.

(3) Searching and recording the coordinates of four corners of the quadrilateral regions in 2D image plane found in step (2) for pose estimation.

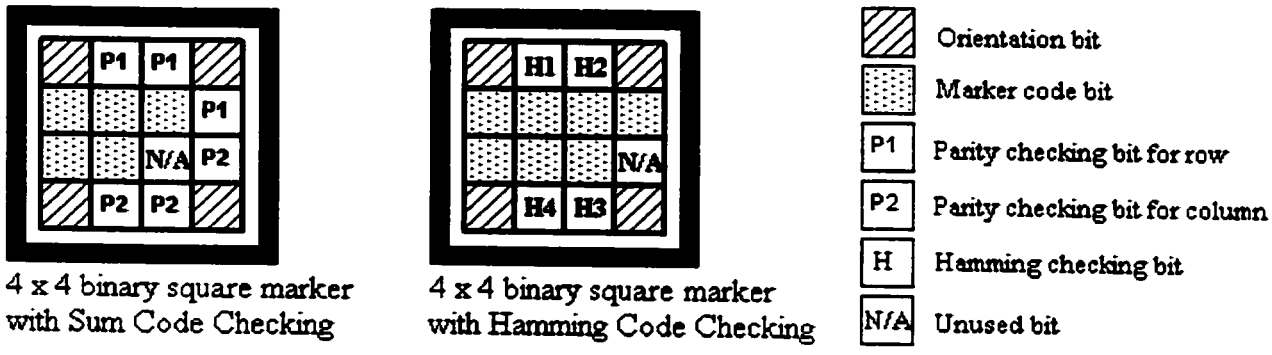
3.2.2 Pattern Recognition

(1) The system recognizes the 2-D binary code pattern inside the square marker region.

(2) Extracting the binary code.

In the system, the marker pattern is defined as a 4x4 matrix (Figure 3.6 (a) and (b)). For a system in which the resolution of the video camera is very high, 640x480 pixels for example, the pattern can be defined as a 5x5 matrix with more complexity, in order to make the system not only recognize more patterns, but also recognize them as precisely as the system with lower camera resolution. Every grid in the matrix represents one bit of the binary code of the marker. The whole pattern is in black and white color. The grid in black represents 1, and the grid in white represents 0. For example, in Figure 3.9 (a) the binary code of the marker, from top left corner to bottom right corner, is 1000 0010 0100 0010. In the pattern, 4 bits determine the orientation of a marker, some bits are used for error checking, the rest bits are to determine the marker ID. Only one corner of the four corners in the pattern is in black color, which makes the pattern asymmetric. The corner in black color is used to determine the location of the most significant bit of the 16bit marker code.

Our algorithm is based on the following observation: Given the four vertices of the marker region, the projection of every point on the marker can be computed as a linear combination of



4 x 4 binary square marker with Sum Code Checking

(a)

4 x 4 binary square marker with Hamming Code Checking

(b)

Figure 3.6. Binary square marker

the projections of the four vertices. The reference frame is defined as a non-Euclidean, affine frame. Although Euclidean geometry describes the three-dimensional real world very well, it is not the only type of geometry. When we consider the imaging process of a camera, the Euclidean geometry is no longer sufficient. Since projective geometry allows a larger class of transformations, which includes perspective projections besides translation and rotation, it models the imaging process of camera very well. Another geometry -- affine geometry, is in between. The allowed transformations of the three different geometries and the measures that remain invariant under those transformations are different. The projective transformations preserve type (point is point, line remains line), incidence (a point lies on a line), and a measure known as cross ratio. The affine transformations preserve type, incidence, parallelism, cross ratio, and division ratio. Besides all the measures preserved by the above two geometries, the Euclidean transformations also preserve length, angle and ratio of lengths.

In the design of our AR prototype, we used one camera for vision-based tracking. Although the projective geometry models the imaging process of a camera from 3D space to 2D image very well, the implementation of the algorithm is with high complexity, which makes it not work very well in real-time augmented reality applications. In order to simplify the algorithm, we model the imaging process as an affine transformation, assuming that the reference frame is defined as an affine frame. We designed the algorithm to compute the coordinates of points in every grid area of the marker in the camera image plane by using the *division ratio* property of the affine transformation invariant.

For the purposes of computer graphics, the division ratio in terms of affine and ideal points defined by I. Herman is as follows [24].

Let A, B and C be three different collinear affine points in the projective plane, denoted by IPE^2, or in the projective space, denoted by IPE^3. The division ratio of the points A, B, and C is denoted by (ABC), which is defined to be the following (non-zero) real number:

$$(ABC) = \frac{\overline{AC}}{\overline{CB}} \tag{13}$$

where $\overline{AC}$ *and* $\overline{CB}$ *mean the directed Euclidean distances of the two points (that is* $\overline{AC} = -\overline{CA}$*).*

If the point C tends to infinity, the limit of the corresponding division ratio (with the points A and B remaining fixed) will be −1; consequently, it seems to be feasible to extend equation 13 to the case when the point C is an ideal point, namely let

$$(ABC) = -1 \tag{14}$$

in this case.

According to the theorem in [24], the division ratio is affine invariant, which means

$$(ABC) = (T(A)T(B)T(C)) \tag{15}$$

If A, B and C are three different arbitrarily chosen collinear points of the plane (or space) and T is an arbitrary affine transformation, (ABC) is the division ratio of the collinear three points A, B and C.

Clearly, since the relative location of every grid in the pattern is predefined, it is easy to calculate the central point coordinates of any grid in the projective plane with reference to the projections of four vertices of the marker. Consequently, the color or value of each grid in the pattern is determined by the color of the corresponding pixels in the binarized image. The following example shows how to compute the central point coordinates of the grid K (Figure 3.7) in the 2D projective plane by using the division ratio invariant.

$$X_K = (X_C - X_A) \times 21/90 + X_A \tag{16}$$

$$Y_K = (Y_C - Y_A) \times 21/90 + Y_A \tag{17}$$

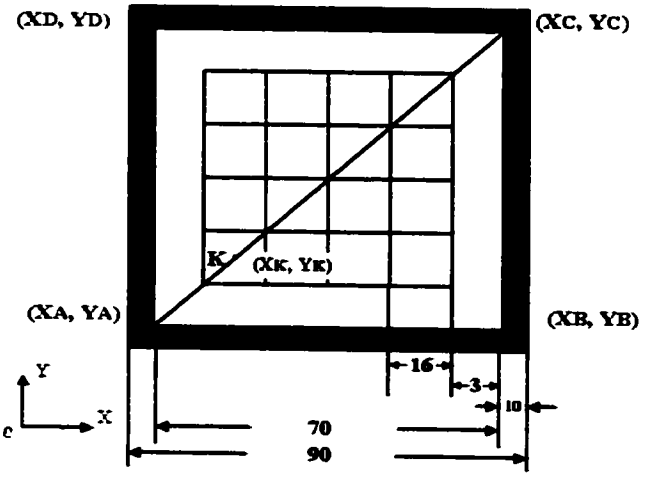


Figure 3.7: An Example shows how to compute the coordinates of a point K in a predefined marker pattern

(3) Error detection and correction.

Block sum checking

The pattern of a 4x4 2D marker represents a 16bit binary code. 5 bits are used to represent the marker ID. 4 bits determine the orientation of the marker in the camera's dynamic view. 6 bits are for the block sum checking, in which 3 bits are odd parity bits for 3 rows and 3 bits are even parity bits for 3 columns. In our marker pattern, the data block is of 3 rows and 3 columns. The block sum checking can correct any single bit error in the bit block. The error bit is detected by checking the 6-bit parity code.

Original matrix is

0	1	0	0
0	0	1	0
0	1	0	0
	0	1	0

sin *gle bit error matrix is*

0	1	0	0
0	0	1	0
1#	1	0	1*
	0	1	1#

* *represents error bit.*

\# *represents changed parity bit.*

If one bit error is detected, the system can either drop the current image and do the image processing on the next video image or correct the error bit. The following example shows how to correct a single bit error of the 4x4 marker illustrated in Figure 3.9(a). In the 16bits, 5bits (00101) represent the marker ID and 4bits (1000) are for marker orientation. The data block is a 3x3 matrix.

In the rest 7 bits, 3 bits (000) make an odd parity code. 3 bits (010) make an even parity code. If one error bit occurs in the bit block, 2 bits of the 6 parity bits will change. The intersection of the row with changed parity bit in it and the column with changed parity bit in it is the error bit in the matrix.

For a 4 x 4 coding pattern, such a coding scheme requires seven check bits. Thus, it is known as a (16, 9) block code with a rate of 9/16 and a redundancy of 1-9/16. For a 5 x 5 coding pattern, this coding scheme requires nine check bits. Thus, it is known as a (25, 16) block code with a rate of 16/25 and a redundancy of 1-16/25.

Hamming code checking

The pattern of a 4x4 2D marker represents a 16bit binary code. 7 bits are used to represent the marker ID. 4 bits determine the orientation of the marker in the camera's dynamic view. In the rest 5 bits, 4 bits are for the Hamming code checking.

Bit Position	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bit value	N/A	0	0	0	0	1	1	0	H_4	1	0	0	H_3	1	H_2	H_1

If one bit error is detected, the system can either drop the current image and do the image processing on the next video image or correct the error bit. The following example shows how to correct a single bit error of the 4x4 coding pattern. The four bit positions marked with H1~H4 are used for the check bits, which are derived as follows. The four-bit binary numbers corresponding to those bit positions having a binary 1 are added together using modulo 2 arithmetic and the four check bits are then the four-bit sum:

```
11    =1011
10    =1010
 7    =0111
 3    =0011
       0101
```

The encoded marker code is thus:

Bit Position	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bit value	N/A	0	0	0	0	1	1	0	0	1	0	0	1	1	0	1

In error checking, the four-bit binary numbers corresponding to those bit positions having a binary 1, including the check bits, are again added together and, if no errors have occurred, the modulo 2 sum should be zero.

```
11    =1011
10    =1010
 7    =0111
 4    =0100
 2    =0011
 1    =0001
       0000
```

Now consider a single-bit error. If bit 11 is corrupted from 1 to 0, the new modulo 2 sum would be:

```
10    =1010
 7    =0111
 4    =0100
 3    =0011
 1    =0001
       1011
```

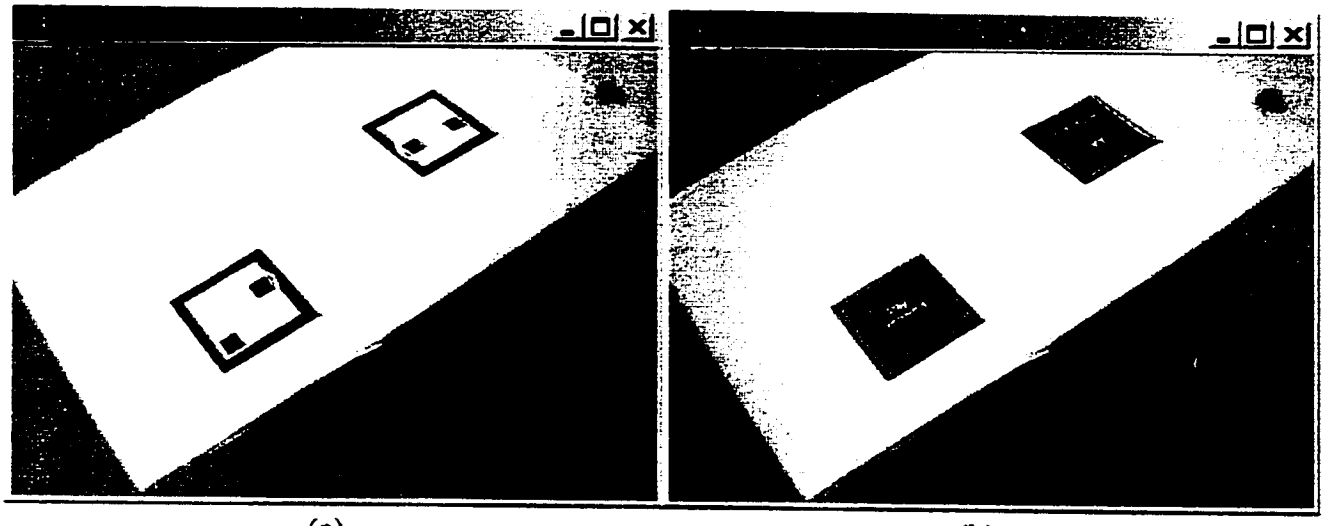
First, the sum is non-zero, which indicates an error. Second, the sum, which is equivalent to decimal 11, indicates that bit 11 is the erroneous bit. The error bit would therefore be inverted to obtain the corrected marker code.

For a 4 x 4 coding pattern, such a coding scheme requires five check bits since with this scheme the check bits occupy all bit positions that are powers of 2. Such a code is thus known as a (16, 11) block code with a rate of 11/16 and a redundancy of 1-11/16. For a 5 x 5 coding pattern, this coding scheme requires five check bits. Such a code is thus known as a (25, 20) block code with a rate of 20/25 and a redundancy of 1-20/25.

(5) Determine the orientation of the marker.

Four bits at the corners of the pattern determine the orientation of the marker in the camera's view dynamically. The system can keep tracking the orientation of the square marker, register the virtual model on the marker accurately even if the marker rotates, and read the binary code of the marker in correct order. In the user's view, no matter how the marker is rotated, clockwise or counter clockwise, the virtual object always keeps the relative pose to the marker. It looks like the virtual model sticks to the marker.

3.2.3 Camera Pose Estimation



(a) (b)

Figure 3.8 (a) Image of a white board, captured from a CCD camera. The black regions in the circles determine the orientation of the markers. (b) Computer generated graphic image superimposed on the white board.

In 1998, M. Jethwa, A. Zisserman and A. Fitzgibbon proposed an approach to solve the pose calculation with reduced DOF [25][26][27]. They proved that the projective mapping from 3D space to 2D plane can be reduced to a plane to plane homography (a linear transformation that registers a 3D plane between 2 images) by defining a world coordinates system for corresponding points on the plane to have z coordinate of 0. A minimum of eight parameters getting from the coordinates of four points on the image plane are required to solve a planar homography. With a single 2D plane in the 3D space, this approach makes a projective mapping to correctly align virtual objects with a plane in the world. This method has been employed for registering virtual and real objects without knowing the 3D position and pose of the camera relative to the real objects.

The recognized marker region is used for estimating the camera position and orientation relative to the marker tagged object. From the coordinates of four corners of the marker region on the projective image plane, a matrix representing the translation and rotation of the camera in the real world coordinate system can be calculated. Several algorithms have been created and implemented in experimental augmented reality systems. They are the Hung-Yeh-Harwood pose estimation algorithm [28] and the Rekimoto 3-D position reconstruction algorithm [19].

We currently use the algorithm proposed by Kato-Billinghurst-Weghorst-Furness [29], which incorporates camera lens distortion. Once the transformation matrix is calculated, all kinds of context information corresponding to the marker could be rendered upon the real world through human-machine interfaces.

3.2.4 Augmenting Virtual Environment and Context Information into Reality

In order to help the user to do work in the real world, the AR interface provides the user computer-generated information, which is context information of the objects appeared in the user's view. Currently, our system is implemented to embed the following context information into the real world from the user's perception.

(1) Text, such as character description of physical objects and instruction for performing real world tasks in form of annotation. For example, text instructions displayed beside the markers direct the user to open the computer case or guide the pipeline repairman to fix the pipe in red color.

(2) Audio, such as speech. For example, while a marker tagged on a switchboard which represents a Pentium 4 processor chip runs into the user's view, the user will hear voice instruction telling the user what kind of chip this marker represents and how to install it.

(3) Image. For example, if a marker is tagged besides a painting, the image mapped on the marker could be the photo of the artist.

(4) Real-time video stream. When working in the real world, doing surgery for example, some times we need a live guide from experts. By posting a marker besides the patient, the doctor can watch a remote real-time directive while keeping the operation part of the patient's body in his/her view.

(5) 3D model, created in OpenGL or VRML. If you want to see the internal structure of a machine without opening the case of the machine, you can do it simply by sticking a marker representing that machine on the case. When the marker runs into your view through the HMD, the internal structure in 3D model will be presented to you. Any object in the real world can have a corresponding virtual model created by OpenGL or VRML or other tools.

For a 4x4 square marker with hamming code checking function, 7 bits out of 16 bits of the marker code can be used to identify 2^7 different objects. For a 4x4 square marker with block sum checking, 5 bits out of 16 bits of the marker code can be used to identify 2^5 different objects. However how to decide what computer generated context information should be rendered on which object labeled with a square marker becomes an issue to solve. We implemented a database using C++. In the database, the key word for each record is the marker code ID, and the property fields are the context information the system will render when the corresponding marker labeled object runs into the camera's view (Figure 3.8).

3.3 Comparison with Other Visual Marking Technologies

Some systems, like the Augmented Reality for Construction (ARC) [30], use a 1D barcode as the marker to handle a large number of real world objects. The drawback is that an extra barcode reader is needed for each user. Another similar visual marking method adds color into the 1D barcode. Each bar in a special color represents a certain value. With the same amount of bars, the color barcode can identify more objects than a traditional black and white color barcode. One constraint is that the marker must be in an environment where daylight or fluorescent light is available. Otherwise the color of the bar will not be recognized correctly.

Two existing vision-based AR systems use a 2D square marker. In one system [29], the pattern of the marker region is compared with templates, which were given to the system before identifying a specific marker ID. The pattern template could be user names, characters or an image. A significant disadvantage of the method is that it does not have a specific way to differentiate one pattern from the other. The ability to correctly pick up the template matching the marker pattern mostly depends on the resolution of the camera and the algorithm to match the pattern. In another system, Jun Rekimoto proposed a method to use a 2-D matrix code marker as landmark in his vision-based system [17]. The complexity of detecting a 2-D barcode and extracting bar code ID depends on how the 2-D matrix code marker is defined. Compared to the marker definition method we proposed, the 2-D matrix barcode has some limitations. Every marker has a guide bar aside, which helps to locate the four corners of the marker and determine the orientation of the marker. Hence, an extra step is needed to find the guide bar. Since the

marker code is a binary code, error correction is possible to implement. The problem is that once an error bit is detected, the system will have to skip the current image and extract the marker code from the next available video image instead of correcting the error bit in the current image. In our experimental system, we found that the error bit doesn't occur independently from one image to another in that in most cases the error is caused by the low camera resolution, lighting in the environment and the material made of the marker. Therefore, even dropping the currently processed image when an error occurs, the system would probably not get the correct marker code in the subsequent video images if the environment does not change. A good method to solve the problem mentioned above is to add an error checking function into the pattern recognition algorithm. Block sum checking and hamming code checking are two methods being used. They both can correct any single bit error in the 2-D binary code block.

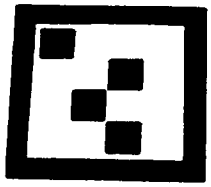
	Coding	Error Checking	Error Correction	Marker Orientation	Marker Shape
ARToolKit Marker	Template matching	no	no	template	square shape
CyberCode Marker	2D binary code	Yes	no	guide bar	rectangle
Binary Square Marker	2D binary code	Yes	yes	4 bits of the marker code	Square shape

Table 3.1: Features of markers in different tagging systems

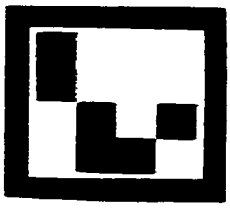
This raises a question. Why do we use a square block for checking? In a vision-based AR application, the object labeled with a marker could be of any size. In order to make our system recognize not only big but also small objects, we should minimize the marker size as far as possible. Suppose the marker code is a N-bit code with X rows and Y columns. The number of parity bits is X+Y. The total code is X x Y + X + Y bits. In order to minimize the total code length, X must be equal to Y.

$$
\begin{aligned}
& X \times Y = N \\
& X \times Y + X + Y = N + X + N/X \\
& (N + X + N/X)' = 1 - N/X^2 \\
& 1 - N/X^2 = 0 \\
& X = Y = \sqrt{N}
\end{aligned}
\tag{18}
$$

3.4 Binary Square Marker Tracking Results



(a) Maker1



(b) Marker2

Figure 3.9: Markers used in tracking

Marker1 Id: 1000 0010 0100 0010

Pattern Size	Visible Range (tilt 0^0)
9cm x 9cm	142cm
7.5cm x 7.5cm	128cm
4.5cm x 4.5cm	78cm

Marker2 Id: 1000 1000 0101 0110

Pattern Size	Visible Range (tilt 0^0)
9cm x 9cm	140cm
7.5cm x 7.5cm	126cm
4.5cm x 4.5cm	80cm

(a) Visible Range (tilt 0^0) for Marker1 and Marker2 with different pattern size

Marker1 Id: 1000 0010 0100 0010

Pattern Size	Visible Range (tilt 0^0)	Visible Range (tilt15^0)	Visible Range (tilt 30^0)	Visible Range (tilt 45^0)
7.5cm x 7.5cm	128cm	120cm	115cm	105cm

(b) Test result of maker1 without error control

Marker2 Id: 1000 1000 0101 0110

Pattern Size	Visible Range (tilt 0^0)	Visible Range (tilt15^0)	Visible Range (tilt 30^0)	Visible Range (tilt 45^0)
7.5cm x 7.5cm	126cm	120cm	118cm	110cm

(c) Test result of maker2 without error control

Marker1 Id: 1000 0010 0100 0010

Pattern Size	Visible Range (tilt 0^0)	Visible Range (tilt15^0)	Visible Range (tilt 30^0)	Visible Range (tilt 45^0)
7.5cm x 7.5cm	140cm	135cm	130cm	120cm

(d) Test result of maker1 with error control

Marker2 Id: 1000 1000 0101 0110

Pattern Size	Visible Range (tilt 0^0)	Visible Range (tilt15^0)	Visible Range (tilt 30^0)	Visible Range (tilt 45^0)
7.5cm x 7.5cm	150cm	140cm	130cm	120cm

(e) Test result of maker2 with error control

Table 3.2: Test results of vision-based tracking using Binary Squire Markers

A limitation of vision-based AR systems is that the context information of the objects represented by tagged markers can only appear when the markers are fully in the user's view. When occlusions occur, the marker ID and marker pose information are lost, therefore the graphic image of the 3D model and annotation will not be displayed to the scene. Since the marker recognition algorithm is based on individual video images, if the marker is partially covered by other objects, it's pose in the real world can't be estimated based on the current image, nor can it be calculated based on the marker pose information estimated from the last image. This limits the size and the volume of movement of the objects.

The proposed Binary Square Marker recognition algorithm was tested using two different markers. The 16bit binary code of marker1 is 1000 0010 0100 0010 (Figure 3.9(a)). The 16bit binary code of marker2 is 1000 1000 0101 0110 (Figure 3.9(b)). The data in table 3.2 (a) are the maximum visible ranges of marker1 and marker2 of 3 different sizes. These visible ranges were measured without error recovery. The maximum visible ranges for square markers placing in different orientations relative to the camera (tilted degree) were gathered and shown in Table 3.2(b-e). The data in Table 3.2 (b) and (c) are the test results of the object recognition algorithm without an error correction function. The data in Table 3.2 (d) and (e) are the test results of the algorithm with an error correction function (Block sum checking). In the test, the error control bits are encoded in the marker patterns. Although one marker pattern represents only one 16bit code, the marker IDs of the same pattern with error control and without error control are different. Since the error checking bits bring redundancy to the binary code of the marker, there are fewer error correction marker tags than there are non-error correction marker tags.

It is clear to see that the visible ranges of the markers are affected by the tilted degree of the markers. As the markers are more tilted, less patterns are visible. As a result the recognition becomes more unreliable. The larger the size of physical markers, the further away the patterns of the markers can be detected. The visible range of the markers with an error correction function is 12.8% larger on the average than that of the markers without an error correction function. In conclusion, by using the marker of the same size, the Binary Square Marker recognition algorithm we proposed can make the marker being recognized in a larger range in comparison with other 2D square marker recognition algorithms without error control. In the test, two markers are used. The pattern of marker2 (see Figure3.9 (b)) with larger black regions (black color grids are connected) can be recognized correctly in a larger range in comparison with marker1 (see Figure3.9 (a)), which has no connected black color grids. Therefore, we can say that the pattern of the marker also affects somewhat the visible range of the object tagged by the marker. Another important observation is that, when the marker is tilted, the probability that error bits occur on the edge of the pattern is very high. When we locate the error checking bits in the central area of the pattern, the probability that the error occurs on the error checking bits is very low. In this design pattern, the maximum visible range of the marker will increase and the pattern recognition algorithm will become more robust.

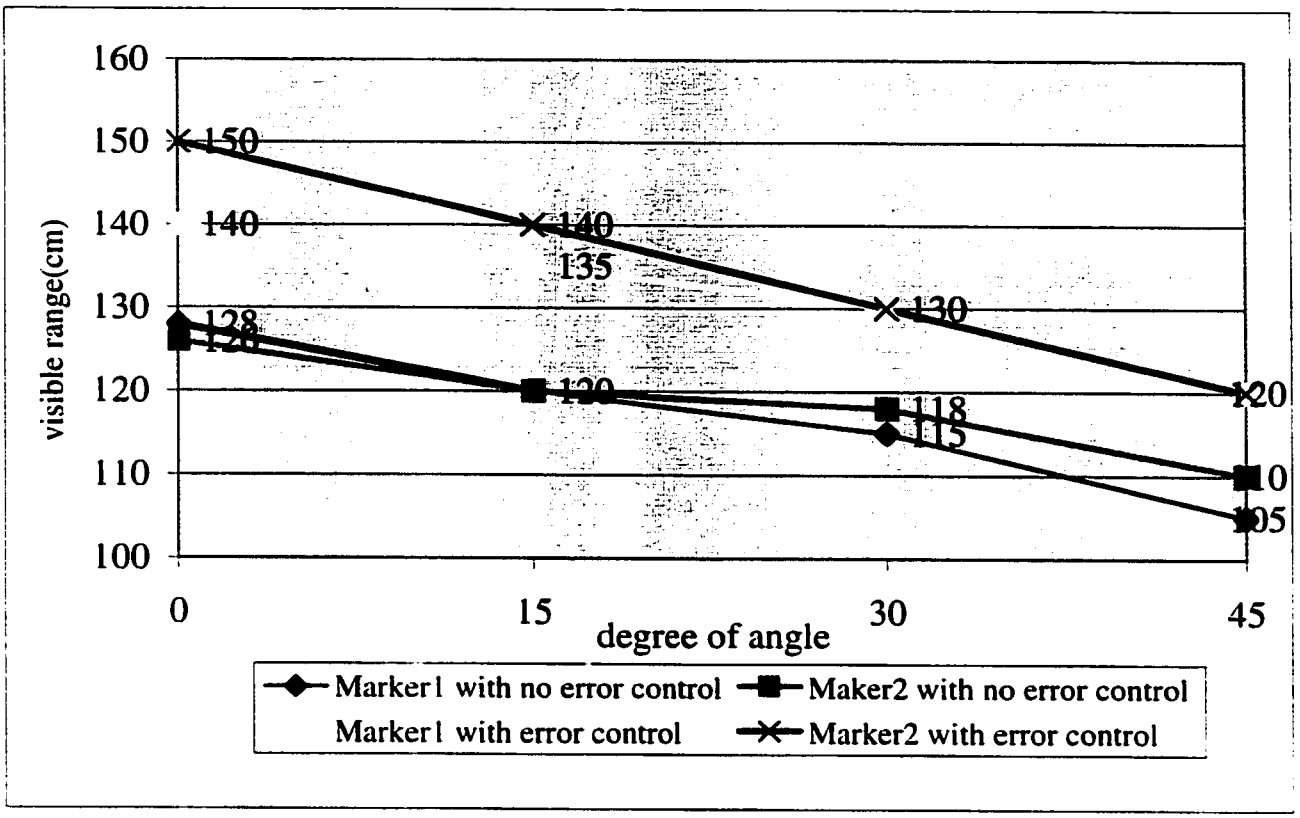


Figure 3.10: A line chart shows trends in visible range at equal intervals

3.5 Potential Applications Based on Augmented Reality Technology

In this section, we propose several potential applications using AR and Binary Square Marker recognition techniques. These new applications could be used in industry, entertainment and education. The development of these applications will be attempted in the near future.

3.5.1 Tracking

The Binary Square Marker could be used as 3D mouse in a Virtual Environment (VE), such as the CAVE™ system. The user in the VE holds an object with a marker tagged on it. A camera is put in a location where the marker will be in the camera's view when it moves in a certain area. When the user moves around, the camera can track the position and orientation of the marker, which means that the 6 DOFs of the user's hand are extracted from the known geometry of the

Binary Square Marker pattern. More than one mouse could be tracked simultaneously, if the marker codes of each mouse are unique.

3.5.2 Collaborative architecture design in a shared space

In doing industrial training, attending a conference or playing a game, people often work together face to face. That is a kind of traditional work style and most people intend to keep this tradition because by working in a shared space, such as a classroom or workshop, one can interact with others directly which will obviously improve the work efficiency. Augmented reality provides computer-generated information to the real world in a 3-D space. It helps users perform collaborative real world tasks. One potential application is Collaborative Architecture Design. Some markers representing virtual objects, such as an office building, a parking lot, a residence or a gym, are put on a big table. Several users wearing wearable computers and HMDs look at the big table. Users see graphic images of the virtual objects superimposed on the associated markers through the HMDs. The users can discuss the design of the campus and relocate any facilities either by moving the marker associated with the facility to a new location, or by translating or rotating the virtual objects through user interfaces such as keyboard and mouse. Those pose changes of one user will be distributed to the other users through a wireless connected server. The other users will be able to see the virtual objects in a changed position.

3.5.3 Mobile tour guide system

By combining augmented reality technology and the mobility of wearable computers, AR applications could be used anywhere beside the research laboratory. Wearing a wearable computer which has an AR system installed, a person can do many computer aided real world tasks in a workroom, museum, or even outdoors. In this part, we will introduce a mobile tour guide system, which could be used in an art museum. It is difficult for a visitor to understand the art pieces in an art museum very well without a guide. Although the visitors wish they could have their own guide so that they can navigate in the museum freely, this is not very practical. Even if the guide could talk to the visitor face to face about the work of art, the information the guide provides is very limited. The mobile tour guide system compensates the limitations of human guides. When the visitor sees the marker labeled on the art piece, for example a painting,

s/he will hear the music of the artist's time from the earphone, see the photo or video of the artist through the HMD, etc.

Chapter 4

4 System Architecture

4.1 Project Description

The field of Augmented Reality (AR) has existed for just over one decade, but the growth and progress in the past few years has been remarkable. Besides working on the basic enabling technologies, such as displays, tracking, registration, and calibration, researchers are considering problems of how users will interact and control AR applications, and how AR displays should present information.

Some objects may have behaviours that are not visible in the real world, but may be represented in a virtual environment. For example, electrical circuits operating at high frequencies emit electromagnetic radiation that can be rendered visible in a virtual environment. In this case, the virtual environment is not a mutually exclusive alternative to the real environment, but consists of additional information that can be overlaid on the real world so that both the virtual and real environments are visible simultaneously. The development of augmented reality interfaces for interacting and controlling both the virtual and real objects, as an extension of Virtual Environment research, is becoming widely popular. This would require a wearable interface to the virtual environment that would continually project the virtual environment into the visual field of the users as they walk through the real world. Rendering some behaviours of virtual objects would provide real-time computer-enhanced assistance for the user to do the work in the real world, which could be industrial training, product design, production, maintenance and product repair. For example, a computer in a virtual laboratory may be rendered in such a way as

to depict aspects of its operation, such as open sockets to other machines that are important to the user, but are not normally visible in the real world.

MCRLab[1] (Multimedia Communications Research Laboratory) is developing enabling technology and multimedia applications for Distributed Virtual Environments (DVE). The applications[2] allow multiple globally-situated participants to collaborate in industrial training situations over IP networks connected to heterogeneous computing resources and large data stores, investigate user perception and user needs, and design good DVE and Augmented Reality user interfaces. Our task in this project is to design an Augmented Reality prototype for the industrial training application that will be the focus and emphasis of our research. This, for example, will permit a "trainee" to overlay on equipment 3D graphics of the internal structure of that equipment to which the trainee has to provide repairs or maintenance. This AR application will greatly facilitate those repair and/or maintenance functions.

The aim of this thesis is to use augmented reality technology to provide virtual reality users with an augmented reality interface through which the user can interact with both the virtual and the real world intuitively. The implemented interface is expected to enhance the user's perception of and interaction with the real world. In order to provide accurate alignment of real and virtual objects, a vision-based tracking technique is added in the system.

The augmented reality prototype uses a see through display combined with a tracking device to overlay upon the display graphic images and to provide links to a multimedia information database about all the context information of the objects in the real world tracked by the tracking subsystem. The database may provide the user with the object's properties, its internal structure in 3D, its maintenance history, operation/repair instructions, or the object's video information, etc. To better understand the prototype, consider the following scenario:

---

[1] The MCRLab is a laboratory in the S.I.T.E., University of Ottawa.
http://www.mcrlab.uottawa.ca

[2] Project: Distributed Virtual Environments with Training Applications (DIVERTIONS).
Sponsor: National Capital Institute of Telecommunications (NCIT).

A worker is standing in front of an assembly line. His work is installing chips onto computer main boards. By wearing a see-through HMD with a wearable computer in his backpack and a small keyboard mounted on his arm, his hands are free for operating circuit board and chips. The worker sees the markers tagged on the circuit board in the real world through the HMD. Each marker represents a chip that should be installed onto the area tagged by the marker. The worker also sees the graphic image of the 3D model of the chip and annotations, such as the installation instruction and chip parameters table, overlaying on the marker. While viewing the overlaying information, the worker can also hear the speech instruction through earphones mounted on the HMD. The context information provided by the prototype can become a useful tool in various facility installation/maintenance related applications. It can also be used as an educational tool. In addition to the chip installation guide and the 2D drawing, the 3D model superimposed on the real main board in the real space observed by the worker can help him/her better understand the work and save a lot of time for checking the circuit board drawing in the guide book.

For implementing the project two separate tasks were done. Onc task is creating the multimedia context information database. The other task is implementing the augmented reality displaying system. Figure 4.1 shows the flow diagram of the system.

The Data Creating Task creates context information for every unique marker. There are links between all kinds of context information and their corresponding marker ID stored in the local media database. The context information of a marker which represents an object in the real world could be a video stream, which comes from a remote computer in case the IP address of that computer is provided in the database, a JPEG image, an audio file, text, or a VRML 3D model. The video stream is created on the remote computer and sent out in real time. The local wearable computer receives the video stream and renders it to the video see-through display in real time. The non real-time information is created in advance by special tools and stored in the local computer hard disk.

Tools currently used to create and edit context information in the first task of our system are:

1. Video stream

   Tool: MS Vision SDK 1.2

   Function: Camera image capture

   Tool: Intel JPEG Library Version 1.51

   Function: Image encode/decode

   Tool: MS WinSock Versions 1.1

   Function: Video stream transmission through UDP socket over the internet.

2. JPEG image

   Tool: MicroSoft PhotoEditor

   Function: Image editing

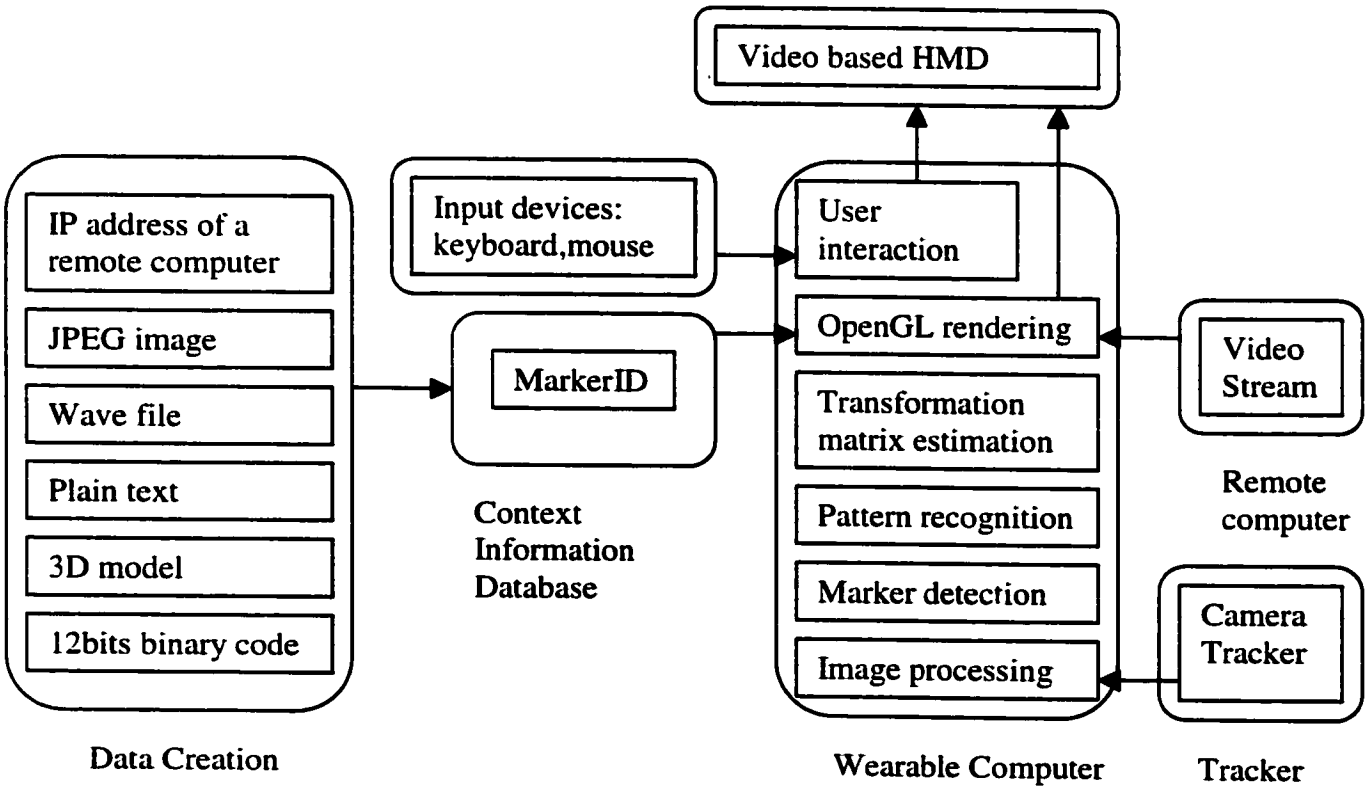


Figure 4.1: Augmented reality system diagram

3. Audio file

   Tool: MS Sound Recorder

   Function: Speech recording, editing. The voice is recorded through the microphone.

4. Plain text

   Tool: MS notepad

   Function: Plain text editing

5. 3D model

   Tool: Cosmo world 2.0

   Function: 3D model creation

   Tool: VrmlPad 1.3

   Function: VRML text file editor

The second task is to implement the displaying system on the wearable computer. It extracts the geometrical and context information for the viewing space from the context information database. After the information is loaded, the position and orientation of look for the user are continuously calculated based on the data from the tracker. Simultaneously, the geometry is transformed using transformation matrices based on location and direction data to produce an image that aligns with the objects in the real view. The following sections discuss in detail the various considerations in the choice of hardware and the design of software.

4.2 Hardware Requirements

The research did not include the design of any hardware. Commercially available products were integrated with the program. The critical components for any wearable computing system are a display unit, tracking devices, a mobile computer and appropriate input devices. A few commercial wearable computers are available. The wearable computer we use is not as powerful as the desktop computer in processing speed, computer graphic function, and hard disk capability. The input device for the wearable computer, such as keyboard and mouse, is not as easy to use as in a desktop computer. Therefore, most of the development work was done on an

IBM desktop computer with a Pentium III processor and an AccelGMX2000 graphic card with 16MB display RAM and OpenGL 1.1 support. For this project, we use a wearable Xybernaut MA IV computer to run the augmented reality prototype and present the demo to students and professors in the testing phase. The Xybernaut Mobile Assistant IV (MA IV) is the most

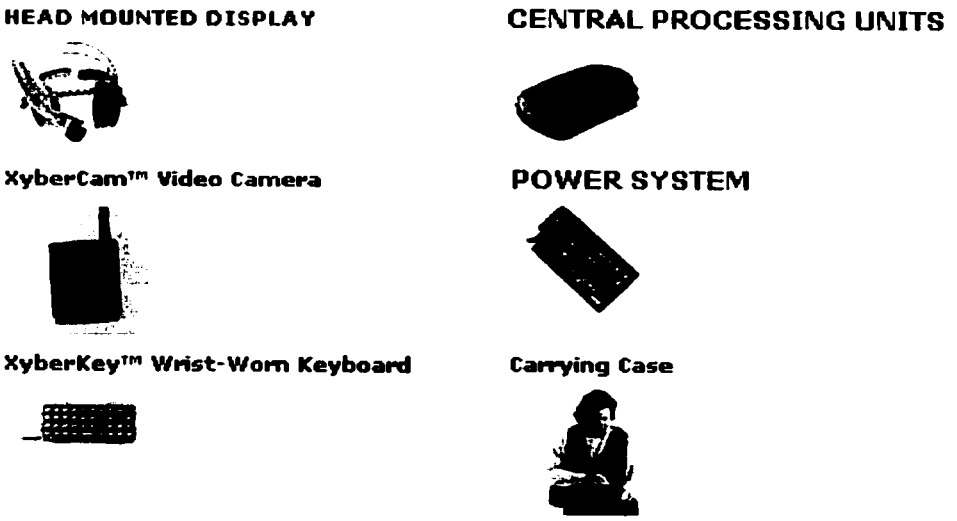


Figure 4.2: Xybernaut Mobile Assistant IV components diagram

advanced wearable computer in the world - a featherweight computer with all the functionality and connectivity of a fully-functional networked desktop PC. It works where laptops or palmtops cannot. Available with a head mounted color display and a hands-free voice recognition and activation, the MA IV makes it possible for workers to access data, file reports, send e-mail and connect with the Internet almost anywhere under an engine, atop a utility pole, beside a pipeline, inside a clean room or walking down a warehouse aisle. Operation can be performed non-stop, without interrupting a worker by looking away or letting go of a safety grip. The next section deals with each component in detail (Figure 4.2) [31].

4.2.1 Head Mounted Display

Humans use the visual sense more than any other sense to process information. Video images captured by camera in the vision based AR system are critical information for tracking. Hence, the capabilities of the display device become much more critical. The display device has to be able to provide necessary quality at an acceptable cost. Some factors are to be considered in choosing a display device for an augmented reality application.

1. *See through display* is required for this project. It allows the viewer see the real world image and graphic image projected on a special optic.
2. *Display resolution* is the number of pixels in the horizontal and the vertical directions making up an image. This resolution affects the quality of the rendered image output directly. Low-resolution display will make it difficult for the viewer to distinguish between separate entities.
3. Ideally the *FOV* of a display device should be equal to the FOV of human eyes, which is about 180^0. However the FOV for most systems varies from 30^0 to 60^0.
4. The display device for the AR system should be *color capable* for more readability of augmented information and greater realistic feeling of the world.
5. It is not inconvenient for the user to wear on the head for a long time. Especially for carrying out certain physical labor, such as maintenance related work, the device shouldn't be too heavy to exhaust the user.
6. Since the user is mobile in his/her work place, the system has to operate on batteries, which makes the *power* consumption a key factor.

The XyberView Head Mounted Color Display (HMD) allows you to operate the MA IV hands-free, giving you safety, convenience and productivity. The HMD uses a head-mounted microphone, earpiece speaker and works with any standard voice activation program. The HMD can be adapted to a hard hat or other protective head gear, multiplying the environments in which the MA IV increases productivity, without compromising the safety.

A HMD contains:

Headset -- Holds the display unit and the optional XyberCam™ video camera.

Display Unit -- Projects an image onto a reflector (a partially transparent mirror) that displays the image to your eye. The VGA display measures just 1.1" diagonally, but using special optics, provides the same full-color, full-screen view as a 15" monitor seen from two feet.

XyberCam™ Video Camera.

4.2.2 Tracking Device

Tracking is used where the orientation and the position of a real physical object are required. For augmented reality applications pose tracking of the viewer are critical. First the tracking system locates the user in the space using position tracking. In this step, the coordinate (x,y,z) of the user with respect to a reference point is calculated. Then which direction the user is looking in is to be determined. This requires the orientation to be specified by three angles called pitch (elevation), roll and yaw (azimuth). After that the complete pose of the viewer is described .

For an AR system to be successful, the 6DOF tracking should be done continuously as the user may move and therefore alter his/her position and orientation. A Tracker could be used to track the motion of the user's head, hands or eyes. Different tracking techniques are used for different applications. In the AR system we developed, a camera is used for tracking. The computer extracts and estimates the marker's pose relative to the camera by processing the camera captured video images of the objects in real space. Chapter 3 gave an introduction to the vision based tracking system in detail. In addition to camera tracks, there are magnetic, mechanical, optical, acoustic and inertial trackers. These kinds of trackers can be mounted on a glove, body suit devices, etc., to provide tracking of a user's hand, head or some other body part.

The registration errors and a delay due to the time interval between measuring the head location and superimposing the corresponding graphic images are the biggest challenges for AR, which will cause misalignment of real and virtual objects and the virtual objects appear to swim around real objects. The critical characteristics for a tracking device are as follows.

The *resolution* to locate a reported position;
The *range* within which a measured position is correct;
The *sample rate* at which sensors are checked for data;
The *update rate* at which the system reports new position coordinates to the host computer;
The *time delay*.

The tracking device in our system is XyberCam™ Video Camera -- It transmits or records video images. Using this digital color camera, we can capture and transmit still and motion images

through a local area network, a wide area network, on the Internet, or record images to the MA IV hard drive.

Video camera specifications:

VGA color monocular 640 X 480 resolution

FOV Horizontal: 20° Vertical 15°

Left or right side wearable

Display Unit: Length 61mm (2.4 in), Width 89 mm (3.5 in), Depth 256 mm (10.1 in), Weight 280 g (0.62 lbs)

4.2.3 Computing Device

The ideal computing device for the AR applications is the wearable computer, for it provides maximum freedom to the user in terms of movement, ruggedness and ergonomics. Next, we will depict some important features of the computing device of this project.

1. The Central Processing Unit (CPU) is the brains behind the MA IV™. A fully functioning Pentium MMX PC, the CPU includes:

Intel Pentium 233 MHz MMX processor

Industry-standard operating system (Windows98).

160 MB RAM

4.3 GB hard drive

Long life, hot-swappable, lithium-ion battery

The light-weight MA IV clips onto a belt. It contains ports for connecting peripherals and power cables as well as PCMCIA cards.

2. Video

Support for simultaneous use of HMD or FPD and external VGA monitor (640x480)

1MB Video Ram

Video NeoMagic MagicGraph 128ZV

3. Audio

Soundblaster compatible audio chip (ESS Technology)

4.2.4 Input Device

The interaction methodology with the system is a research topic itself. The input device in a mobile computing device is very critical because the user need to interact with the system to get information, such as graphic images, video or annotations, from the computer while performing some other tasks, for example opening a computer case or doing surgery.

The input device used in the project is XyberKey™ Wrist-Worn Keyboard - The lightweight QWERTY full-function keyboard is an activation option for the MA IV. Using the keyboard, you can compose and file a complete report on-site.

4.3 Software Design

The models implemented in the system architecture are introduced in this section. Figure 4.3 in a high level describes the components within the application and the data stream flows among

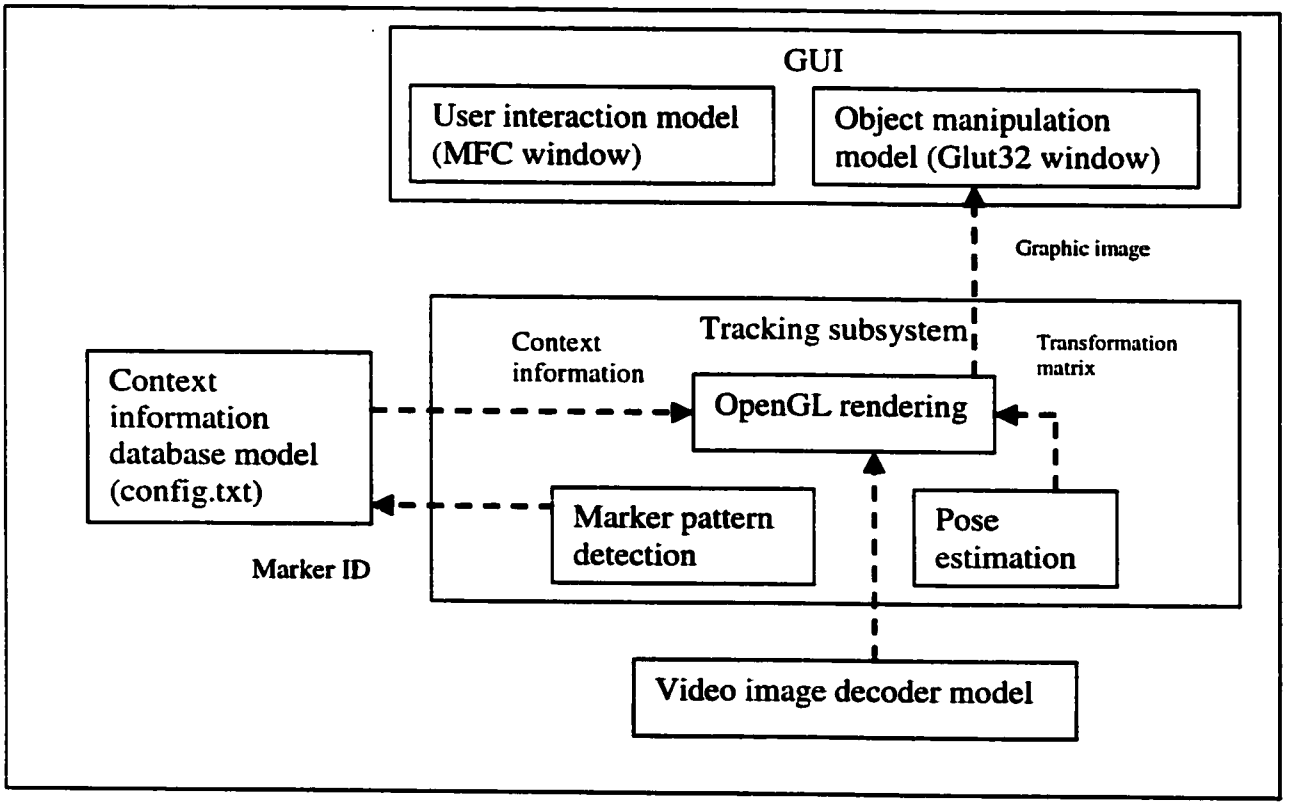


Figure 4.3: Display system architecture

them. The models in figure 4.3 are all running in the local wearable computer. When a user wearing a mobile computer wants to watch a real-time video program captured by a camera equipped in a remote computer, the user just inputs the IP address of that computer through a user interaction GUI. The local system requests the video stream from the remote node to the local node through a local area network or the internet. The user can view the video image rendered in 3D space mapping onto the surface of objects in the real space. The focus of the project is to provide the system all kinds of computer-generated information, which helps to enhance the user's perception and understanding of the real world. Quality of service for the video stream transmitted over the internet is not a research topic in our project as of today. Figure 4.5 illustrates the connection to different nodes through the network. The connection can be made using the UDP protocol. Although the UDP protocol provides unreliable connection, it is not as time consuming as the TCP protocol, which makes it meet the requirement of real time transmission of a video stream over an IP network.

4.3.1 Designing the Local Node

The display system is running in the local wearable computer. It is linked to a local database and written in C++ language. We are using the Microsoft Visual C++ 6.0 compiler to develop the application.

Context information database model

The database is defined in the text file "config.txt". All the records in the database are stored in the file. The program design is isolated from database design, which lets the user add, remove or change the fields and their properties in the wearable computer or any other computer even if the AR application is running locally. Since the "config.txt" file is created in plain text, it is easy to be edited by the user in popular text editors, such as NotePad or WordPad. No additional software is required. An example of a record defined in the database is given in Table 4.1. The context information stored in the database or linked to the database is defined in advance by the data creation task and stored in the local disk. On launching the program, the updated database and linked 3D geometry are loaded.

Field symbol	Property	Note
!	0001 0000 0010 0010	16bits Marker ID
&	Computer Name: BLUE15 Config: PIII, RAM 125MHz, HardDisk 10G, WINNT4. User Name: Peiran Liu.#	Annotation or IP address
$	./wrl/dodec1.wrl	A link to 3D model file in VRML
@	./sound/P4Sound.wav	A link to audio file

Table 4.1: An example of a record in media database

The marker pattern detection model as a part of tracking subsystem is implemented in three steps.

Step 1: Camera calibration

Camera calibration is the process of estimating the intrinsic and extrinsic parameters of a camera, which are used to determine the relationship between what appears on the image plane and where it is located in the 3D world. *Registration* is the process of acquiring the objects' pose. *Camera tracking* can be regarded as a continuous update of extrinsic parameters. Camera tracking addresses the problem of accurately tracking the 3D motion of the camera in a known 3D environment and dynamically estimating the 3-D camera location.

Camera calibration parameters are calculated by using the *calib_cparam* application included in the ARToolKit software package, which is developed by the University of Washington in 1999 and is free to the public for non-commercial use. Camera calibration parameters are used in step 4 for calculating the transformation matrix.

Step 2: Image processing.

In vision based AR systems, the video image is critical for tracking. We implemented the image capture by calling MS Vision SDK 1.2 Lib functions. A code example is as follows.

```
// Connect to the VFW camera
CVisImageSource imagesource = VisFindImageSource("");
CVisSequence<CVisRGBABytePixel> sequence;
sequence.ConnectToSource(imagesource, true, false);
sequence.ImageSource().SetUseContinuousGrab(false);
// Grab an initial image
sequence.Pop(image, 40000);
```

The image labeling process groups pixels together into regions of similarity according to the rate of changes of their intensity. After that, the contour of the region is extracted by taking the outside line of the region and recording the coordinates of the pixels lying on the line. Those regions that are in shape of a parallelogram are labeled as candidates for marker regions. The implementation of how quadrilateral regions are found in the captured video images is made by calling GetContour() and check_square() functions imported from the HITL Shared Space library [7].

Step 3: Marker detection and pattern recognition

In order to track multiple markers, the marker patterns are extracted from all the candidate marker regions and represented in binary codes, which are the marker IDs. A Marker ID is recognized using computer vision technology. The marker detection algorithm was introduced in detail in *Section 3.2.2 Pattern recognition.*

Pose estimation model

A marker is a 2D plane in 3D space the value of the Z-axis of which is 0. Every marker in real world space has its own distinct coordinate system. The algorithm proposed by Kato-Billinghurst-Weghorst-Furness [29] is used to acquire the transformation matrix or the relation between 2D screen coordinates of the marker on the video image and its 3D coordinates in the real space. This transformation matrix is calculated for each marker detected from the camera's image plane and is used for graphic image rendering. Please refer *to Section 3.2.3 Camera pose estimation* for the algorithm description.

Video image decoder model

The video stream sent from a remote node to a local computer is in form of an encoded JPEG image sequence. The functions of this model are building a connection with the designated remote computer with the UDP protocol, receiving compressed images frame by frame in a package from the windows socket, decoding the JPEG images and finally sending the uncompressed images to the OpenGL rendering model for display.

The MS Winsock library v1.1 is used to implement the network transmission. JPEG image decompression is done by calling the functions in Intel JPEG Library v1.51.

OpenGL rendering model

OpenGL rendering is a routine called from the GLUT main loop '*void mainLoop(void)*' repeatedly when the system is idle. In this routine, the bulk of OpenGL rendering function calls were made. According to figure 4.3, three data streams flow in and one flows out of this model.

The marker pattern detection model sends the detected marker ID to the context information database model in which the context information of the object tagged by the marker is extracted and sent to the OpenGL rendering model. In case the annotation text of the maker is an IP address, the OpenGL rendering model will call the Video image decoder model to build the network connection and send the Video image back to it. Once this model gets all the context information of the currently tracked marker, it will call the Pose estimation model to calculate the view transmission matrix from 3D coordinates to 2D coordinates based on the maker coordinates in image plane. Finally, the model calls the OpenGL rendering functions to transform the 3D model, annotation and video image from 3D to 2D in reverse of the transformation from 2D screen coordinates of the marker on the video image plane to it's 3D coordinates in the real space.

Object manipulation model

In the AR system, the graphic image generated in the OpenGL rendering model and the video image captured by the video camera in the local wearable computer are overlaid together and displayed onto the HMD of the wearable computer. In software design, we open an OpenGL GLUT32 window in full screen to display the overlaid image. Some function calls of the GLUT32 library were made to manipulate the window size and popup menu. The user can interact with the application by using the Wrist-Worn keyboard of the wearable computer in the GLUT window and see the response of system from the HMD (Figure 4.4).

Building an intuitive interface for the user to interact with both the virtual and the real world is a major research topic of the project. In addition to creating a window to display the images, the object manipulation model built a control mechanism for the user to translate and rotate the virtual models rendered in the scene. This mechanism fully utilized the function of the keyboard that makes the user to manipulate the virtual models more precisely in comparison with the function of the mouse, which has only 3 keys to operate. The following source code illustrates how to put the keyboard into use.

```
switch (key){
        /* translate along x axis, y axis, z axis */
        case 'q':mx=mx+3;break;
        case 'w':mx=mx-3;break;
        case 'a':my=my+3;break;
        case 's':my=my-3;break;
        case 'z':mz=mz+3;break;
        case 'x':mz=mz-3;break;
        /* rotate along x axis, y axis, z axis */
        case 'o':rx=(rx+2.5);break;
        case 'p':rx=(rx-2.5);break;
        case 'k':ry=(ry+2.5);break;
        case 'l':ry=(ry-2.5);break;
        case 'n':rz=(rz+2.5);break;
```

```
        case 'm':rz=(rz-2.5);break;
        /* scale the model */
        case 'c':ms=ms+1;break;
        case 'v':ms=ms-1;break;
       /* recover the pose of the model */
        case 'b':mx=0;my=0;mz=0;rx=0;ry=0;rz=0;ms=1;break;
        default: break;
}
```

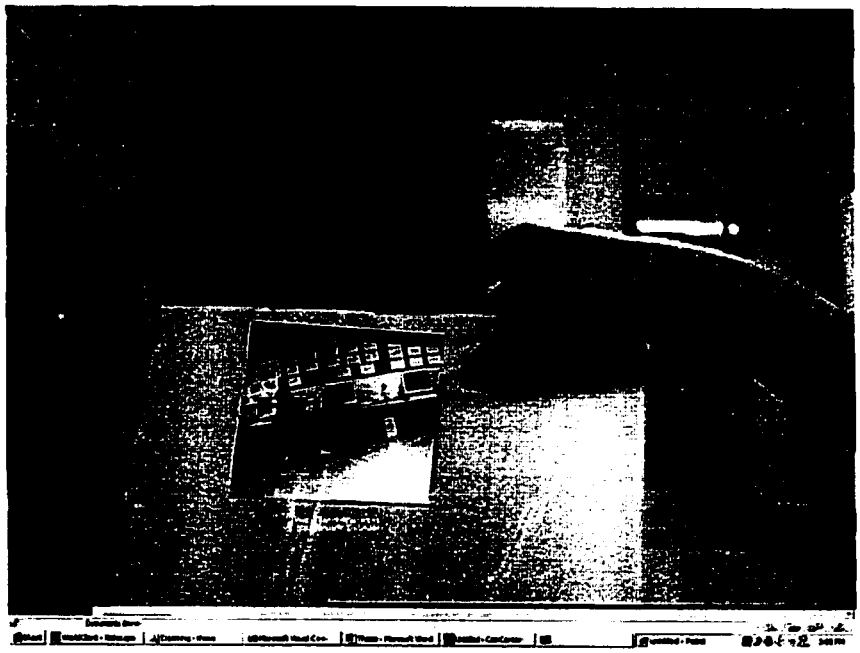


Figure 4.4: OpenGL image display GUI

User interaction model

When a user starts the AR application, a MFC dialog window will pop up to the screen at first. In this window, the user can open a GLUT window for displaying images by selecting the menu item in the dialog window. If the system is currently connecting to a remote node and is displaying the video stream transmitted from that node, the user can set the transmission rate (image/sec) manually. He/she can also suspend or resume the transmission by just pressing a button in the dialog. It is easy for the user to switch the connection from one node to another just by input a new IP address. When the program is running, the user can switch between the GLUT window and the MFC dialog (Figure 4.6).

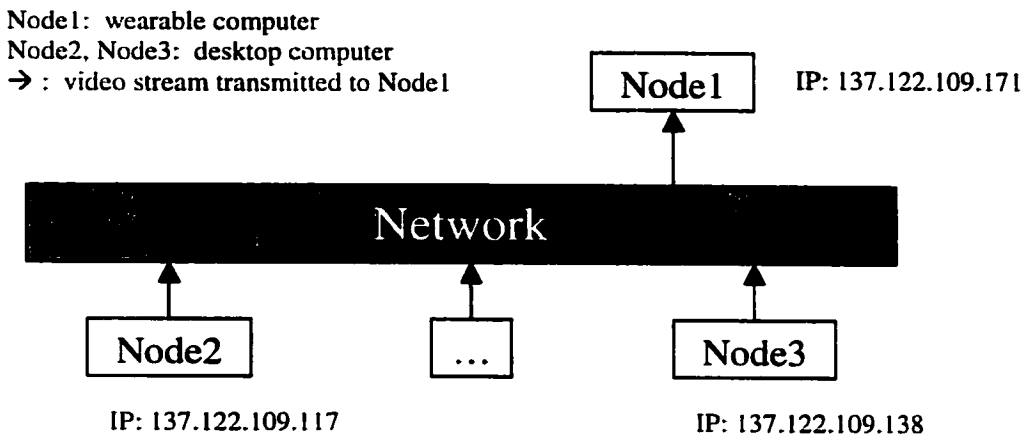


Figure 4.5: Network connection

4.3.2 Designing the Remote Node

In the project we developed a simple program running on the remote node. The network connection is illustrated in Figure 4.5. Three functions of the program are described in the following part.

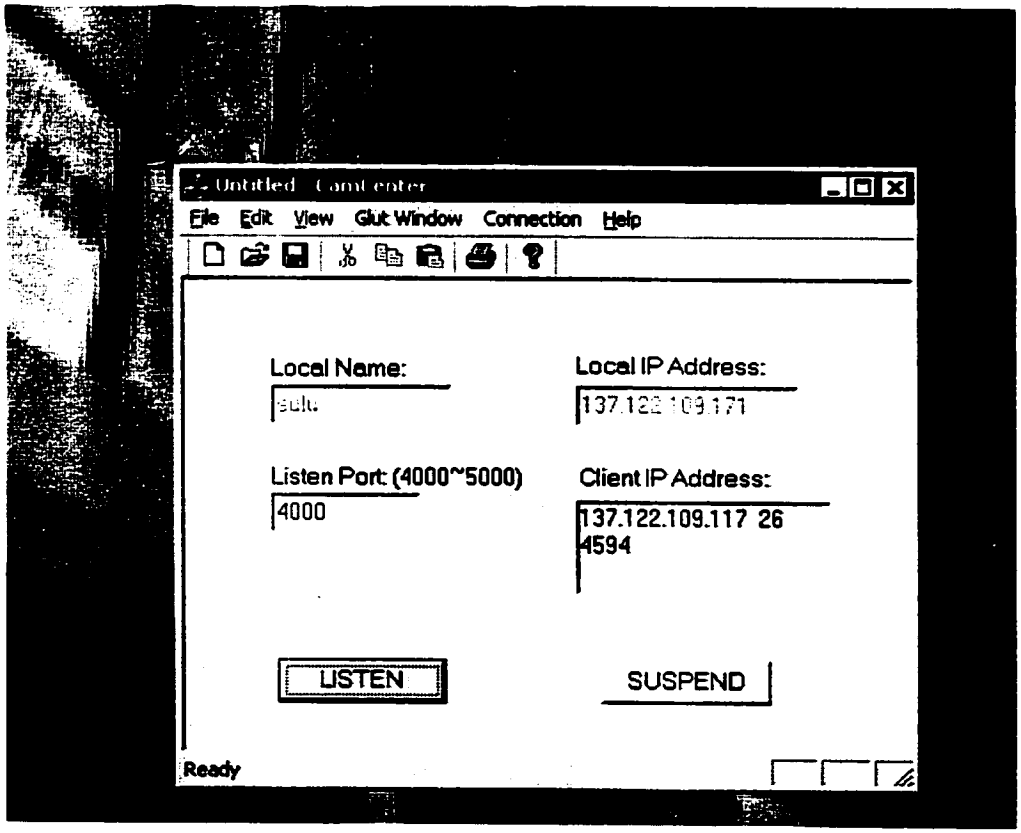


Figure 4.6: Configure dialog in local node

GUI

The GUI in this program is developed in C++. The program must be started before running the AR application in the local wearable computer. The user interface of this program is a MFC dialog window shown in figure 4.7. The user can designate the node to which the real time video stream captured by video camera will be sent by just inputting the IP address of the receiver in the dialog window. The user can also suspend or resume the transmission through this user interface.

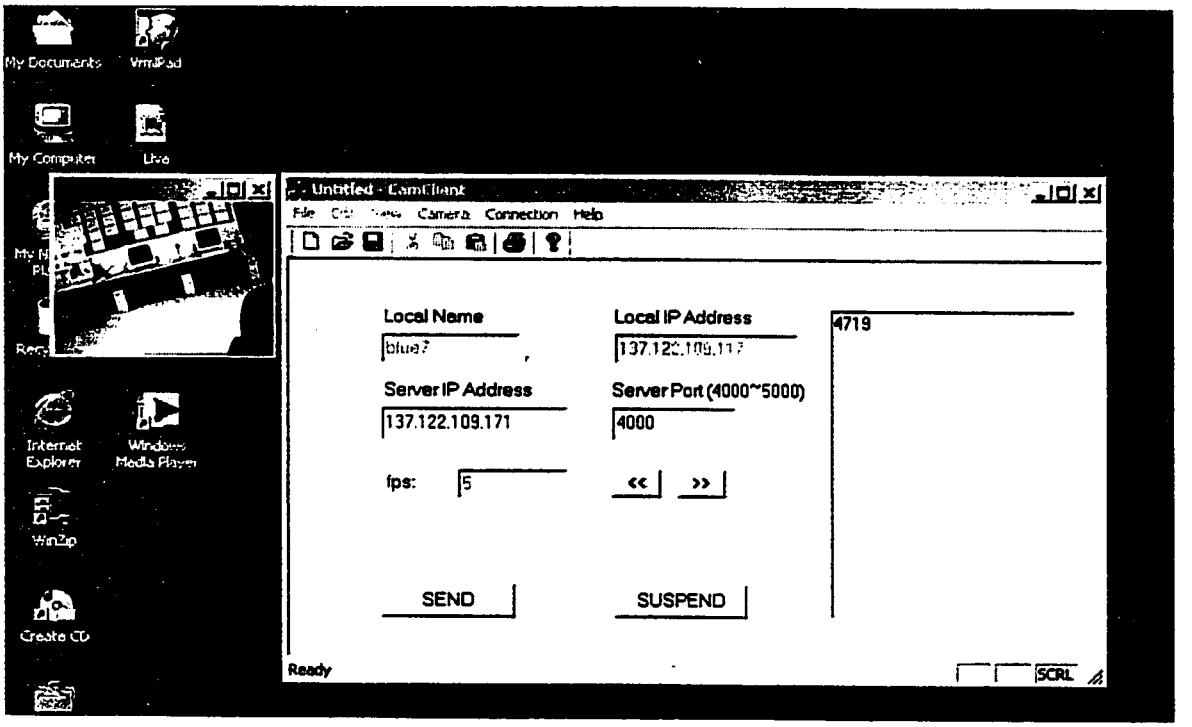


Figure 4.7: Configure dialog in remote node

Video image encoder

In order to get a compressed video stream for transmission over the network, the program first calls the MS VisionSDK library to capture video images from the video camera on the remote node, then put the sequence of images into a buffer. Next, the images are encoded from BGR pixel format to JPEG format frame by frame using the Intel JPEG Library v1.51.

Video stream transmission

The video stream sent from the remote node to the local node is in form of an encoded JPEG image sequence. The connection between the two nodes is built with the UDP protocol. The remote node sends compressed images frame by frame in packages through the windows socket. The MS Winsock library v1.1 is used to implement the network transmission.

Chapter 5

5. Evaluation of the Real-time Vision-based AR System

In this chapter, we discuss performance issues of the real-time vision-based AR prototype using the enriched 2D matrix code marker (Binary Square Marker).

5.1 Observation and Discussion

Based on the new Binary Square Marker object recognition algorithm, the system can recognize multiple objects that appear in the user's view in real-time (12 frames/sec) on a Pentium III workstation over a MS NT4 platform with a graphic device. The frame rate of the application is affected significantly by several factors. (1) The graphic capability of the computer. Since a major task of the application is to render graphic images of 3D models and to display video images and graphic images to the screen, all these works require the computer to have powerful graphic processing capability, otherwise the application can only ran in very low frame rate. To solve the problem, we installed an AccelGMX 2000 Video Accelerator on the workstation, which supports OpenGL graphics. (2) The complexity of the 3D model. In general, the more complex the 3D model is, the more slowly the graphic images are rendered. The complexity of a 3D model is measured by its properties, such as the total number of faces and vertices, animations, lighting, textures, materials, and so on. It cannot be measured by just one aspect. The virtual models we are using in the prototype are in VRML, and their sizes are smaller than 600KB. Since the VRML is a universal description language for describing 3-D shapes and scenery, we deem the size of a VRML file as one way to estimate the complexity of a 3D model. (3) The number of markers being recognized in the user's view. Due to the low resolution and

the limited Field of Vision (FOV) of the video camera, our system can recognize up to 8 markers in one scene under the constraint of minimum 10f/s frame rate.

We also run the prototype on a Xybernaut MA IV wearable computer with a 233MHz Pentium MMX, which does not support OpenGL graphics. The processing rate there is limited to 3 frames/sec. Running in the wearable computer, more than half of the processing time is actually consumed in graphic rendering. This observation implies that enhancing the computer graphic hardware in the wearable computer can significantly improve the performance.

The Binary Square Marker tracking algorithm is based on the ARToolKit system and the CyberCode system. It takes the advantages of both systems, such as the template based orientation and the square shape marker in the ARToolKit system and the binary coding in the CyberCode system. At the same time, it overcomes the disadvantages of both systems, such as the ambiguity of the template design in the ARToolKit system and the lack of error control in the CyberCode system. In practical experiments, 4x4 binary square markers are used. For a 4x4 square marker with hamming code checking function, the proposed object recognition algorithm can detect 2^7 objects in the real world. For a 4x4 square marker with block sum checking function, the proposed object recognition algorithm can detect 2^5 objects in the real world. When the matrix of the marker pattern becomes larger, the system can theoretically detect more objects. In practice, to recognize more objects, we not only have to increase the complexity of the marker pattern, but also need a good quality video camera with higher resolution.

5.2 Subjective Evaluation

Subjective evaluation tells the evaluator how the users feel about the system being assessed. This is different from how effectively or how efficiently they perform with the system. It complements data from efficiency and effectiveness measures. Subjective assessment usually produces a list of satisfying and unsatisfying system features which is especially useful if testing is taking place during development. The usual method of assessment is to use a standardized opinion questionnaire to avoid criticisms of subjectivity. In the evaluation, respondents are asked to fill out a questionnaire immediately after the usability testing session. We also used a

"screening questionnaire" to get some background data on each respondent. The Subjective evaluation questionnaire and the Screening questionnaire were designed according to the Software Usability Measurement Inventory (SUMI), a *de facto* industry standard evaluation questionnaire for assessing quality of use of software by end users [32]. In the evaluation process, twenty (20) persons from the engineering school and from other universities were invited to do the test. Sixteen (16) persons participated in the evaluation.

Summary statistics:

1. Individual user profile analysis table for the Screening Questionnaire

The inventory has 6 statements. Against each statement there are three or four choices. 16 User profiles are listed.

Total users: 16

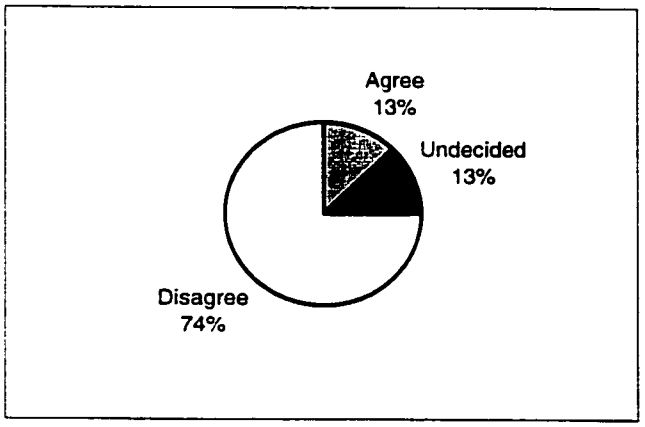
Question	Answers			
Age range	• 18 – 24 (6.25%)	• 25 – 34 (81.25%)	• 35 – 45 (12.5%)	• Over 45 (0%)
For how long have you been using computers in your work	• Less than 1 year (0%)	• 1 – 3 years (6.25%)	• More than 3 years (93.75%)	
How much information have you gotten about Augmented Reality?	• Not at all (6.25%)	• A little (31.25%)	• Some (43.75%)	• A great deal (18.74%)
Frequency of use of the system being evaluated.	• None (37.5%)	• Seldom (25%)	• Occasionally (31.25%)	• Often (6.25%)
Have you received any training with this system?	• None (68.75%)	• 1 day (25%)	• More than 1 day (6.25%)	
Job level	• Researcher (81.25%)	• Developer (12.5%)	• Manager (6.25%)	• Others (0%)

2. Item analysis diagrams for the Subjective Evaluation Questionnaire

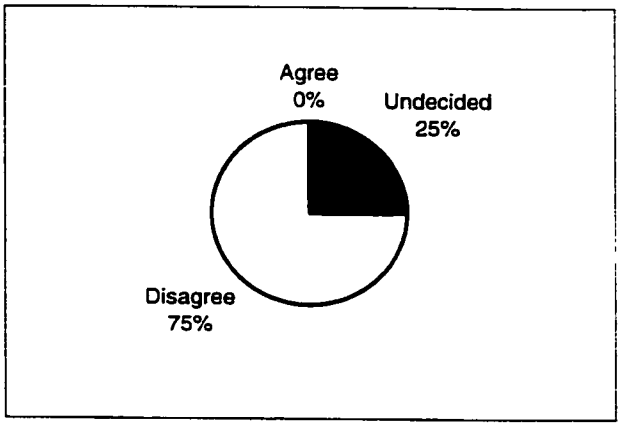
The inventory has 16 statements. Against each statement there are three choices. Users should mark the first choice if they generally AGREE with the statement; mark the second choice if they are UNDECIDED, or if the statement has no relevance to the system or to the situation; mark the third choice if they generally DISAGREE with the statement.

Total users: 16

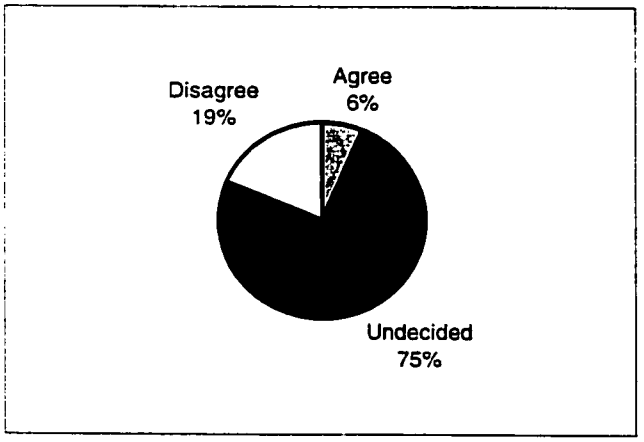
1. This system responds too slowly to inputs.



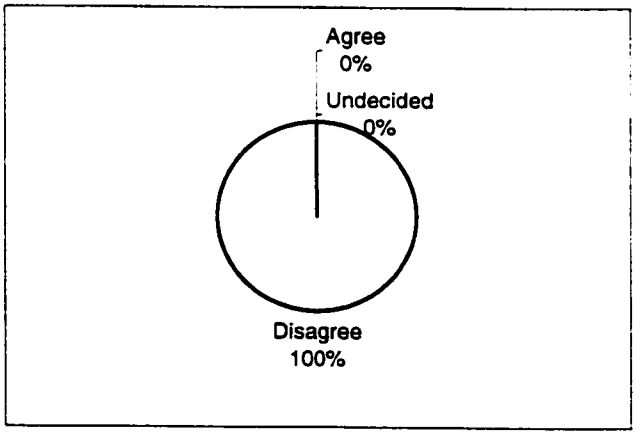
2. The system has at some time stopped unexpectedly.



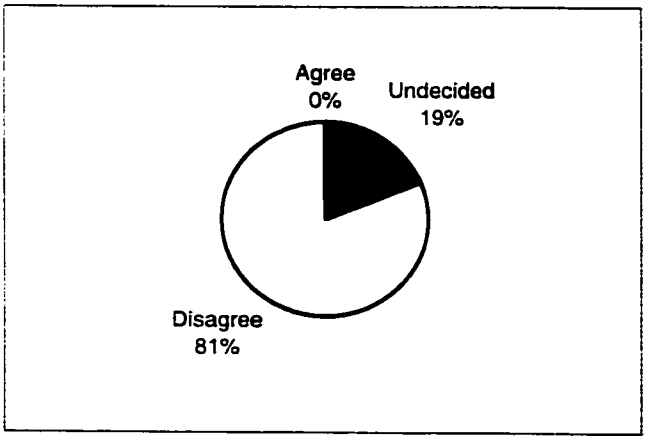
3. If this system stops, it is not easy to restart it.



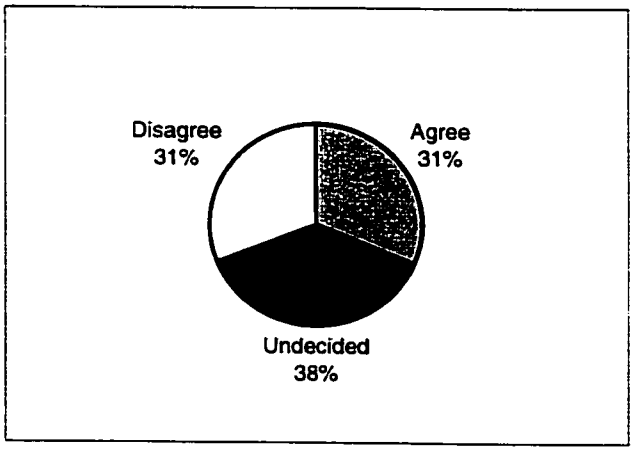
4. Learning to operate this system initially is full of problems.



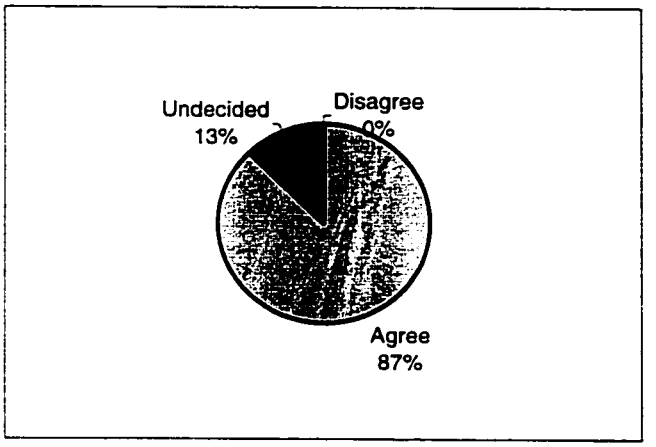
5. I think this system has made me have a headache on occasion.



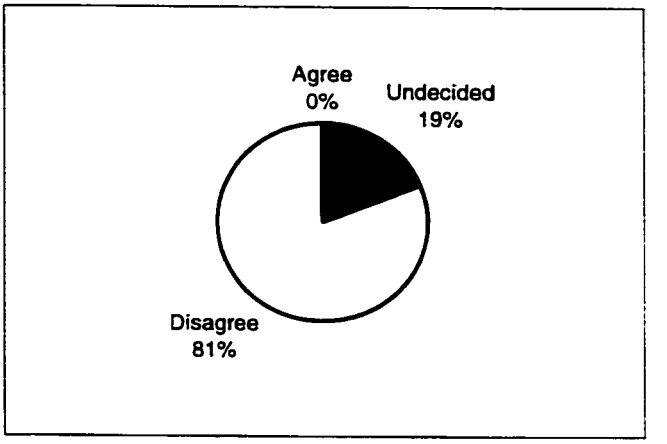
6. Working with this system is mentally stimulating.



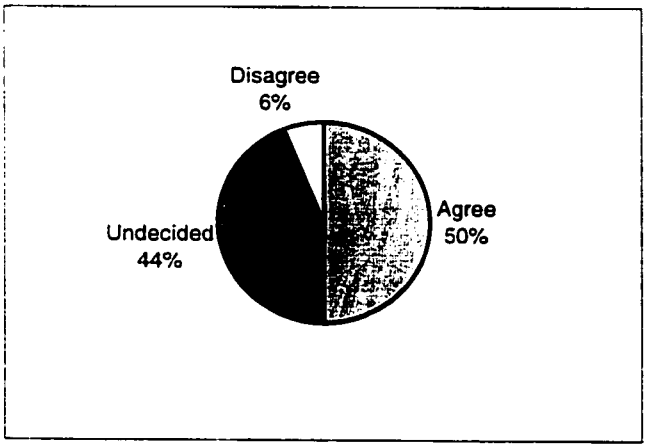
7. I can understand and act on the information provided by this system.



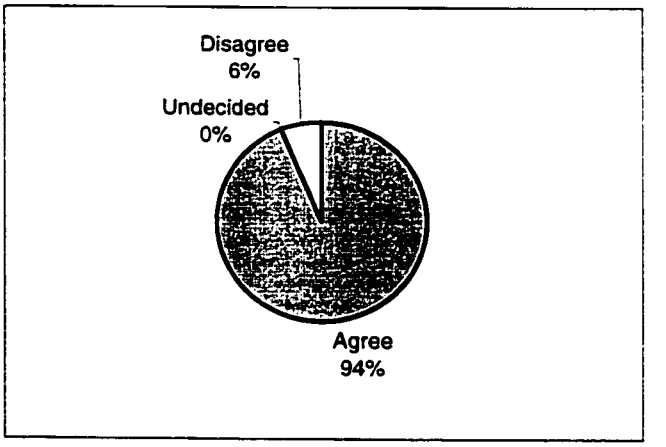
8. There is too much to read before you can use the system.



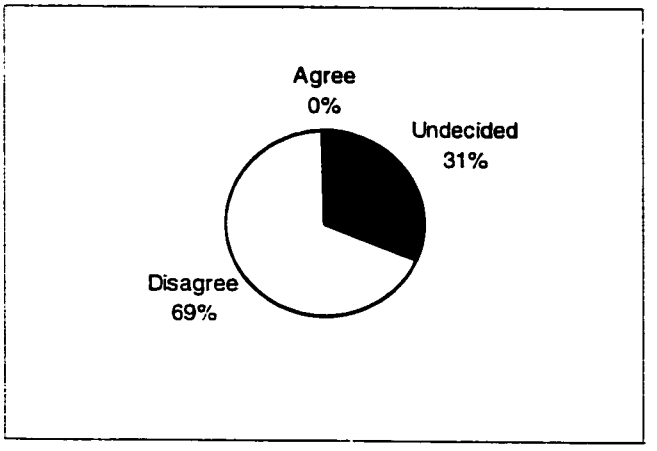
9. The speed of this system is fast enough.



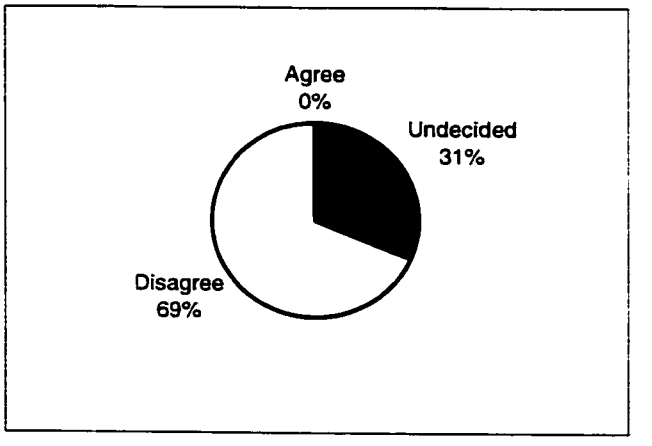
10. I enjoy my sessions with this system.



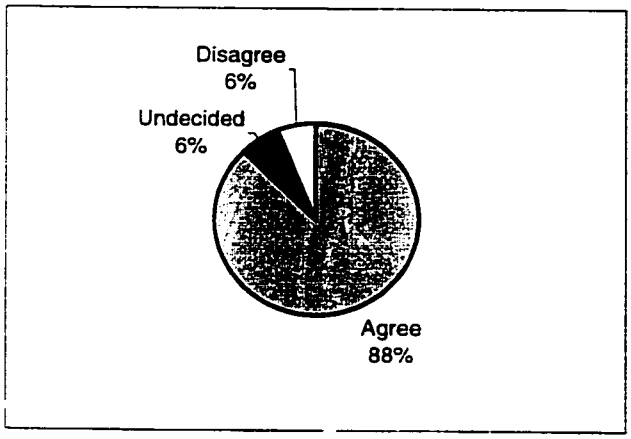
11. This system occasionally behaves in a way which can't be understood.



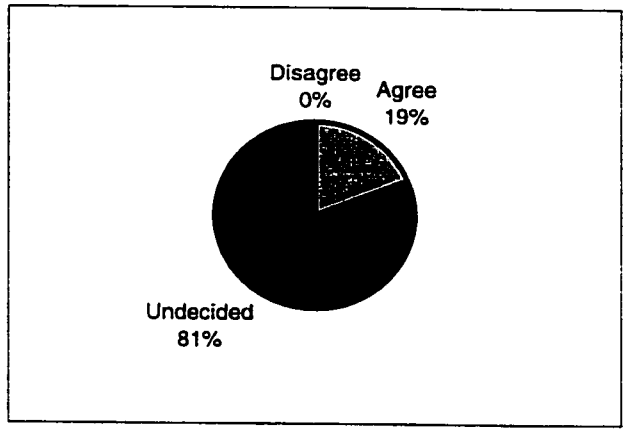
12. I have to look for assistance most times when I use this system.



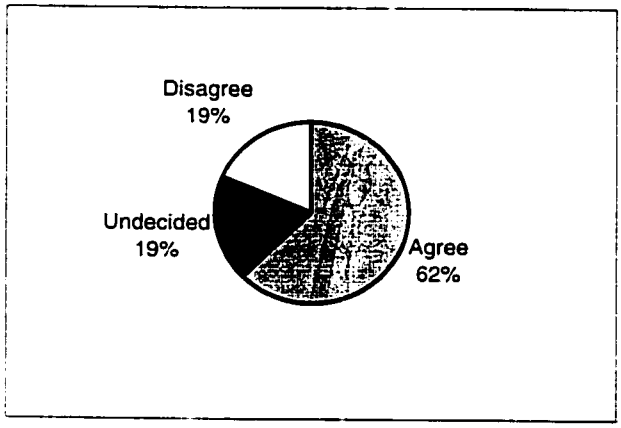
13. The way that system information is presented is clear and understandable.



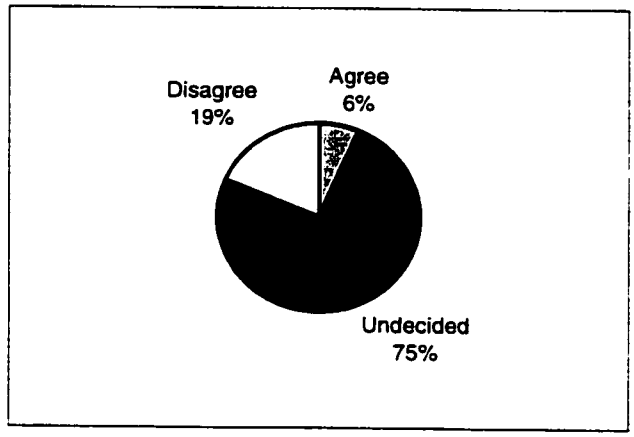
14. Whenever I make a mistake using the system, I recover easily and quickly.



15. The interface of this system is pleasant.



16. Error prevention messages are not adequate.



In the above diagrams, we see the users' ratings, statement by statement. The ratings (pie chart) indicate how much more the users agreed or disagreed with the item as applied to the tested prototype. The evaluation items are in precisely the same wording that they were presented to the users. The degrees of individual sectors indicate how much more the overall users agreed or disagreed with the items. According to the diagnostic data, the system features that influenced the subjective assessment are as follows.

- The system response in real time.
- The system easiness to operate.
- The system stability.
- The system presentation.

In the 1st statement, about 74% of the users felt this system responds fast enough to inputs. In the 9th statement, only 6% of the users thought the speed of the system is not fast enough. These valuation results indicate that this prototype can be operated in real time.

In the 3rd statement, 19% of the users thought the system is easy to restart, with a great majority unable to respond. 100% of the users in the 4th statement felt that they do not have any difficulty to operate this system initially. In the 7th statement, about 87% of the users could understand and act on the information provided by this system. No one thought there is too much to read before operating the system. In statement 12th, 69% of the users didn't need assistance most times when they use the system. About 19% of the users thought they couldn't recover the system easily and quickly when they made a mistake using it. These results demonstrate that this system is easy to operate.

In the whole evaluation process, this system did not stop unexpectedly. 0% users felt that this system occasionally behaves in a way that cannot be understood. From these facts, we can say that, in the subjective assessment, we exhibited to the users a quite stable prototype system.

In the 5th statement, 0% of the users felt a headache made by this system on occasion. In the 10th statement, 94% of the users enjoyed the session with this system. In the 13th statement, 88% of the users thought the way the system information was presented are clear and understandable. Only 19% of the users felt the interface of this system was not pleasant, in the 13th statement. About 19% of the users in the 16th statement thought the error prevention message were adequate.

In conclusion, the system presentation is good. The synthetic images of the virtual world and the real world do not make users feel a headache. The delay between head motion and virtual feedback does not impair the adaptation and illusion significantly. Users can enjoy the system most of the time and can interact with the system easily. The interface of this system is pleasant. However, although the system runs very stable and is easy to operate, once errors occur or the system crushes, it does not provide to the users enough error prevention information. Therefore, it is difficult for the users to recover from the errors or to prevent the errors in advance.

Chapter 6

6. Conclusions and Future Work

6.1 Conclusions

In this thesis, we addressed the problem of identifying a great number of real world objects with a robust marking approach by using computer vision techniques. In the first part, we gave an introduction to the augmented reality technology. Then, we described some existing problems in this research area. In chapter 3, we presented the workflow of our prototype using a new marking approach. After that, a comparison among our solution and other visual marking technologies was made. The binary code error detection and correction functions used in the marker recognition algorithm make the lighting-sensitive, vision-based application more robust. In the rest part of this thesis, we introduced the hardware and software design issue of our prototype. Then, we introduced some potential augmented applications in industry, education and entertainment, all of which are based on the vision-based augmented reality environment we built.

The emphasis of the project had been on testing the feasibility and efficiency of AR user interaction in industrial training applications. The prototype system we developed demonstrated that such an application could be very successful. The system has achieved fairly accurate results while using existing low cost hardware and the object recognition algorithm we proposed. The camera calibration and pose estimation process uses an off the shelf software library.

Although, testing results show that the technology in its present form is ready for the applications, the registration accuracy and the rendering speed can be improved furthermore after the price for higher resolution camera and displays, and more powerful computer graphic device come to an affordable level. We are currently building a more scalable collaborative augmented reality application. The whole system will be running on new hardware with stronger computer graphic capabilities.

In conclusion, AR is a developing and still not completely explored field that can find many applications from mobile application to remote present to collaborative operation to web based applications. The potential commercial values of AR applications will make this research area prosperous in the near future.

6.2 Problems Encountered

Pose tracking

In order to accurately superimpose computer-generated graphics or text prompts upon objects in the real world, tracking the position and orientation of the object is the most significant task to complete. Due to the low processing speed of the wearable computer, a pose tracking algorithm with low complexity and high accuracy was needed.

Marker pattern recognition

In most AR applications, more than one object in the user's view need to be recognized and overlapped with the geometry. How to recognize multiple objects and how to distinguish one from the others are the problems we had to solve.

False tolerant

The vision-based augmented reality system recognizes objects by detecting the markers attached on the objects. The probability that the pattern of the marker can be recognized correctly depends on many factors.

- The visible range for the camera to detect the marker
- The marker orientation relative to the camera
- The camera resolution
- The materials the marker is made from
- The lighting conditions of the environment

When the pattern cannot be recognized correctly, the real world object will not be overlapped with computer-generated graphics. How to detect and correct the errors was another problem we met.

Augmented context information management

In AR applications, all kinds of computer-generated information can be augmented to the real world. Video stream, audio, image, 3D graphic image and text can be merged to the real world. Every specific object in the real world has certain computer-generated context information to augment. We had to manage this augmented information in certain efficient way.

Augmented reality user interface

The user interface design is another area that needs a lot of work. One purpose of our project is to design an intuitive user interface through which the user can interact and manipulate the augmented information efficiently. For the AR applications, the user interface has two tasks to accomplish. One is the interaction between the user and the real world object. Since in AR systems the user see the real world and often desires to interact with real objects, it is appropriate for the AR interface to have a real component (the marker) instead of remaining entirely virtual. The other is interaction between the user and the virtual object. Our implementation is based on the used designs from well-developed Virtual Environment system. Moreover, the way a user would input data and the way information from the database would be presented to the user are very critical to the success of the applications.

6.3 Future Work

In the new AR applications, such as outdoor or mobile AR, the users could be repairman, technical service provider, visitor, physicians, or workers on the assembly lines. Those people's hands must be free for other work. Assembly line workers, for example, can access a database and look at graphic images of the internal structure of what's being worked on while doing the operations with his hands at the same time. Physicians are using them to access patients' charts and monitoring the patient's blood pressure, while doing the surgery. The wearable computer is a very good equipment to meet the special requirement of those mobile AR applications. The wearable computer is composed of a HMD and a light weighted small PC being worn in a belt around the waist. Commands are either keyed into a small keyboard worn on the wrist or spoken into a microphone. The body of the user wearing the computer can move freely in the work place, his hands are free to do other works besides operating the computer. The special requirements of the AR applications are also the main motivation of the development of wearable computing technology. Most wearable computer products in the market today have limited CPU processing power, and the graphic capability is very weak.

The software performance of our AR prototype is partially limited by the hardware. To meet the need of 10fps image generation rate, and to exploit the mobility of the wearable computer and HMD, we will move most of the workload to a remote workstation and leave only the image compression/decompression, display and data transmission to the wearable computer. The wearable computer (Xybernaut MA IV wearable computer with a 233MHz Pentium MMX) and the remote graphic workstation will be connected by 10Mbps wireless Ethernet. This method will be particularly effective in applications where multiple users collaborate with each other through a wireless-LAN. The remote workstation can work as a server to distribute data to wearable computers.

Later on, to improve the performance of the application running on the NT workstation, we will immigrate the computer graphic task of our system to the SGI IRIX workstation, and add a 3D

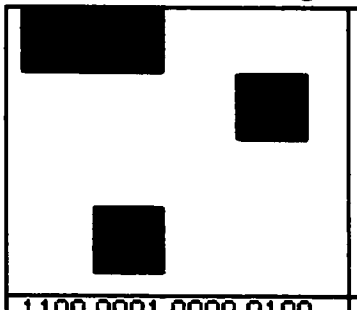
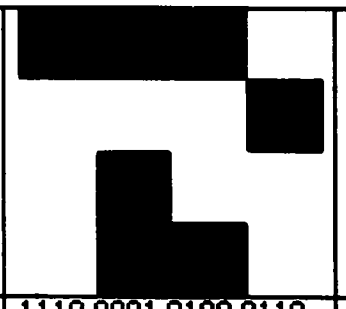
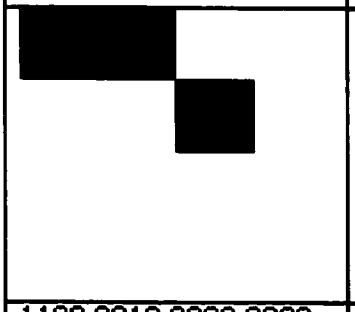
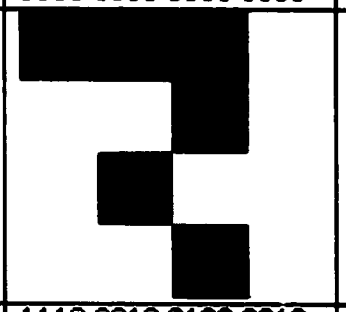
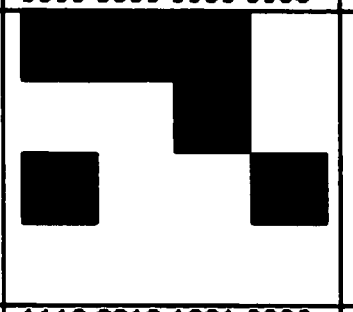
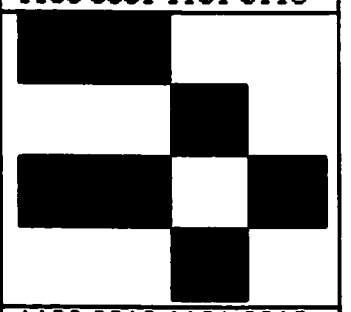
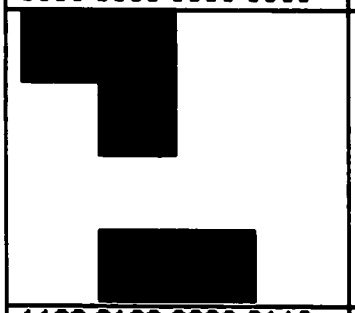
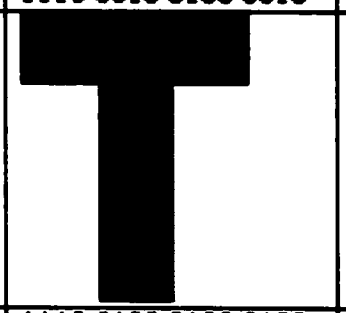
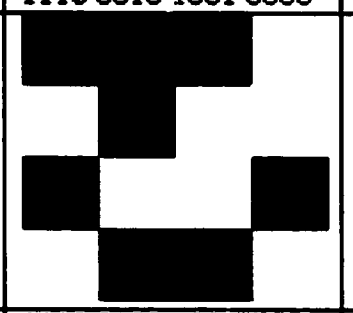
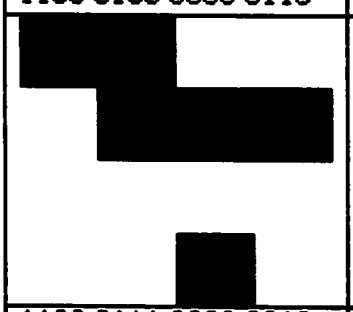
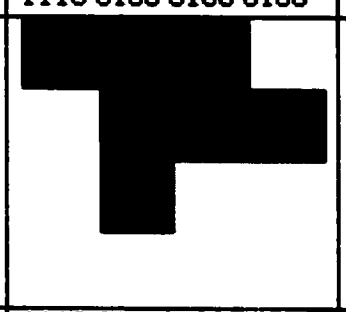
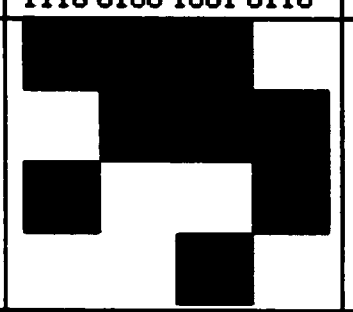
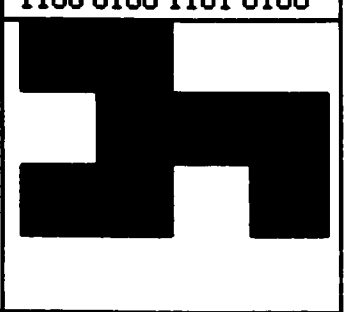
graphic card into the wearable computer to increase the rendering speed and off the workload of the central server in collaborative applications.

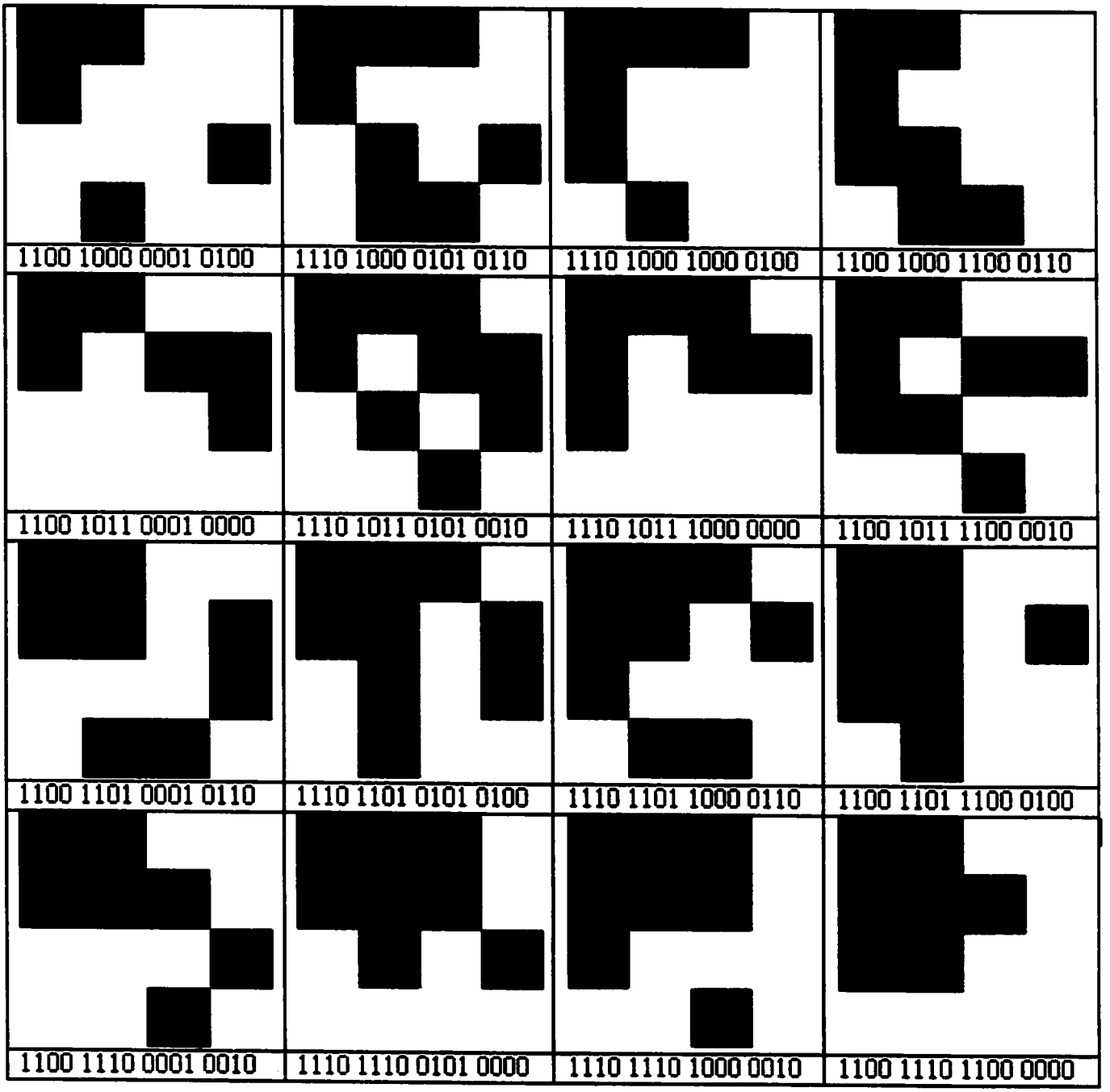
We will implement the proposed augmented reality applications based on our existing prototypes. The user interface design for the software can be improved in the way a user controls the motion and some properties of the augmented 3D model, and in how the user manipulates the database easily and on the spot.

Although, AR technology has been studied for more than a decade, there are still a lot of works to do to optimize the object recognition and registration algorithms. AR applications over the internet is a new research topic yet to be explored. A few such applications could be internet training or shopping online by rendering 3D models over the video images on the remote site.

Appendix

32 4x4 Binary Square Marker patterns and corresponding 16bit marker codes with block sum checking function.

			
1100 0001 0000 0100	1110 0001 0100 0110	1110 0001 1001 0100	1100 0001 1101 0110
			
1100 0010 0000 0000	1110 0010 0100 0010	1110 0010 1001 0000	1100 0010 1101 0010
			
1100 0100 0000 0110	1110 0100 0100 0100	1110 0100 1001 0110	1100 0100 1101 0100
			
1100 0111 0000 0010	1110 0111 0100 0000	1110 0111 1001 0010	1100 0111 1101 0000


1100 1000 0001 0100
1110 1000 0101 0110
1110 1000 1000 0100
1100 1000 1100 0110
1100 1011 0001 0000
1110 1011 0101 0010
1110 1011 1000 0000
1100 1011 1100 0010
1100 1101 0001 0110
1110 1101 0101 0100
1110 1101 1000 0110
1100 1101 1100 0100
1100 1110 0001 0010
1110 1110 0101 0000
1110 1110 1000 0010
1100 1110 1100 0000

Bibliography

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

[2] R.Azuma, Y.Baillot, R.Behringer, S.Feiner, S.Julier, B.MacIntyre, *Recent Advances in Augmented Reality.* IEEE Computer Graphics and Applications 21, 6 (Nov/Dec 2001), pp.34-47.

[3] S. Mann and J. Fung, *Diminished Reality research.* http://wearcam.org/diminished_reality.htm

[4] R.T. Azuma, *SIGGRAPH '95 Course Notes: A Survey of Augmented Reality.* Los Angeles, Association for Computing Machinery, pp.1-38, 1995.

[5] Mixed Reality Systems Laboratory Inc. http://www.mr-system.co.jp/index_e.shtml

[6] Project ARVIKA in Germany. http://www.arvika.de/www/e/home/home.htm

[7] *Shared Space: Collaborative Augmented Reality*, the HITLab at the University of Washington. http://www.hitl.washington.edu/research/shared_space

[8] MIT Media Lab, Wearable Computing. http://www.media.mit.edu/wearables/

[9] Sony Computer Science Laboratory. http://www.csl.sony.co.jp/

[10] N.I. Durlach, *Psychophysical Considerations in the Design of Human-Machine Interfaces for Teleoperator and Virtual-Environment Systems.* In Medicine Meets Virtual Reality II: Interactive Technology & Healthcare: Visionary Applications for Simulation Visualization Robotics (pp. 45-47). San Diego, CA, USA: Aligned Management Associates, 1994.

[11] Instructional Systems - Augmented reality. http://www.boeing.com/defense-space/military/aerosupport/instruct/augmented.htm

[12] Bell Canada, IBM Canada and Xybernaut(R) Launch World's First Large-Scale Market Trial of Wearable computers.
http://www2.software.ibm.com/casestudies/swcs.nsf/customername

[13] Xybernaut(R) Mobile Assistant(R),
http://www.mobileinfo.com/News_2001/Issue32/Bell_Canada.htm.

[14] W. Hoff, K. Nguyen, and T. Lyon. *Computer vision based registration techniques for augmented reality.* In Proceedings of Intelligent Robots and Computer Vision XV, SPIE Vol.2904, Nov 18-22, 1996, Boston, MA, pp. 538-548, 1996.

[15] Y. Cho, J. Lee, and U. Neumann, *A Multi-ring Color Fiducial System and An Intensity-invariant Detection Method for Scalable Fiducial-Tracking Augmented Reality.* In IWAR98, pp.147-165, 1998.

[16] A. State, G. Hirota, D.T. Chen, B. Garrett, and M. Livingston, *Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking.* SIGGRAPH 1996, New Orleans, LA, pp.429-438, 1996.

[17] J. Rekimoto, *Matrix: A Realtime Object Identification and Registration Method for Augmented Reality.* In Proceedings of Asia Pacific Computer Human Interaction (APCHI'98), Hayama-machi, Japan, pp.63-69, 1998.

[18] K.N. Kutulakos and J.R. Vallino, *Calibration-Free Augmented Reality.* In IEEE Transactions on visualization and computer graphics, VOL.4, NO.1, pp.1-20, JANUARY-MARCH 1998

[19] J. Rekimoto, and Y. Ayatsuka, *CyberCode: Designing Augmented Reality Environments with Visual Tags,* In proceedings of Designing Augmented Reality Environments (DARE 2000), Elsinore, Denmark, pp.1-10, April, 2000.

[20] M A. Livingston, *Vision-based tracking with dynamic structured light for video see-through augmented reality.* Ph.D. thesis, Department of Computer Science, University of North Carolina, Chapel Hill, 1998.

[21] S.Ditlea, *The PC goes ready-to-wear.* In IEEE Spectrum, VOL.37, NO.10, October 2000.

[22] Camera model and multiple view geometry
http://www.esat.kuleuven.ac.be/~pollefey/tutorial/node35.html

[23] H.Kato, M.Billinghurst, *Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System.* In Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, San Francisco, pp.85-94, October 1999.

[24] I. Herman, *The Use of Projective Geometry in Computer Graphics.* Lecture Notes in Computer Sciences No. 564, eds. G. Goos and J. Hartmanis, Springer Verlag (1992).

[25] M.Lewin, R.Bowden, and M.Sarhadi, Automotive Prototyping Using Augmented Reality. Vision and VR Group, Brunel University. http://www.nottingham.ac.uk/~enzrh/VRSIG7Proc/Lewin/Lewin.html

[26] M. Jethwa, A. Zisserman, and A. Fitzgibbon, *Real-time Panoramic Mosaics and Augmented Reality.* In Proceedings of Britsh Machine Vision Conference, Southampton, pp 852-862, 1998.

[27] J L. Mundy, A. Zisseman, D. Forsyth, *Applications of invariance in computer vision.* Lecture notes in computer science, No. 825, Berlin, New York, Springer-Verlag, c1994.

[28] Y.Hung, P.Yeh, and D.Harwood, *Passive Ranging to Known Planar Point Sets.* In Proceedings of IEEE International Conference on Robotics and Automation, Vol.1, St.Louis, Missouri, 25-28 March, pp.80-85, 1985.

[29] M. Billinghurst and H. Kato, *Collaborative Mixed Reality.* In Proceedings of International Symposium on Mixed Reality (ISMR '99), Mixed Reality--Merging Real and Virtual Worlds, pp. 261-284, 1999.

[30] A. Webster, S. Feiner, B. MacIntyre, M. Massie, and T. Krueger, *Augmented Reality in Architectural Construction, Inspection, and Renovation.* In Proceedings of ASCE Computing in C.E., pp. 913-919, 1996.

[31] Xybernaut Mobile Assistant IV. http://www.twomobile.com/rev_xybernaut.html

[32] Software Usability Measurement Inventory (SUMI). http://sumi.ucc.ie/

[33] A. D. Marshall, *Vision Systems.* http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/MARSHALL/Vision_lecture_caller.html

[34] A. Tripathi, *Augmented Reality: An Application for Architecture.* Master's Thesis, University of Southern California, 2001.

[35] Camera Calibration. http://kwon3d.com/theories.html

[36] Camera Calibration http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT9/lect9.html

[37] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan, *Real-time Vision-Based Camera Tracking for Augmented Reality Applications.* In Proceedings of the Symposium on Virtual Reality Software and Technology (VRST-97), Lausanne, Switzerland, September 15-17, pp. 87-94, 1997.

[38] D.J. Haniff, C. Baber and W.H. Edmondson, *Categorizing Augmented Reality Systems.* In International Conference on Human and Computer (HC-2000), pp.229-234, Japan, 2000.

[39] E. Sharlin, P. Figueroa, M. Green and B. Watson, *A Wireless, Inexpensive Optical Tracker for the CAVE™.* In Proceedings of IEEE Virtual Reality 2000 Conference, New Brunswick, NJ, March 18-22, pp. 271-278, 2000.

[40] G. Reitmayr, D. Schmalstieg, *Mobile Collaborative Augmented Reality.* In proceedings of ISAR '01, the Second IEEE and ACM International Symposium on Augmented Reality, New York, October 2001.

[41] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, *Virtual Object Manipulation on a Table-Top AR Environment.* In Proceedings of ISAR 2000, Oct 5th-6th, pp.111-119, 2000.

[42] H.Tamura, and H.Yamamoto, *Vision and Graphics in Producing Mixed Reality Worlds.* In Proceedings of CVVRHC'98, pp.78-85, 1998.

[43] Intel JPEG Library.
http://www.intel.com/software/products/perflib/ijl/

[44] J. Vallino, *Augmenting Reality with Minimal Calibration*
http://www.cs.rit.edu/~jrv/research/ar/thesis.html

[45] M. Billinghurst, H. Kato, and I. Poupyrev *The MagicBook—Moving Seamlessly between Reality and Virtuality.* IEEE Computer Graphics and Applications, May/June 2001 (Vol. 21, No. 3), pp. 6-8.

[46] Microsoft Research, Vision SDK.
http://research.microsoft.com/projects/VisSDK/

[47] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. *Augmented Reality: A Class of Displays on the Reality-Virutality Continuum.* SPIE Proceedings: Telemanipulator and Telepresence Technologies . H. Das, SPIE Vol. 2351, pp. 282-292, 1994.

[48] O. Faugeras, *Three-Dimensional Computer Vision A Geometric Viewpoint.* Cambridge, MA: MIT Press, 1993.

[49] P.Liu, N.D. Georganas, and P. Boulanger, *Designing real-time vision based augmented reality environments for 3D collaborative applications.* In Proceedings of IEEE Canadian Conference on Electrical & Computer Engineering, Winnipeg, pp.715-720, 2002.

[50] S. Birchfield, *An Introduction to Projective Geometry (for computer vision).* http://robotics.stanford.edu/~birch/projective/

[51] X. Zhong, P. Liu, N.D. Georganas, and P. Boulanger, *Designing a Vision Based Collaborative Augmented Reality Application for Industrial Training,* subm. to ACM Multimedia 2002, Juan LePins, France, Dec. 2002

[52] Z. Zhang, *A Flexible New Technique for Camera Calibratino,* Microsoft technical report. http://research.microsoft.com/~zhang/Calib/