Framework for Indoor Video-based Augmented Reality Applications

Bechir Khabir

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science
(Electrical and Computer Engineering)
at
Concordia University
Montreal, Quebec, Canada

August 2008

Your file Votre référence
ISBN: 978-0-494-45307-0
Our file Notre référence
ISBN: 978-0-494-45307-0

ABSTRACT

Framework for Indoor Video-based Augmented Reality Applications

Bechir Khabir

Augmented Reality (AR) has been proven to be useful in many fields such as medical surgery, military training, engineering design, tourist guiding, manufacturing and maintenance. Several AR systems and tracking tools have been reviewed and examined. Taking into consideration the different shortcomings of the available AR systems, a framework for indoor video-based AR applications is proposed to integrate four main components of AR applications, which are large scale virtual environment, mobile devices, interaction methods and video-tracking, in one system. The proposed framework benefits from the rapidly evolving technology in virtual modeling by combing GIS maps and 3D virtual models of cities and building interiors in one single platform. Interaction methods for AR applications are introduced, such as the automatic 3D picking which allows for a location-based data access. In addition, a practical method is proposed for the configuration and the deployment of video tracking. This method makes use of the XML mark-up language to allow for future extensions and simplified interchangeability. An implementation of the proposed approach is developed to demonstrate the feasibility of the framework. Different case studies are carried out to validate the applicability of the system and identify its benefits and limitations.

ACKOWNLEGEMENTS

It is indeed a great pleasure for me to express my sincere appreciation and thanks to my respectable supervisor Dr. Amin Hammad who gave me the opportunity to pursue my Master's degree and provided the initial concept for this research project. His profound knowledge and thoughtful instructions have always shed light on my way to pursue this thesis work.

I would like to express my special gratitude to my dear mother Mabrouka, my dear father Saïd and my wife Kaouthar for having been always there for me by encouraging and caring for my aspirations. To them, I dedicate this thesis.

During these demanding years of study, the support of our Infra-group was of extreme importance. I would like to thank Cheng Zhang, Elaheh Muzaffari, Sara Fadaee, Khaled El-Emmari, Prasana Devarakonda, Yongxin Hu and Jian Jiang for their cooperation and contribution in the development of the 3D model and the database. In addition, the help of all other friends in this project is appreciated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

Abbreviation	**Description**
2D	Two-dimensional
3D	Three-dimensional
AR	Augmented Reality
DB	Database
GIS	Geographic Information System
GPS	Global Positioning System
GUI	Graphical User Interface
HMD	Head-Mounted Display
I/O	Input/Output
JDBC	Java Database Connectivity
LBC	Location-Based Computing
MR	Mixed Reality
VE	Virtual Environment
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
HVAC	Heating, Ventilation and Air Conditioning

CHAPTER 1 INTRODUCTION

1.1 GENERAL BACKGROUND

Augmented Reality (AR) allows interaction with 3D virtual objects and other types of information superimposed over real world objects in real time (Azuma, 1997; Azuma et al., 2001). The benefits of AR have been discussed in many engineering applications including design perception (Dunston et al., 2002, see Chapter 2). The augmentation can be realized by looking at the real world using a see-through Head-Mounted Display (HMD) equipped with sensors that accurately track head movements (3 displacements and 3 rotations) to register the virtual objects with the real objects in real time. The augmentation can be a representation of both physical and non-physical objects. Non-physical objects can be navigation aids such as paths and navigation arrows. Physical objects can be existing in the real world but hidden. In this case, the augmentation can help the user perceive virtual models of hidden physicals objects or their attributes based on the task context. In addition, symbolic representations of objects that are difficult to see can be added, such as defects, as will be explained in the Case Study 2 in Chapter 4. On the other hand, physical objects can be non-existing in the real world such as future plans for new buildings that could be shown as augmentation to help urban planners visualize these plans, as will be discussed in Case Study 3 in Chapter 4.

As opposed to Virtual Reality (VR) applications that allow the user to navigate, using input devices, in many different motions and directions such as flying, walking and orbiting, navigation in AR applications is tightly associated with the user's physical movement. In AR environments, the user's movements are continuously tracked and the virtual augmentation is updated according to his/her current location and orientation. Using AR, users do not have to look back and forth at the real world and the computer screen to mentally achieve the spatial mapping of the information displayed on the screen. This helps them better focus on their actual tasks and improves their efficiency and safety. The main challenge of AR is the requirement for accurate 3D spatial databases and head tracking. A broadening range of AR applications have been demonstrated in many fields such as medical surgery aid, aerospace maintenance guide, military training, engineering design, etc. (See Chapter 2).

1.2 RESEARCH OBJECTIVES

This research investigates a framework for indoor video-based AR applications. The framework includes a practical and simplified method for configuring video-based tracking in indoor AR applications. The research has the following main objectives:

(1) To investigate a framework for indoor video-based AR applications.

(2) To investigate a practical method for the configuration of video tracking in AR applications.

(3) To investigate the deployment of large scale Virtual Environments (VEs) in AR applications.

1.3 THESIS ORGANIZATION

This study will be presented as follows:

Chapter 2 Literature Review: This chapter presents the current situation of AR and its applications in different fields. Different tracking technologies for AR are presented as well as different tracking systems and tools. In addition, interaction and navigation methods in AR systems are summarized.

Chapter 3 Framework for indoor video-based AR applications: In this chapter, a generic framework for large scale indoor vide-based AR applications is introduced as well as a practical method for the configuration and deployment of video tracking for AR applications.

Chapter 4 Implementation and Case Studies: In this chapter, several case studies are used to demonstrate the prototype system using the proposed approaches.

Chapter 5 Summary, Conclusions, and Future work: This chapter summarizes the present research work, highlights its contributions, and suggests recommendations for future research.

CHAPTER 2 LITERATURE REVIEW

2.1 INTRODUCTION

VR has been described as a visualization technology that completely substitutes the user's view of the real world with a computer generated graphics that represent a view of a synthetic virtual world to allow him/her to interact in more natural ways with a world that can be, due to its huge size, out of reach (Livingston, 1998). On the other hand, instead of completely replacing the user's view, AR systems, keep the original real world view and add to it virtual objects that are merged in such a way that they appear consistently aligned with the real world and behave the same way as their real counterpart. The main hardware components of AR systems are a display subsystem through which the user sees the virtual and the real world, an image generation subsystem that renders the proper image onto the display subsystem, and a tracking subsystem that detects the user's location and orientation in real time.

2.2 MIXED REALITY (MR)

Mixed Reality (MR) was introduced by Milgram and Kishino (1994) to depict the different combinations of the virtual and real components through a virtuality continuum (Figure 2.1). At the two extremes of the continuum are the real world (Figure 2.1(a)) and the totally VE (Figure 2.1(d)). In the middle region lies the MR. Near the real world end is the AR (Figure 2.1(b) where the perception of the real world is augmented by computer generated data.

This technology combines the viewing of the real-world or video-based environments with superimposed 3D virtual objects that can be manipulated by the viewer. Thus, AR supplements, rather than replaces, the user's real world (Virtual Reality Laboratory website, 2006). The most recent advancement in AR is a wearable system in which users wear a backpack with a portable computer, see-through HMD, and headphones with motion trackers to place and manipulate virtual objects as they move within their real world (Halden Virtual Reality Center, 2006). The other variation of MR is the Augmented Virtuality (AV) (Figure 2.1(c)) which is a term created by Milgram and Kishino (1994) to identify systems which are mostly synthetic with some real world imagery added, such as texture mapping video, onto virtual objects.

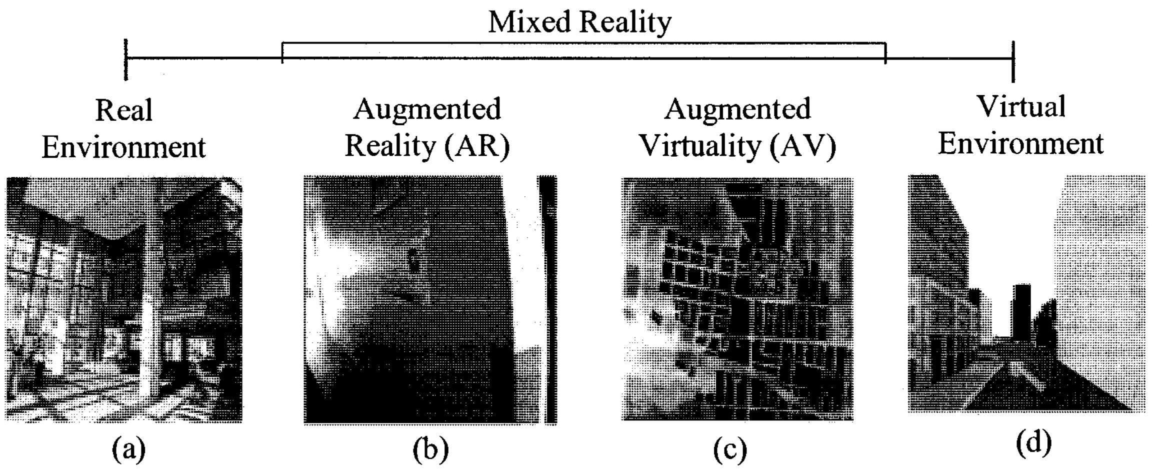


Figure 2.1: MR continuum (adapted from Milgram and Kishino, 1994)

2.3 AUGMENTED REALITY (AR)

AR systems aim to enhance the user's perception of the real world and the interaction with it by augmenting it with 3D virtual objects that are rendered to co-exist in the same space. To create the illusion of seeing both the real world objects and the virtual objects in the same environment, in other words to blend them together, they need to be accurately aligned; this alignment is called registration (Azuma et al., 2001). Azuma (1997) defines AR systems as having the following properties: run interactively and in real time, combine real and virtual objects in a real environment, and register them with each other. Theoretically, the graphics of the virtual world would flawlessly integrate into the real world but, practically, due to many possible imperfection factors, registration errors can occur causing jittering and instability in the rendered graphics.

AR systems can be classified into two categories: optical-based technologies (Figure 2.2) and video-based technologies (Figure 2.3). In optical-based systems, a Head Mounted Display (HMD), equipped

with see-through lenses, is used to see the real world combined with computer generated virtual objects. The virtual objects are superimposed on the real world by means of combiner lenses placed in front of the eyes allowing the user to look directly into the real world. On the other hand, in the video-based systems, a closed view HMD is used to allow the user to see a video of the real world, captured by a video camera, blended with the virtual objects and displayed on opaque monitors placed in front of the eyes.

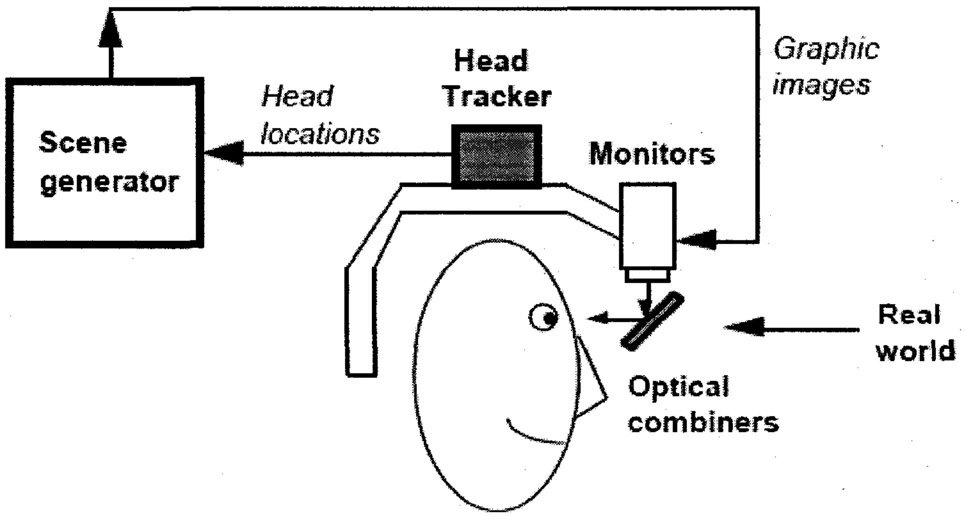


Figure 2.2: Schematic of an optical see-through HMD AR system (Azuma, 1997)

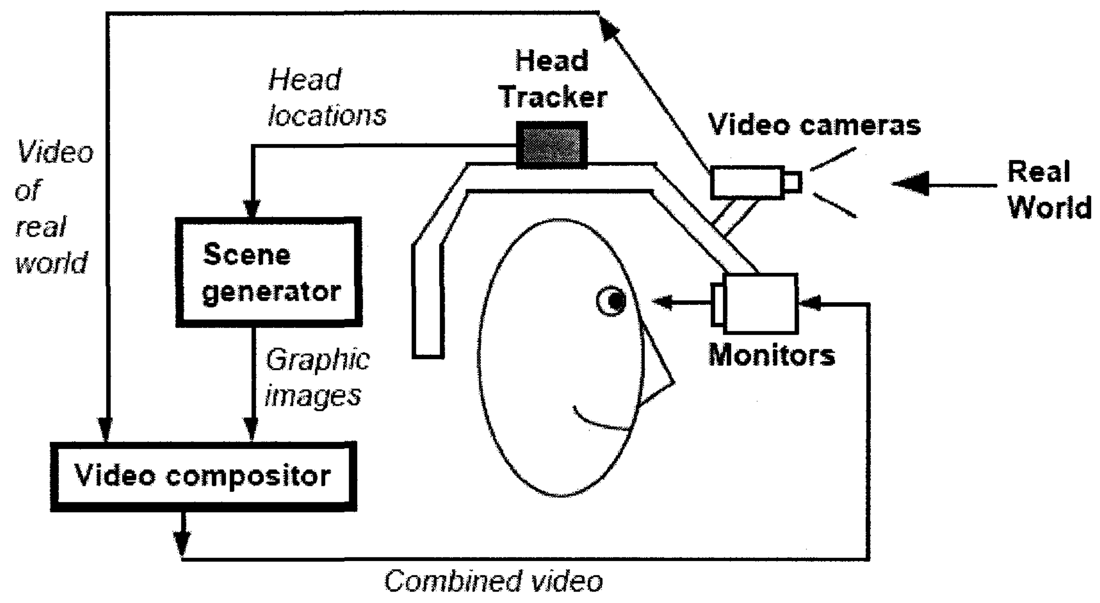


Figure 2.3: Schematic of a video-based AR system (Azuma, 1997)

The optical see-through HMD, when compared with the video see-through HMD (Azuma, 1997), has the advantage of not limiting the real world resolution by allowing the user to directly perceive the real surrounding environment and also has the advantage of being safer in case of sudden electricity cut off; but, the virtual objects for augmentation cannot completely occlude the real world since they always appear semi-transparent. On the other hand, the video see-through HMD has the advantage of flexibility in merging the real world with the virtual objects and thus allows for a realistic occlusion. In addition, the video see-through HMD allows for easier match of brightness of the real and the virtual objects since the brightness of both the video frames and the VE can be adjusted by the user. Also, in case of video see-through AR, the video frames can be used as an additional source for the user location tracking as opposed to the optical see-through that has to rely on other tracking technologies.

2.4 WHY AR?

AR has been proven to make the tasks easier to perform in many fields (Brian, 2004), such as manufacturing, building and civil engineering and inspection, by augmenting the human senses with information that cannot be directly detected by the user's own senses. Also, AR can reduce attention switching between the virtual media and the real world which helps in eliminating short term memory demands. Reducing attention switching can also help in improving the user's safety by keeping his/her focus on the virtual model without losing focus on the real world. On the other hand, AR can direct the user's attention by providing task information that is chronologically organized according to the user's task sequence. By providing the user with only the relevant information to the task at the right time, the user can focus on this task more easily. Through AR, information becomes more specific, efficient, timely and accurate (Chung, 2002).

In addition, AR systems, compared to VR systems, are characterized by requiring much less computational resources to compute synthetic images to display on real images since only the images for augmentation need to be computed. Also, AR systems provide more realistic view of the environment, through the real images, which keeps the user connected with the real world (Romao et al., 2004).

2.5 APPLICATIONS OF AR

(1) Medical

One of the medical applications of the AR is during a surgery. Before carrying out a surgery, data resulting from Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scan is gathered to generate a 3D model of an affected area in the human body; then, the model is registered with the patient and displayed right on his/her body during a surgery which helps the surgical team to find the path to the affected area.

One other medical application of AR is during the ultrasound scanning; the ultrasound images are used to create a volumetric representation of the fetus that is displayed on the abdomen of the pregnant woman in real time (Azuma, 1997); this helps the doctor to get a more realistic view of the fetus during the different pregnancy stages.

(2) Manufacturing, Maintenance and Repair

During the performance of aerospace maintenance tasks, AR can help in information processing by controlling attention and by supporting short and long term memory through immediate access to information. Instead of referring to several repair manuals, the maintenance technician could use an AR display to see only the information pertinent to the repair. In this display, the real equipment would be augmented with annotations and step-by-step instructions. For example, the location of a hardware that must be removed could be highlighted (Vallino, 1998).

Boeing researchers have been developing an AR display to replace the large work frames used for making wiring harnesses for their aircraft (Vallino, 1998). Using this experimental system, the technicians are guided by the augmented display that shows the routing of the cables on a generic frame used for all harnesses. The augmented display allows a single fixture to be used for making the multiple harnesses.

(3) Museum Exhibitions and Tourist Guiding

AR has been used in a cultural heritage application (Figure 2.4) to provide museum visitors with an AR interface to visualize cultural artifacts (Liarokapis et al., 2004) The most important advantage of such application is the capacity to exhibit a great number of artifacts in a limited space, in this case a table-top environment.

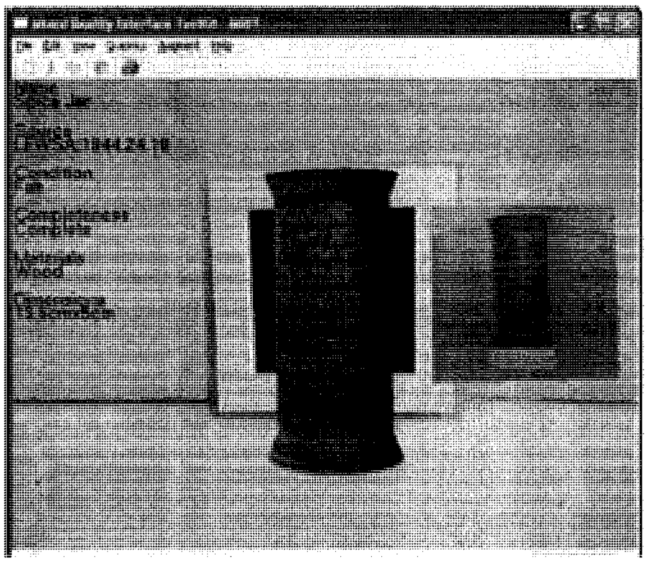


Figure 2.4: Spice jar in an AR museum application (Liarokapis et al., 2004)

(4) Entertainment

AR Pong (Figure 2.5), Cannon Fodder (Figure 2.6), Sphere of Influence and others are prototypes of AR games that have been developed by ARPE group (2003); the two main components of the games are an AR Gaming Table defined using ARToolkit (Kato et al., 1999) paper markers and a rig composed of a webcam, a projector, and a high-end consumer-level computer. In the AR Pong, two players physically interact with the game using two paddle glyphs (paper markers) to move their paddles and two other glyphs are used to calibrate the gaming table to adjust the virtual ball into the real physical environment. In the Cannon Fodder game, the players directly and physically control the VE using virtual cannon with which to shoot attacking goblins. The goblins pass between two cannons attempting to get to a castle. The game ends when a certain number of goblins reach the castle.

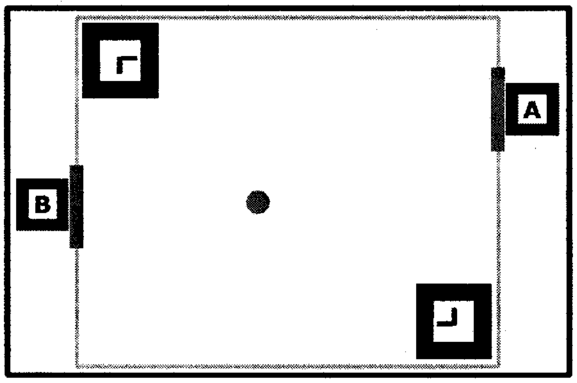


Figure 2.5: AR Pong marker setup (ARPE, 2003)

Figure 2.6: Cannon Fodder AR game (ARPE, 2003)

Another example of AR application in entertainment is during a baseball game broadcasting. The system places an advertisement in the broadcasted image so that it appears on the outfield wall of the stadium. To calibrate the electronic billboard to the stadium, images are taken from typical camera angles and from zoom settings to build a map of the stadium to locate the spots where advertisements will be inserted. Pre-specified reference-points in the stadium are used to allow the system to automatically determine the camera angle being used and refer to the pre-defined stadium map to insert the advertisement in the correct spot.

(5) Military Training

One of the applications of AR in military is a Battlefield AR System (Livingston et al., 2002) prototype that has been developed for military operations in urban terrains. The system consists of a wearable computer, a wireless network system, and a tracked see-through HMD. The system is aimed to assist the war-fighter by improving the situational awareness for effective military operations in urban environment. A 3D model of objects in the real environment, that is used to generate the registered graphical overlay, is stored in a shared database available for all users. This database contains information about the objects, e.g., threat classification and general description, etc. Information about the objects' relevance to each other and to the user's task is also stored in the shared database. This system can be used to facilitate collaboration between different mobile users as well as between them and a command center through the automatic information distribution as shown in Figure 2.7 where the a user location is highlighted and made available to all remote users.

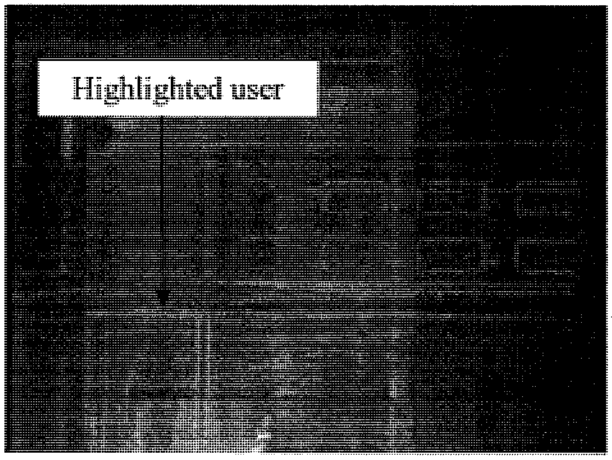


Figure 2.7: A remote user highlighted by a wireframe box (Livingston et al., 2002)

(6) Engineering Design

A prototype system has been proposed (Wang et al., 2004) for design review collaboration in a MR environment: MRCVE (Mixed Reality-based Collaborative Virtual Environments). The system has been aimed to help ensure the quality of designs by amplifying the effectiveness of the design review team. Two scenarios for the system have been proposed: the first scenario consists of a MR-based collaborative design face-to-face conferencing where the 3D designs appear in space among the designers, reviewers and customers. The designs can be modified by some users who are equipped with 3D CAD functionalities. The second scenario consists of an MR-based collaborative virtual space conferencing where a workspace can be setup to allow 3D CAD models to be uploaded, edited or downloaded through any client computer.

In addition, an AR-based platform has been proposed (Behzadan et al., 2006) to generate a mixed reality view of a real world construction site with virtual construction CAD models in an outdoor environment. The proposed platform uses HMD, Global Positioning System (GPS), orientation tracker and a portable computer. One of the prototypes of the system that has been developed is UM-AR-GPS-ROVER; it has been used to demonstrate a chronological simulation of scheduled construction activities for an erection of a structural steel frame. This prototype (Figure 2.8) allows to, interactively, place 3D CAD models at any selected location in an outdoor environment.

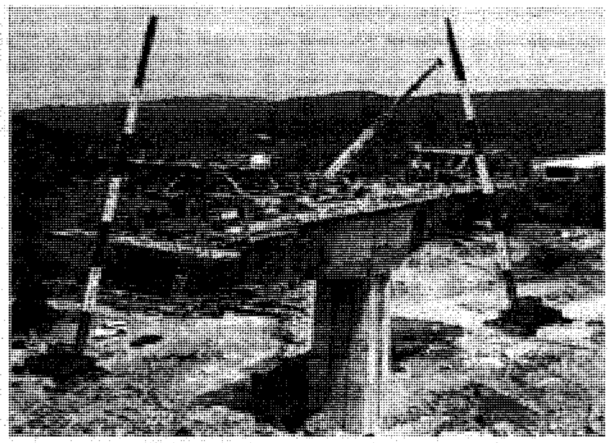


Figure 2.8: AR bridge construction model (Behzadan et al., 2006)

The automobile industry has been, also, one of the fields of application of AR (Frund et al., 2003). An AR-based application has been developed to be used at the design phase of new cars. The application displays virtual car components superimposed on real cars to show design variants or to support design reviews (Figure 2.9). Using this application, the user can select a virtual component from a list of virtual components, presented in a menu, and place them on a real car; the hand gesture is used to interact with the AR scene.

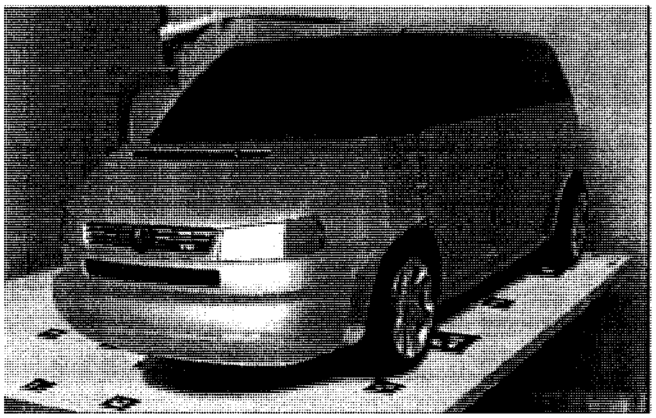


Figure 2.9: Virtual car front superimposed on a real car model (Frund et al., 2003)

(7) Robotics and Telerobotics

In the domain of robotics and telerobotics an augmented display can assist the user of the system. A telerobotic operator uses a visual image of the remote workspace to guide the robot. ARGOS (Milgram et al., 1995) is a tool that has been developed to improve the operator's comprehension of the remote environment and to provide an interactive modeling of the remote world. This tool uses a stereo-graphic cursor as a probing tool that can be positioned anywhere in the stereoscopic video scene. It consists, also, of a virtual manipulator that is a full stereographic 3D model of the remote robot being controlled. The manipulator is then superimposed onto the real robot to be remotely operated within the real 3D work space.

2.6 INTERACTION IN AR ENVIRONMENTS

Slay et al. (2001) separates MR interaction techniques into two categories, exocentric and egocentric metaphors. Exocentric interaction occurs when the user interacts with the scene from outside the VE. Egocentric interaction occurs when the user is imbedded in the VE. There may be far more possible actions that can be performed on objects than such Graphical User Interfaces (GUIs) can realistically provide (Kaiser et al., 2003). Kaiser argues that most prior approaches have placed too much functionality on a too impoverished communications channel (3D arm/hand motions), and that by incorporating multimodal interaction, the burden of various interactive functions can be off-loaded to appropriate modalities, such as speech and gesture, in a synergistic fashion. In particular, by incorporating speech into the interface, the user could describe unseen/unknown objects and locations or invoke functions, while his/her hands and eyes may be engaged in some other tasks. However, unlike direct manipulation interfaces, multimodal interface architectures must cope first and foremost with uncertainty.

In addition to the challenges posed by 3D interaction, AR interaction adds the presence of uncertainty caused by the possibly imperfect knowledge about the components of the system and the low accuracy of the tracking devices. One of the main functionalities in the AR environments affected by uncertainty is the object selection. Kaiser et al. (2003)

proposed multi-modal interaction to disambiguate the object being selected by the user. Kaiser's system integrates speech, head tracking and glove-based finger tracking for gesture recognition to allow the user to select objects using vocal commands, head and hand movement, and 3D virtual volume.

Tracking uncertainty is behind one of the major issues in AR: the registration error. To overcome this issue, Coelho (2005) proposed an uncertainty aware system that deals with uncertainty using registration error estimates. To allow the user to select objects, this system detects the collision between the estimated registration error region of the pointer and the region of each object in the scene.

Interactivity, on the other hand, refers to all types of interaction an application can support. Unlike VR systems, AR can support all kinds of interaction devices as well as tangible interaction mechanisms (Liarokapis, 2006). But, interactivity cannot always be beneficial to the user in the case of high workload conditions (Chung, 2002); in such conditions, a very high interactive system can cause a decrease in the user performance.

2.7 NAVIGATION AND WAYFINDING IN AR ENVIRONMENTS

Navigation is defined as the process of moving in an environment; this definition has been extended to include the process of wayfinding, which is the process of finding one or more routes to a destination in an environment (Liarokapis et al., 2006).

Mobile, wearable and Location-Based Computing (LBC) have emerged in the AR navigation and consequently have posed new challenges related to the unpreparedness for the indoor and outdoor environment.

Unlike VR environments that have to provide full sensory information to the user to be able to carry out a way finding task, AR environments provide just maps or directions to facilitate the task and keep the sensory inputs to the real world. Since the navigation process is cognitive in nature (Brian, 2004), a good representation of spatial knowledge can be significantly beneficial to the user. Spatial knowledge can have three different representations: (1) Landmark knowledge which consists of descriptive information of places in the environment that are clearly distinguishable; (2) Route knowledge which consists of the knowledge of mentally defined routes between different locations; and (3) Survey knowledge which consists of a mental map of the environment (Brian, 2004).

In wayfinding, the user must be able to effectively move in the environment to obtain different views and acquire an accurate ‘‘mental map’’ of his/her surroundings (Mozaffari, 2006); hence, a virtual 2D map can help the user locate him/herself and decide on the next move to get to the destination. Map orientation is, also, significantly important in helping the user during navigation. The orientation of the map can be either egocentric, which means the user's forward orientation is always up, or allocentric, which means the north direction is always up. For environments where the user is directly represented and participating, such AR environments, the literature suggests that the egocentric map orientation is more convenient (Brian, 2004).

Reitmayr et al. (2004) developed a tourist guide AR system that guides the user from his/her current location to a destination through the shortest path. The shortest path is displayed in the form of 3D cylinders, linked by arrows, augmenting the real world scene. The path is dynamically computed based on the user location that is automatically updated while the user is moving. The system allows the user to browse information about the different touristic sites and also allows different concurrent users to collaborate and communicate information. ANTS is another AR navigation system (Romao et al., 2004) that can be used to explore physical structures and natural elements of the surrounding real world for environmental management purposes. The system works on desktop computers and Personal Digital Assistants (PDAs). The ANTS system architecture is made up of three main components: a mobile user AR module that communicates with a 3D model server and a geo-referenced database.

Since, in AR, virtual objects are blended and registered with real world objects, the effect of perceiving depth in the virtual augmentation has been found enhancing for the AR navigation. For example, the virtual objects need to be occluded by real objects that are nearer to the user and need to be lit by the same light sources of the real world (Tatham, 1997) and have consistent shadows. In their assessment of the usefulness of depth cues and the benefits of stereoscopic displays, Sands et al. (2004) concluded that extra sources of depth information (in this case, addition of shadows and coordinates) can significantly increase the accuracy of the selection of targets, rendered on a stereoscopic display, using 3D cursor.

2.8 TRACKING TECHNOLOGIES, TOOLS, AND SYSTEMS

2.8.1 Tracking Technologies

(1) Magnetic Trackers

A magnetic tracker consists of a transmitter and the sensors. The tracking system measures the strength of a set of magnetic fields generated by the transmitter. Four magnetic fields have to be measured: the environmental field, which includes the

Earth's magnetic field, and three orthogonal fields. At each sensor, twelve measurements are taken for each field. All this information is used to compute the position and orientation of the sensor with respect to the transmitter.

Magnetic systems are considered robust, fast and inexpensive compared to other technologies (Livingston, 1998).

(2) Mechanical Trackers

Mechanical trackers are made up of jointed mechanical arms. The orientation and position of each joint of an articulated arm are tracked using rotary transducers (Livingston, 1998).

Mechanical tracking systems are limited to tracking only one object. They have been used widespread for hand tracking as well as tracking all parts of the human body.

The accuracy of the measurements of these systems is considered excellent when compared to the magnetic system. But, on the other hand, they may limit the freedom of movement of the user due to the arms attached to the human body (Brunner et al., 2003).

(3) Acoustic Trackers

Combining a given room temperature and a time of flight of ultrasonic sounds, the acoustic tracking system computes a three degrees-of-freedom position of a transmitter. The system is made up by a transmitter carried by a user and a series of sensors.

Acoustic systems suffer from environmental interference like the temperature variation from possible obstructions between transmitter and receiver. These systems are limited in accuracy and speed.

(4) Inertial Trackers

The inertial trackers measure velocity and acceleration and use the rigid-body kinematics to compute the change in position and orientation based on an initial position. To get a full 6-DOF tracking, this system has to be combined with other tracking system that provides an accurate initial position.

Since the inertial trackers make relative measurements, they accumulate error over time which causes drifting in space. Misalignment with gravity vector is an additional cause of error. A tilt error of one degree in ten seconds can cause about nine meters of position error (Brunner et al., 2003).

(5) Radio Waves Trackers

Radio waves tracking systems measure the time of flight and phase difference of radio waves. An example of these systems is the Global Positioning System (GPS) which provides tracking over the whole surface of the Earth. The GPS uses orbiting satellites to transmit radio waves that are tracked by a receiver. Twenty-four earth-orbiting satellites are used to guarantee at least four of them are available at any time anywhere on the earth. GPS is widely used for mobile mapping and data collection tasks thanks to its availability, low cost and good accuracy.

(6) Optical Trackers

Optical trackers use sensors to detect light and measure angles to determine the user's pose. The target light to be detected by the sensors can be emitted and powered, known as active target, or reflected (not powered), known as passive target. Examples of active targets are the light-emitting diode and the infrared LEDs (ILEDs) (Brunner et al., 2003). Examples of passive targets are reflective materials and high contrast patterns. The targets are detected through simple video cameras or lateral-effect photodiodes. On one hand, optical trackers are considered highly accurate and well suited for real-time systems for their high update rate (Brunner et al., 2003). On the other hand, these trackers require a clear line of sight between the sensor and the target and suffer from obscuration difficulties.

2.8.2 Video Tracking tools and systems

(1) ARToolkit

ARToolkit is one of the most widely used fiducial (also called marker) tracking systems. It was developed in 1999 by Hirokazo Kato and Mark Billinghurst at the Human Interface Technology Lab of the University of Washington. ARToolkit supports full pose calculation and fiducial identification. It is freely distributed as an open source software library written in C and C++ (ARToolkit website, 2005). ARToolkit has been the core of a wide variety of AR systems and applications, that use vision-based tracking, thanks to its ease of use, documentation and free distribution (Middlin, 2002)

ARToolkit allows video tracking of markers using a video camera and computer vision algorithm to calculate the camera position and orientation relative to physical markers in real time. Some of the features of ARToolkit are: single camera position/orientation tracking, the ability to use any square marker patterns, easy camera calibration, and tracking speed suitable for real-time AR applications. The tracking range can be improved by varying the size of the markers (Hammad et al., 2005).

ARToolkit markers are square shaped images surrounded by a predetermined black band. The black band is used to detect candidate images from the captured ones. Then, the interior image is used to find the final candidate. The fiducial relative position and orientation with respect to the camera is computed using its four corners. The marker recognition algorithm (Rekimoto, 1998) used by ARToolkit consists of 5 steps: (1) Binarization: each captured video image is binarized using the adaptive threshold method; (2) Connected component Analysis: the system searches for connected regions of binary-1 (black) pixels. Then, for each region a heuristic checking is done to select code candidate areas; (3) Code frame fitting: for each region a quad-tangle is fitted on the frame of the region using the least-square method. Then, transformation parameters are computed based on the four corners of this quad-tangle; (4) Decoding and error checking: a corresponding image is projected on the code rectangle space then a Cyclic Redundancy Check (CRC) is done to get the recognized code ID; and (5) Camera position and pose estimation: the recognized code frame is used to estimate the camera pose. ARToolkit allows using multi markers at the same time (Hammad et al., 2005).

While there is no systematic process for designing ARToolkit fiducials, some characteristics are recommended to get a reliable and acceptable tracking (see Section 3.3.2 of Chapter 3).

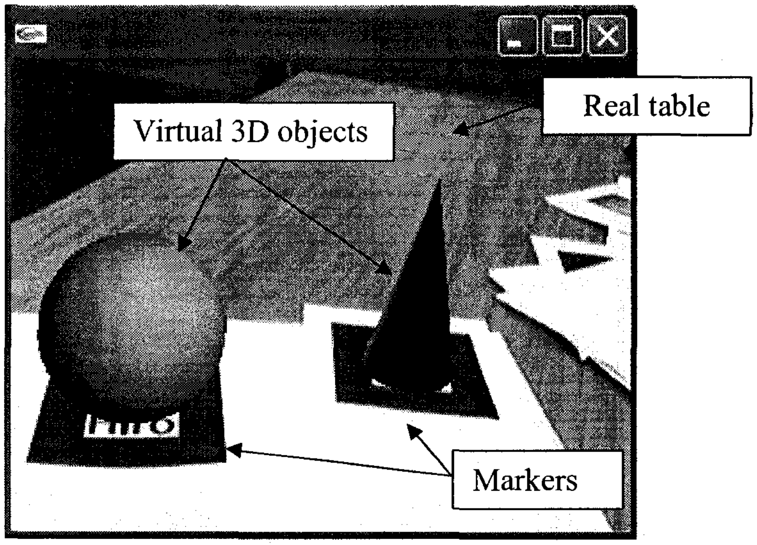


Figure 2.10: An ARToolkit test example (ARToolkit website, 2005)

Figure 2.10 shows an example where two paper markers are randomly placed on a table. Each marker is detected using ARToolkit and virtual 3D objects are rendered. The virtual objects remain displayed on the markers as long as the markers are properly visible even if they are moving. The detection range of the markers and the accuracy of the tracking depend on many parameters, as discussed in Section 3.3.6 of Chapter 3, but the main parameters are the marker edge size, the slant in the depth direction and the distance of the camera from the marker. Figure 2.11 shows the effect of the distance from a marker of 80 mm edge length, and the slant in the depth direction on the accuracy of the tracked positions. The further the marker is from the camera the higher is the error in the position and the more sensitive to the slant the tracking becomes.

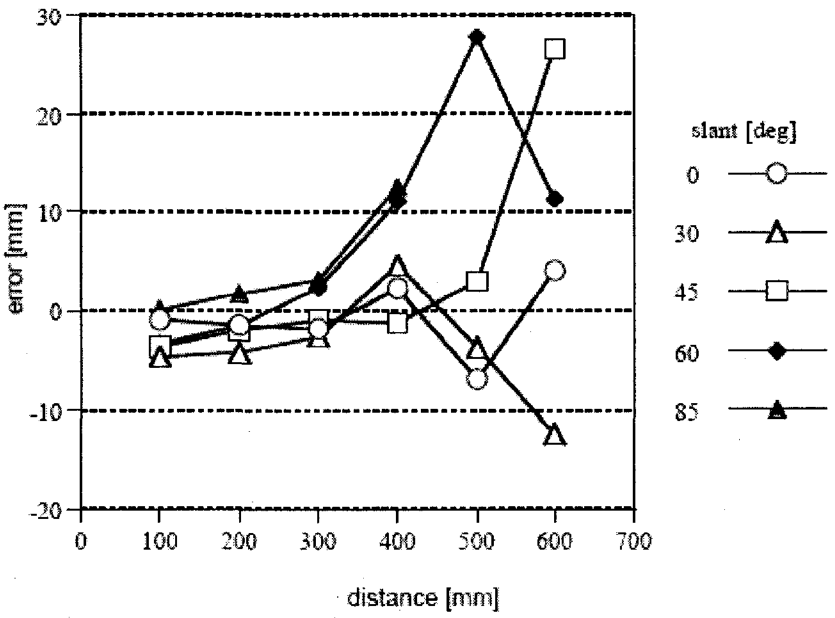


Figure 2.11: Tracking accuracy with respect to distance from a marker of 80 mm

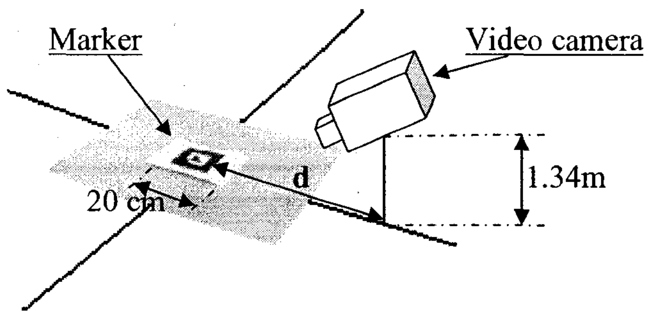


Figure 2.12: Testing the effect of distance on ARToolkit accuracy (Adapted from Pierre et al., 2002)

Figure 2.12 shows the configuration of an experiment carried out to test the accuracy of ARToolkit where the camera is a placed at relatively a long distance ranging from 1 to 2.5 m (Pierre et al., 2002). The marker edge size is 20 cm and the height of the camera with respect to the marker plane is 1.34 m. The results of the test in Table 2.1 show an increasing error value that is directly affected by the increasing distance (d) of the camera from the marker.

Table 2.1: Results of the accuracy test of ARTookit (Pierre et al., 2002)

d (m)	1	1.5	2	2.5
Error (mm)	±14	±18	±22	±27

(2) Cybercode

Cybercode was developed by Jun Rekimoto and Yuji Ayatsuka (Rekimoto et al., 2000) at Sony Computer Science Laboratories in 2000. The Cybercode is a tracking system based on two dimensional barcode fiducials. Cybercode can be used to determine the 3D positions and IDs of tagged objects. The fiducial are required to have four black corners.

(3) ARTag

ARTag is a marker detection system developed by Mark Fiala (2004) at the Institute for Information Technology of the National Research Council. ARTag has been inspired from ARToolkit and is claimed to have lower false positives and lower inter-marker confusion rates. According to ARTag documentation, ARToolkit is faster than ARTag when a small number of patterns are loaded, but for larger number of patterns ARTag is faster. Unlike ARToolkit, ARTag does not use pattern file to detect markers, instead, it uses digital algorithm. ARTag offers the option to use marker arrays to allow different markers of the array to be detected even when one or more are occluded.

(4) Mixed Reality Toolkit (MXRToolkit)

MXRToolkit is a Software Development Kit (SDK) for MR applications; it consists of a set of libraries for fiducial video tracking and for 3D model rendering. MXRToolkit also provides, through its interfaces, much functionality for low-level tracking such as estimation algorithms, optimization, and networking and geometry routines. One of the applications built with MXRToolkit is 3DLive; it consists of an AR system that captures a real person from many viewpoints then superimposes a 3D image of the person onto the fiducial marker in real time (MXRToolkit website, 2005)

(5) Mixed Reality Interface Toolkit (MRIT)

MRIT is an AR interface toolkit developed at the Department of Informatics in the University of Sussex (Liarokapis et al., 2004). MRIT has been proposed to be used as an exemplar for the development of AR applications; it is claimed to provide realistic audio-visual augmentation, such as shadows, without sacrificing its efficiency. MRIT fiducial tracking is based on the wide spread toolkit: ARToolkit. Two display methods are supported in MRIT: the monitor based displays and the video see-through displays with a resolution of 800x600. The tracking robustness of this system is directly affected by ARToolkit registration errors.

(6) OSGAR

OSGAR is a toolkit developed at Georgia Institute of Technology for the development of AR applications (Coelho, 2005); this system automatically computes estimates for the tracking registration errors and makes them available for the AR application designer to be used in setting up the different augmentations corresponding to the different registration errors. OSGAR is claimed to be an adaptive AR system that is aware of the uncertainty in the real world.

(7) Studierstube

Studierstube is an environment for the development of collaborative AR applications. Studierstube is built on top of the Open Inventor graphics API which is, in turn, built on top of OpenGL (Ledermann et al., 2002). Through Open Inventor, Studierstube claims to provide an object-oriented framework for the creation of interactive 3D graphics applications. Studierstube is a collection of Open Inventor extensions, providing support for creating AR applications. Studierstube supports different input and tracking hardware through its tracking middleware, called OpenTracker, as an additional layer between the possibly heterogeneous tracking hardware and its API. The different display devices that can be used with Studierstube are See-through head mounted display, Virtual Table (VT) and other back-projection display surfaces (e. g., a stereo wall or CAVE) and semi-transparent mirror setups, like the Virtual Showcase, that reflect the projected image from a half-silvered mirror.

While Studierstube uses XML as an input language for tracking configuration and uses Open Inventor scripts to load and create 3D models, it does not allow connection between the virtual models and any database management system (DBMS) such as Oracle or SQL Server. In addition, Studierstube is difficult to configure due to the lack of detailed documentation and tutorials and the required manual calculations. Furthermore, Studierstube has been found unable to load VRML files of large size such as buildings.

(8) AMIRE

AMIRE is a framework for the design and implementation of MR applications sponsored by the Information Society Technologies program of the European Union (AMIRE website, 2007). AMIRE has been aimed to allow content experts to design and implement mixed reality applications without detailed knowledge about the needed MR technologies. In this framework, object-oriented properties are used as a generic mechanism for configuration. Different components of the designed MR application communicate through communication slots. AMIRE allows dynamically loading and replacing C++ and XML libraries at run time. The 3D virtual models are loaded in the form of 3ds files (Hoffmann, 2001).

2.9 SUMMARY AND CONCLUSIONS

In this chapter different variations of MR have been explained focusing on AR. The benefits of AR and its applications have been presented as well as the different tracking technologies and the tracking tools used for indoor video-tracking. In addition, several AR systems have been reviewed from the literature, but most of them share the following limitations:

(1) Difficult configuration: The main AR systems that have been tried require significant amount of manual calculations during the configuration of the tracking to be able to register the real world with the VE.

(2) Platform dependency: Most of AR systems that have been found in the literature have been written in the C++ programming language which is a platform dependant language.

(3) Lack of GIS integration: The existing AR systems do not use Geographic Information System (GIS) maps; as a result, they do not consider geographic information in the registration of the virtual models with the real world and they do not use a unique and global coordinate system; instead, many local coordinate systems in the different locals, such as rooms, corridors, etc., have to be defined and used in the computations required to configure and deploy the tracking. In addition, not using GIS maps prevents exchanging geographic data with other systems.

(4) Small scale: The existing AR systems have been used for small environments such as one building or just few floors. Consequently, the application range, in which these systems can be used, is limited.

CHAPTER 3 FRAMEWORK FOR INDOOR VIDEO-BASED AUGMENTED REALITY APPLICATIONS

3.1 INTRODUCTION

As discussed in Chapter 2, the existing video-based AR systems have not been intended for large scale environments and do not use GIS maps; in addition, these systems require time consuming manual calculations for the configuration and registration of the tracking markers.

Taking into consideration the aforementioned shortcomings of the different reviewed AR systems, a framework for indoor video-based AR applications is proposed in this chapter. The framework is based on four main components being large scale virtual models, mobile devices, video tracking and interaction. The proposed approach aims to allow the use of large scale virtual models in AR applications, as well as to provide a practical and easy configuration and deployment of markers for video-tracking.

3.2 PROPOSED FRAMEWORK

The proposed framework (Figure 3.1) combines four main components to form a basis for indoor video tracking in large scale AR applications. The four components are large scale virtual models, mobile devices, video tracking, and interaction. In addition to the common 2D plans and drawings, a large number of architectural engineers design 3D virtual models of exteriors of buildings as well as interiors. This framework proposes, for the creation of the large scale VE, the use of 3D virtual models of the interior and exterior of buildings provided in a standard format such as VRML, GIS maps and Digital Elevation Models (DEMs) of terrains.

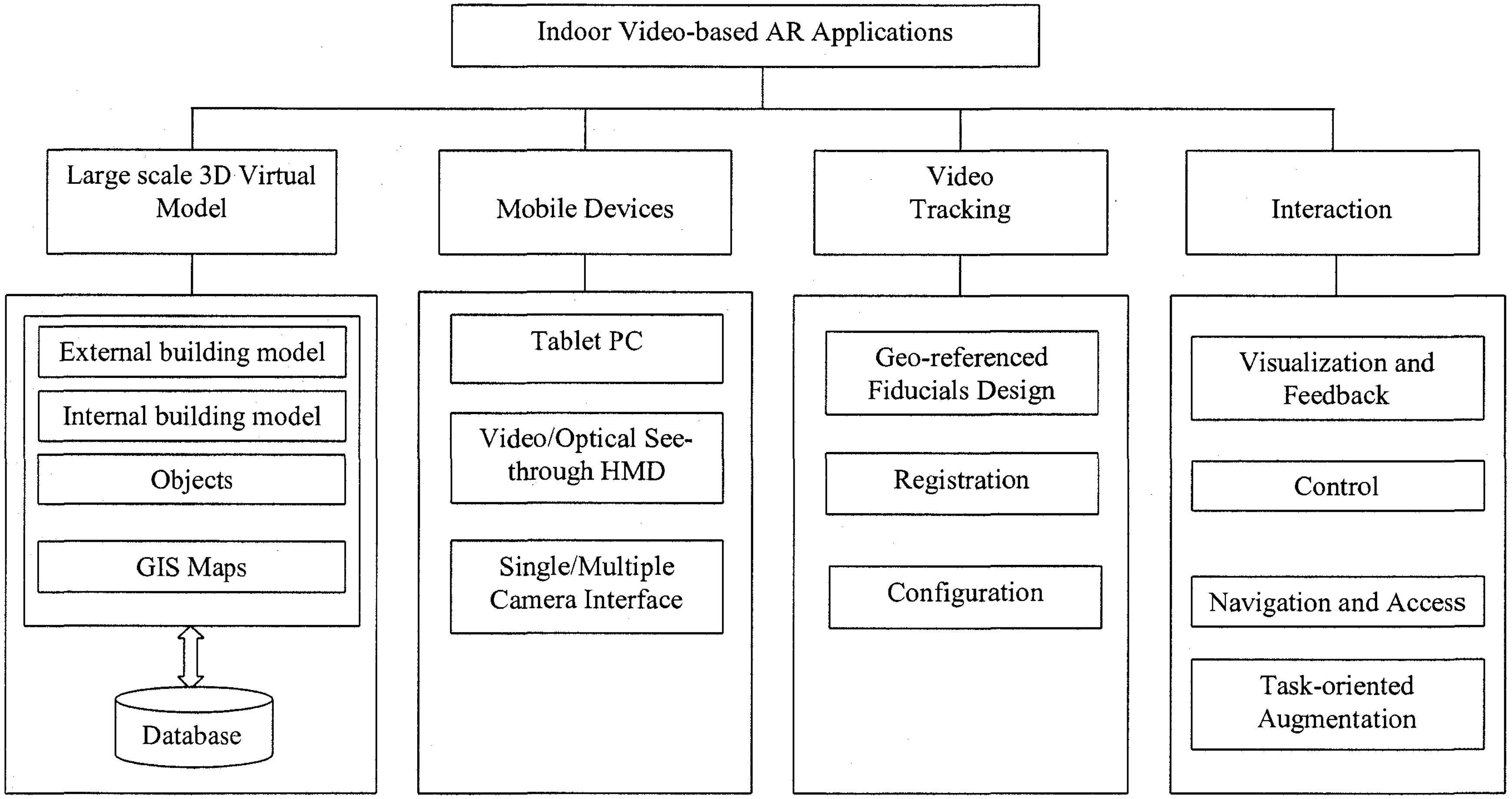
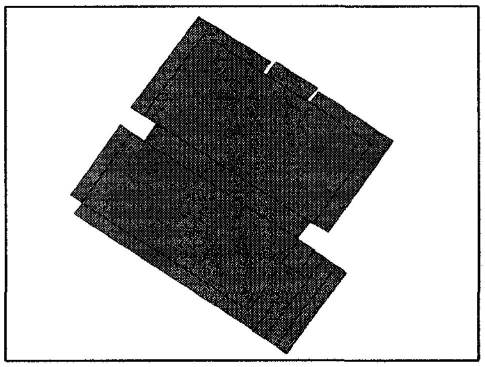

Indoor Video-based AR Applications
Large scale 3D Virtual Model
Mobile Devices
Video Tracking
Interaction
External building model
Internal building model
Objects
GIS Maps
Database
Tablet PC
Video/Optical See-through HMD
Single/Multiple Camera Interface
Geo-referenced Fiducials Design
Registration
Configuration
Visualization and Feedback
Control
Navigation and Access
Task-oriented Augmentation

Figure 3.1 Summary of the proposed framework

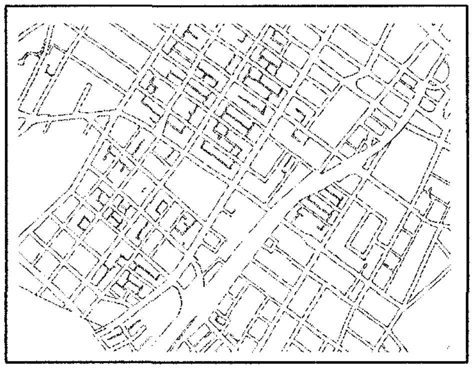
Mobile devices form one of the components proposed in the framework to allow the user to move inside buildings and to be able to see the augmentation on the video of the real world. User interaction is, also, one of the main components of the framework. It is based on five sub-components: visualization and feedback, control, navigation and access. The fourth component of the framework is video tracking. In this framework, the user location and orientation are tracked using real-time video recognition of geo-referenced fiducials.

3.2.1 Large scale 3D virtual model

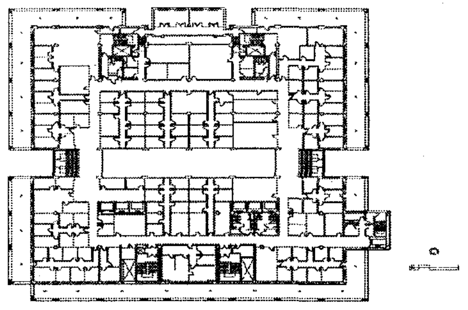
The large scale VE is mainly built based on several GIS layers. First, the exterior contour representation of buildings is constructed and added to a building layer (Figure 3.2(a)). GIS attributes for each polygon in the layer are used to specify different base levels and the heights. Another layer is also added to represent pedestrian areas surrounding the buildings as shown in Figure 3.2(b). Objects of interest such as traffic lights, street lights and street furniture, are added to the 3D model as an object layer (point layer) in the GIS. The facades of buildings are captured in images then applied on virtual buildings using texture mapping techniques. The locations of the images are retrieved from the GIS layer of the exterior of the buildings. The different GIS layers are then transformed into a large scale 3D model of the environment. The layers are used to automatically extrude the 2D shapes into 3D shapes, to add the texture mapping, and to insert the 3D objects into the virtual 3D model. An example of the resulting 3D model is shown in Figure 3.2(d). The interior representation of a building is loaded into the VE in the form of 3D CAD models that have been transformed from 2D drawings, as shown in Figure 3.2(c), to 3D models and translated to a suitable format, such as the VR Modeling Language (VRML).



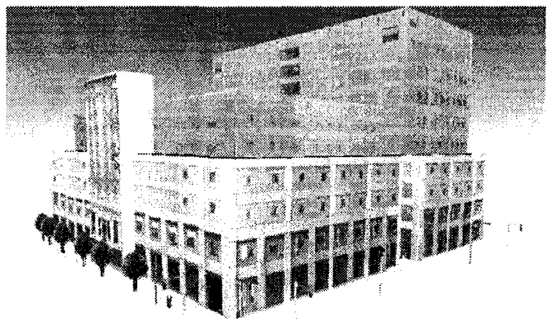
(a) Building layer (polygons)



(b) Block layer



(c) Floor Plan (for extrusion)

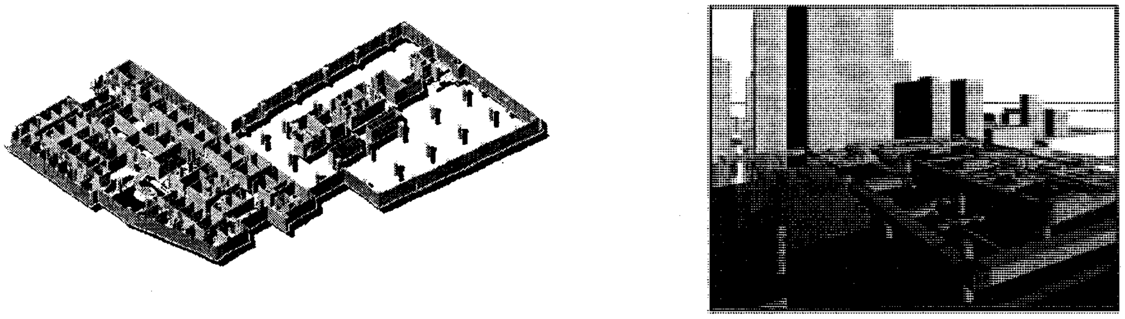


(d) Rendering example of a building exterior

Figure 3.2: Example of data used in creating the 3D

Figure 3.3(a) shows an example of a floor in a building and Figure 3.3(b) shows an example of a Heating, Ventilation and Air Conditioning (HVAC) system of a floor. To be able to load the model of the building into the virtual 3D model at the right global location, two diagonal points from the VE are selected to compute the location and orientation of the CAD model of the building interior. The coordinates of the two points are used to calculate the transformation matrix, including rotation, scaling and translation, from the local coordinate system of the CAD model to the global coordinate system of the VE including the difference that may be in the orientation of the coordinate axes used in different visualization software.

In addition to the geometry, each element of the buildings and the objects in the virtual model is linked to a unique record in a database where the attributes, related to it, are saved. The database can serve as a lifecycle database for cost, scheduling and other information of maintenance, inspection and other activities.



(a) Model of one floor

(b) Model of an HVAC system

Figure 3.3: Examples of detailed 3D CAD models

3.2.2 Mobile Devices

Mobile devices make up one of the main components of indoor AR environments. They are essential for real-time AR applications since the user location is tracked while moving and the view, displayed for the user, needs real-time update to reflect the changes in the location. In this framework, the proposed mobile devices consist of a Tablet-PC equipped with a high-resolution digital video camera, an HMD and a wireless network card; this latter would allow the user to exchange information with other users through a wireless local area network and the Internet. This feature is crucial in case of emergency where the user needs real-time guidance and collaboration with other users; in this case, the different users would be able to see their current locations and also get assistance from a remote office where the real-time locations of the all the user can be spotted. In addition to the aforementioned mobile devices, this framework proposes the use of single or multiple (spherical) digital video cameras, which is a new type of digital video cameras, to be able to cover a wide field of view in the captured video frames, and hence make the video tracking more reliable and stable.

3.2.3 Video Tracking

As mentioned in the conclusions of Chapter 2, the main available video tracking toolkits have been found difficult to configure and to deploy. They all require time consuming manual calculations to find the real world locations and orientations of the tracking markers and to register them with the right virtual world location and orientation. The problem is manifested especially when the toolkits are deployed in larger areas, such as a whole campus or even one large building.

In this framework, an innovative and practical approach has been proposed to facilitate the configuration and deployment of the video tracking for AR applications by eliminating all manual calculations. In the proposed approach, the 3D virtual models of the deployment areas are used not only to be displayed (partially or totally) as augmentation but also to compute the locations and orientations of the markers and to register them with the real world. Since the 3D models are based on real GIS maps and are all in the same coordinate system as the real world, absolute real world locations of the markers are automatically computed by the system. During the tracking, the absolute locations of the markers simplify the system computation of the user absolute location which, in turn, simplifies locating the user inside a building as well as in the global real world coordinate system.

The virtual models are also used to automatically compute the relative locations and orientations of the markers to allow the user to easily locate them in the real world.

The proposed system tracks visual markers placed at configured locations in the real world. The markers are monochrome patterns printed on white papers inside black squares. Before using the patterns, they are captured by the user using a tracking toolkit called ARToolkit and a digital video camera then saved in binary pattern files. The binary pattern files are preloaded into the system prior to the tracking.

The proposed approach is made up of two phases: configuration phase and tracking phase. In the configuration phase (see Section 3.3 for details), the user makes use of the system to interactively specify the locations and orientations of the maker relative to the virtual model and then pastes them in their appropriate locations. Since the VE coordinate system is the same as the real world coordinate system, the measurements and positions in the virtual world can be used in the real world without any scaling or conversion. In the tracking phase (see Section 3.4 for details), based on the absolute real world locations and orientations of the markers and the relative user locations and orientations with respect to the markers, the absolute user location and orientation within the real and virtual world environment are automatically computed in real time and the user's virtual view is updated accordingly.

3.2.4 Interaction

The proposed user interaction is based on the following components (Barrilleaux, 2001): (1) visualization and feedback, (2) control, (3) access and navigation, and (4) task-oriented augmentation.

The proposed framework aims to facilitate the user interaction with the real and the virtual world in AR applications. It improves the data collection and access by allowing the user to interact with geo-referenced infrastructure models to automatically retrieve the necessary information in real time, based on their location and orientation, and the task context.

(1) Visualization and feedback

<u>Displaying graphical details</u>: The proposed system graphically displays to the user detailed information about a room in a building, such as the direction to that room, the working hours of the staff, the status of the personnel (busy, in a meeting, will be right back, etc.) retrieved in real-time from a central database. This can happen in a proactive way based on spatial events, such as the proximity of the user to a door of an office. This helps focus the user's attention on specific locations. The user of the system can control the level of details of representing objects depending on his or her needs.

<u>Displaying non-graphical information and instructions</u>: The user interface can provide links to documents related to the building, such as, regulations and specifications. In addition, the system allows for displaying context sensitive instructions on the steps involved in a specific task, such as instructions about submitting an application form to an admission desk in a university.

(2) Control

The proposed system interprets the user input differently depending on the selected feature and the context. For example, clicking with a pointing device can result in selecting a menu option or in picking an object from the 3D virtual world; it can also result in the creation of a virtual object at the picked location to be used as an indicator for future reference (see Chapter 4, Case Study 2). In addition, the user can directly control the system by physically moving. In this case, the system tracks his/her physical location and orientation and updates his/her view point accordingly.

(3) Access & Navigation

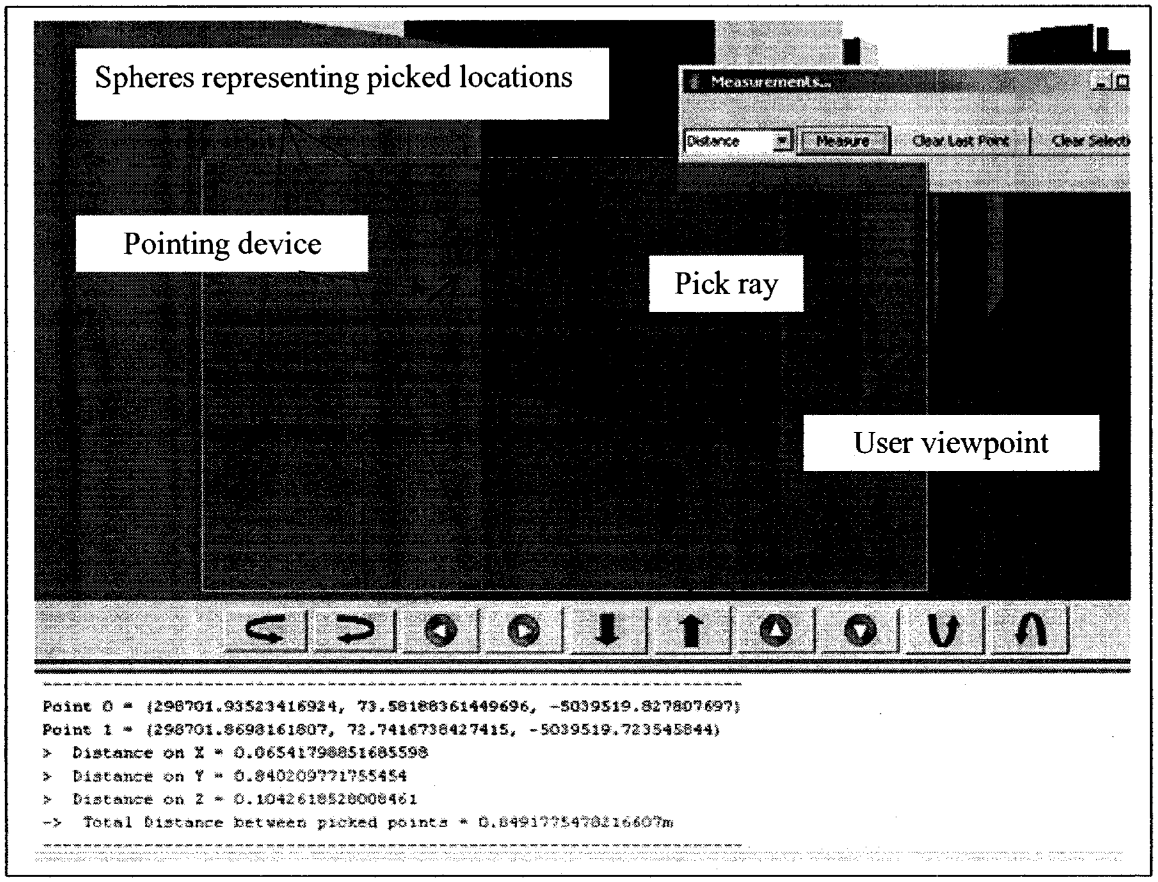
(a) **Access:** Accessing data is directly achieved through a location based behaviour that runs real-time video tracking and triggers the automatic picking feature. The user just needs to physically walk or move to the object for which information is needed then stands in front of it for a short period of time. Then, relevant data will be automatically retrieved from the database and displayed as an augmentation to the real world. Other information can also be retrieved on demand through manual picking or by using menus and buttons.

(b) **Picking Behavior**

The picking behavior consists of using pointing devices to pick 3D locations from the 3D virtual model. It also permits the user to get absolute location in the real world coordinate system without having to perform any conversions.

The picking starts when the user clicks on the scene; once a click is detected, the 2D screen coordinates of the pointing device are retrieved; a 3D location corresponding to the 2D coordinates is computed by referring to the current user view point in the virtual world. A picking ray (other picking shapes such as cylinders and cones can be used) is created and extends from the 3D user location at the view point to the 3D model. A set of vertices where the ray intersects with the model are returned in a list. The closest vertex to the user view point is computed and used to get the closest 3D object; then information about the object is fetched from the database. To each 3D object in the 3D virtual world corresponds an ID that links it to a corresponding record in a database. The database contains relevant information about each 3D object in the virtual world.

Figure 3.4 shows an example of picking a wall from a 3D virtual model of a building. The picking ray extends from the view point to the 3D model. The first object encountered by the ray is the wall which makes it the picked object.

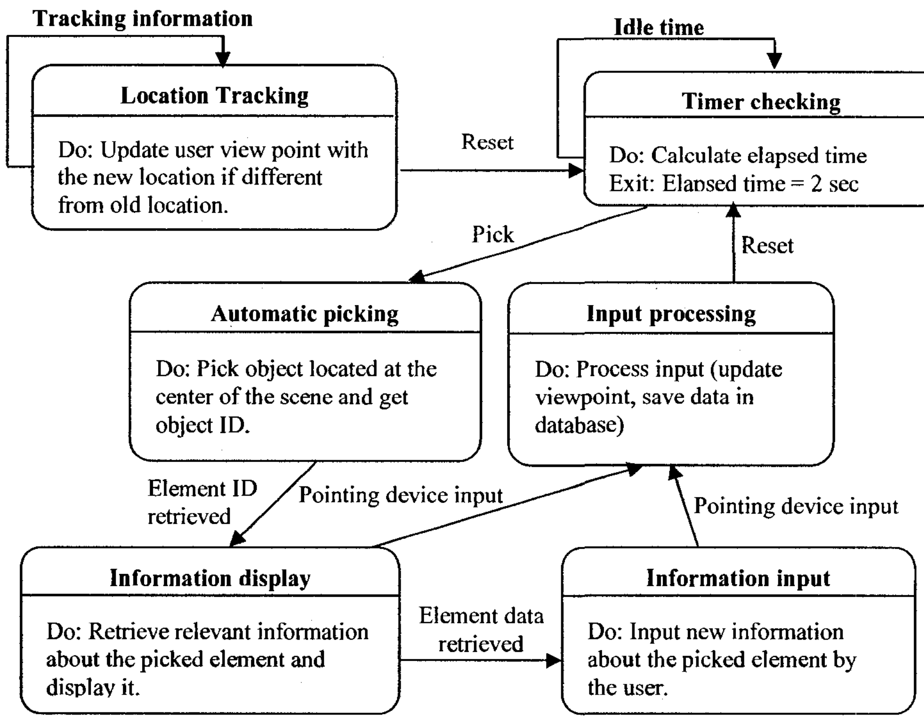


```
Point 0 = (298701.93523416924, 73.58188361449696, -5039519.827807697)
Point 1 = (298701.8698161807, 72.7416738427415, -5039519.723545844)
>  Distance on X = 0.06541798851685598
>  Distance on Y = 0.840209771755454
>  Distance on Z = 0.10426185280008461
->  Total Distance between picked points = 0.8491775478216607m
```

Figure 3.4: Picking Behavior

(c) **Automatic Picking**

The automatic picking process consists of six main operations as shown in Figure 3.5: (1) Tracking user location: the user location is continuously tracked using the indoor video tracking; (2) Tracking user interaction: The user interaction is tracked to find out if the system has entered in an idle state. Whenever the user moves or generates an event (e.g., using the pointing device), a trigger is automatically launched to reset the idle time;



Tracking information
Location Tracking
Do: Update user view point with the new location if different from old location.
Reset
Idle time
Timer checking
Do: Calculate elapsed time
Exit: Elapsed time = 2 sec
Pick
Reset
Automatic picking
Do: Pick object located at the center of the scene and get object ID.
Input processing
Do: Process input (update viewpoint, save data in database)
Element ID retrieved
Pointing device input
Pointing device input
Information display
Do: Retrieve relevant information about the picked element and display it.
Element data retrieved
Information input
Do: Input new information about the picked element by the user.

Figure 3.5: Automatic picking state machine

(3) Computing idle time: The idle time is calculated by comparing the current system time with the last system time when the idle timer was restarted. If the idle time is two seconds, then the automatic picking is launched; (4) Automatic picking: The element that lies at the center of the scene is the one that will be automatically picked. The picking consists of creating a picking tool, selecting a picking mode, creating a picking shape,

picking the closest element, calculating the intersection point, and retrieving the picked element; (5) Displaying relevant information: Based on the ID of the picked element, a query is generated and executed on the database. The returned data is displayed as a text augmentation; (6) Information input: Based on the displayed data about the picked element, the user is able to input new information about the element and save it in the database for future reference.

(d) **Navigation:** In addition to the conventional navigation systems based on 2D maps, the system can also present navigation information in 3D. Within a specific task, the system can guide a user by providing him/her with 3D navigation arrow and focusing his/her attention on the next element of the task. For example, in emergency situations, the system displays 3D arrows to navigate the user to the closest location of an emergency exit. The system may give the user some tips about the emergency type and the recommended actions to take, thus reducing the risk of injuries.

The 3D arrows displayed for the user show him/her the path to any object of interest. The arrow gets oriented from the current user location to the next closest point on the path to the target. The different paths are predefined based on the floor plan and preloaded into the system. If the user cannot take a straight path to the target object because of an obstacle, he/she can take a different path because the system will dynamically guide him/her, based on his/her changing location, to the closest point on the path.

(4) Task–oriented Augmentation

Augmentation can be adjusted by the system based on the task selected by the user. The task context affects the type of objects (physical and/or non-physical) to be displayed as augmentation. For example, in the case of navigation guidance, only non-physical objects, such as 3D arrows, are shown as augmentation to allow for minimum occlusion of the real world. On the other hand, in the case of urban planning, that will be discussed in Chapter 4, the virtual models of future planned buildings can be shown as augmentation.

3.3 TRACKING PROCESS

3.3.1 Registration

The registration refers, as described in Section 2.2, to the alignment of the real world objects with the virtual world objects in the same environment to create the illusion, for the user, that they both co-exist in the same world. Because of the high accuracy requirements of the registration, numerous errors can occur. These errors can be divided into two types: static and dynamic. Static errors are the ones that cause registration errors even when the user's viewpoint and the objects in the environment remain completely still. Dynamic errors are the ones that have no effect until either the viewpoint or the objects begin moving (Drasic et al., 1996).

3.3.2 Marker Requirements

As was mentioned earlier, a marker consists of a custom-drawn pattern printed inside a black square on white paper. Using a toolkit and a camera, the patterns are captured and saved in binary files. To get robust marker detection and a high recognition performance, the patterns must have the following characteristics (Charles et al., 2002, ARToolkit website, 2004):

(1) Unique within the used set of patterns: The patterns should be unlikely to be confused with the other patterns used by the tracking application.

(2) Clear and simple: The patterns should be clearly distinguishable from the surrounding environment and should be as simple as possible to reduce the recognition time.

(3) Black & white: Monochrome patterns are favoured to color ones because the used toolkit (ARToolkit) has been designed to recognize black and white patterns and because uniquely identifying a colour pattern can be drastically affected by the lighting conditions.

(4) Inside a black square: The shape of a square design yields four corner points for tracking purposes. Edges are constructed straight between the corner points. This allows the corners to be determined by line fitting to the edges, yielding measurements that are less sensitive to noise in the vicinity of the corner and quantization errors. The black border of the square yields a maximum contrast relative to the white background. Once the corners have been located, the interior can be warped to a common frame of reference for comparison to a database of marker images.

(5) Not rotationally symmetric: asymmetric patterns allow for the unambiguous determination of the position and orientation of the marker relative to a calibrated camera.

3.3.3 Marker Recognition

The marker recognition is carried out using ARToolkit. The recognition algorithm has been explained in Section 2.8.2. The detection range of a marker depends on many parameters such as the size and the lighting conditions; more details about these parameters can be found in Section 3.3.6.

3.3.4 Tracking Algorithm

The video tracking, as shown in Figure 3.6, starts by grabbing a real time video frame from the video camera. The grabbed video frame is then sent to ARToolkit tracking which analyzes the frame and searches for the previously registered markers (or patterns).

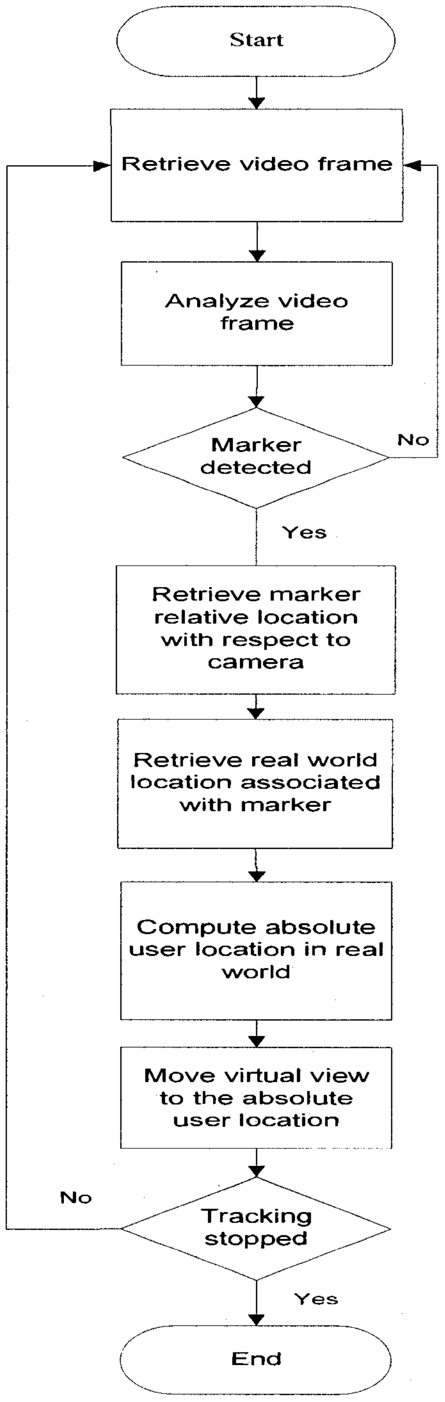

Start
Retrieve video frame
Analyze video frame
Marker detected
No
Yes
Retrieve marker relative location with respect to camera
Retrieve real world location associated with marker
Compute absolute user location in real world
Move virtual view to the absolute user location
Tracking stopped
No
Yes
End

Figure 3.6: Tracking algorithm

When a marker is detected, the transformation matrix defining the marker 3D location and orientation with respect to the camera is computed. Then, the absolute location of the marker in the real world is retrieved from the database. Next, the transformation matrix and the absolute location are combined to compute the absolute camera location and orientation (which are considered as the user location and orientation) in the real world. Then, the virtual world view is moved to the computed absolute camera location and orientation. The absolute camera location and orientation in the real world are used in the virtual world without any modification thanks to the capability of the development environment to represent real world coordinate systems within the virtual world. This cycle of tracking continues running until the user stops it.

3.3.5 Computing Transformations

The used tracking toolkit (ARToolkit) provides a 3D relative location and orientation of the detected paper marker in the video camera coordinate system.

To compute the absolute user location and orientation in the virtual world, two transformations are needed (as shown in Figure 3.7). First, the transformation defining the paper marker location and orientation in the video camera coordinate system (T_{MU}) is inverted to get the location and orientation of the user (i.e. video camera) with respect to the paper marker (T_{UM} Equation 3.1). Second, based on the transformation defining the absolute marker location and orientation in the world, noted as T_{MW}, the final absolute location and orientation of the user in the world (T_{UW} Equation 3.2), are computed as follows:

$$T_{UM} = T_{MU}^{-1} \qquad (3.1)$$

$$T_{UW} = T_{UM} \cdot T_{MW} \quad (3.2)$$

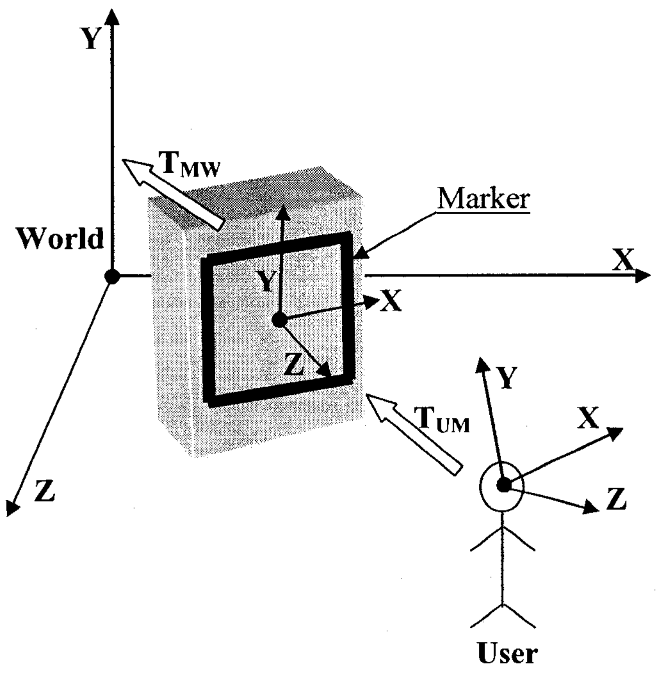


Figure 3.7: Different transformations used in computing the user location and orientation

3.3.6 Video Tracking Parameters

The video tracking in AR applications can be affected by several parameters that are inter-related and strongly depend on the available video capture devices: (1) Field of view: The wider the field of view is the wider will be the area covered and thus the higher will be the probability of detecting a marker. (2) Video resolution and frame rate: The higher the resolution is the more precise will be the captured frames, which reduces false positives. On the other hand, the higher the resolution is the lower will be the frame rate. The frame rate of the video is a main factor that affects the video tracking; the higher the frame rate is the more robust will be the tracking (3) Marker edge size: The bigger the size of the marker is the wider will be the detection range. (4) Light condition: The darker the video frame is the lower is the chance of detecting a marker. The following paragraphs explain how these parameters have been considered in the proposed approach.

(1) Field of View

The field of view of the video of the real world is one of the major factors that can affect indoor AR applications. The wider the field view is, the more complete our understanding of the world around us will be (Drasic et al., 1996). A narrow field of view decreases our confidence in the video of the real world. The field of view of the video is actually the field of view of the digital video camera.

Since the proposed video AR system tracks markers through a video camera, a narrow field of view can also effect the tracking awareness of the system and hence the system robustness. To get a wide field of view, a new type of video camera called spherical can be used in the future. This camera is equipped with five cameras on five sides of a cube; the system tracks the markers from the five sides; the possibility of detecting a marker around the user is very much higher than when just one camera is used.

On the other hand, the field of view of the virtual world is also a crucial parameter in the video AR applications. The virtual objects augmenting the real world have to be in the same field of view as the real world in order to get an augmentation that is realistic and fully consistent with the real world surrounding the user. The proposed system allows the user to manually adjust the field of the view of the virtual world to make it fit the field of view of the camera.

(2) Video Resolution and Frame Rate

A high video resolution provides high precision of video frames which reduces the possibility of confusing markers and thus making the tracking more accurate and reliable. The frame rate is also one of the main factors that affect video tracking. The frame rate is usually related to the video resolution; the higher the video resolution is the lower is the frame rate. A combination of resolution and frame rate needs to be selected based on the user requirements and the hardware limitations. On the other hand, the video resolution affects the field of view; a low video resolution cannot display a wide area covered by a wide field of view; so a wide field of view requires a high resolution to display the whole captured area.

The proposed system allows the user to select a combination of video resolution and frame rate based on the available hardware capabilities.

(3) Marker Edge Size

The marker size directly affects its detection range. The bigger the edge size is the longer will be the detection range. The choice of the marker edge size is related to the number and distribution of markers that can be visible within a certain range; the marker needs to have an edge size that makes its detection range interfere with the detection range of another marker to get a continuous detection range from one marker to the other. So, the choice of the marker edge size is related to the number of markers that can be placed in a certain region. For practical reasons, markers cannot be so numerous or so big in such a way that they disturb the aesthetics or functionalities of the environment by covering most of the wall spaces. Different marker edge sizes have been used in the implementation of the system (more details can be found in Chapter 4).

(4) Lighting Condition

The lighting condition is also one important factor that can affect the video tracking of markers. Since the system tracks markers through a digital video camera, the detection robustness depends on the clarity of the captured video frames; the darker the video frame is the more likely a marker will be confused with other markers and the shorter will be the detection range. The proposed system has been tested in different lighting conditions (more details can be found in Chapter 4).

3.4 CONFIGURATION

The proposed configuration of the video tracking for AR applications is carried out in four phases: (1) Marker design, (2) marker locating in the virtual and the real world, (3) pasting markers at their real locations, and (4) saving the markers locations and the related entities to be deployed by the system.

Phase 1: Designing Markers

Since designing markers that fulfill the marker design requirements is time consuming and needs each marker to be manually captured using the camera and saved into a binary pattern file, a tool called PatternMaker, developped by Johnson et al. (2002, has been used to automatically generate clear and simple patterns including their binary ARToolkit files and their pictures. This tool generates patterns in the form of grids of black squares that are not rotationally symmetric and clear.

Every marker is printed on a white paper along with its name; an arrow pointing to the up direction of the marker is also printed on the paper. The pattern binary file and the picture file also have the same name as the marker to make it easy to match each printed pattern with its files. Figure 3.8 shows an example of a printed marker.

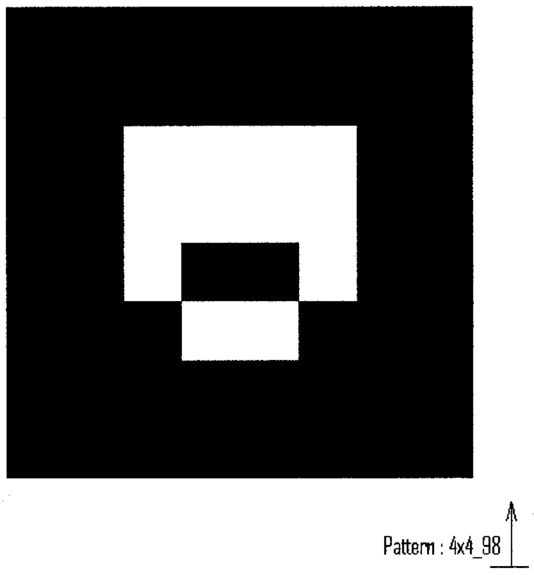


Figure 3.8: Example of printed marker (scaled to fit in the page)

Phase 2: Locating Markers

Marker locating, as shown in Figure 3.9, consists of two main steps: first, the user needs to select the absolute location of the marker in the virtual world that corresponds to its real world location where it will be pasted. Second, to be able to physically paste the marker at their real world location, the user needs to decide on a physical reference edge that will be used in taking measurements that would allow him/her to accurately find the physical location. The two main steps are described below. Markers should be pasted on flat surfaces.

Step 1: Locating markers in the virtual world

(1) The 3D virtual models of the interior of the buildings are loaded into the system.

(2) The user navigates in the virtual scene to the area (e.g. corridor, room) where to place the marker.

(3) Using the picking tool, the user clicks at the virtual location, noted as $\mathbf{P_1}$, which corresponds to the physical location where he/she intends to paste the marker.

(4) The system records the picked location in the format of real world 3D coordinates.

(5) The system computes the normal vector to the surface, noted as $\vec{N}$, at the picked location.

Step 2: Locating markers in the real world

(1) The user decides on an edge, in the virtual model, that corresponds to a physical edge, to be used as a reference in measuring the location of the marker with respect to it (e.g. an edge of a wall).

(2) The user picks two points, noted as $\mathbf{P_2}$ and $\mathbf{P_3}$, on the reference edge to define a vector $\vec{V}$ that defines the direction along which the marker will be aligned. $\mathbf{P_2}$ has to be at a physically identifiable location such as the intersection of a wall edge and the floor.

(3) The system computes the vector $\vec{V}$ and the angle, noted as $\boldsymbol{\alpha}$, between $\vec{V}$ and the line made up of $\mathbf{P_3}$ and $\mathbf{P_1}$.

(4) The system computes the distance, noted as d_1, between $\mathbf{P_3}$ and $\mathbf{P_1}$.

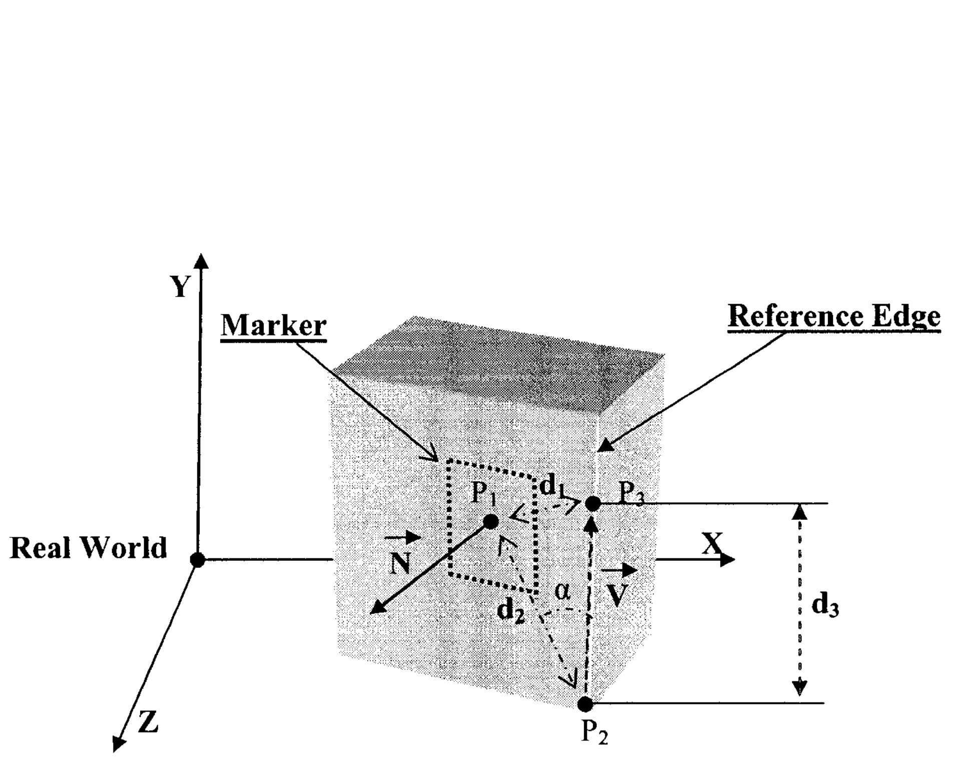


Figure 3.9: Different measurements used in the video tracking configuration

Phase 3: Pasting Marker

While the computed entities $\mathbf{P_1}$, $\vec{V}$ and $\vec{N}$ are sufficient for the system to accurately define the marker location and orientation, the user still needs the other computed entities $\boldsymbol{\alpha}$ and $\mathbf{d_2}$ to easily find the physical location where to paste the marker. The steps to paste the marker are as follows:

(1) Locate the physical reference edge.

(2) At $\mathbf{P_2}$, measure an angle on the surface from the edge equal to α and draw a line of length equal to $\mathbf{d_2}$ from $\mathbf{P_2}$. The end of the drawn line is $\mathbf{P_1}$, which corresponds to the center of the marker to be pasted.

(3) Paste the marker at $\mathbf{P_1}$, making the up direction of the marker parallel to the reference edge.

The computed entities required by the system, vectors $\vec{V}$ and $\vec{N}$ and the point $\mathbf{P}_1$, are then saved in an extensible mark-up language (XML) file. Extra transformations (translation, rotation and scaling) for adjustment to the tracking transformations can also be added in the same XML file.

The measurement of the distances, the angles and the direction and normal vectors are carried out using the measurement tool of the system

The different measurements carried out by the system are depicted in the following equations:

Using the Law of Cosine, the angle α is computed as follows:

$$d_1^2 = d_2^2 + d_3^2 - 2\ d_2\, d_3\ \cos\alpha \qquad (3.3)$$

$$\alpha = \cos^{-1}\left(\frac{d_2^2 + d_3^2 - d_1^2}{2\, d_2\, d_3}\right) \qquad (3.4)$$

Phase 4: Saving the configuration

The different entities computed in the previous phases are saved in an XML file. The format of the XML file has been designed using the XML schema shown in Figure 3.10.

The schema defines a complex type called MarkerLibrary that contains a sequence of markers.A marker is a complex type made up of ten elements as follows:

`3DPoint` is a complex type made up of three components: x, y and z values of type double.

`3DVector` is a complex type made up of three components: x, y and z values of type double.

PatternFileName is a string that contains the file name of the binary pattern file.

LinkID is a string that contains identification information that can be used to link the marker to a record in a database or to any other type of information.

Caption is a string representing a text that can be displayed as an augmentation when the marker is detected; the caption string can be empty.

AbosoluteLocationInWorld is of the type 3DPoint; it represents the x, y and z coordinates of the location in the real world to which the center of the marker is to be associated.

PatternSideDimension is a real number that represents the length, in millimetre, of the edge of the marker.

NormalVector is of type 3DVector; it represents the normal vector to the marker surface when pasted at its real world location.

DirectionVector is of type 3DVector; it specifies the direction vector along which the marker is aligned.

Translation is of type 3DVector; it is made up of three components; each component contains a translation distance, in meter, along each axis (x, y and z axis) of the coordinate system. The translation distance will be added to the tracked user location.

Rotation is of type 3DRotation; it is made up of three components; each component contains a rotation angle, in radian, around each axis (x, y and z axis) of the coordinate system. The rotation angle will be added to the tracked user orientation.

Scale is of type 3DScale; it is made up of three components; each component contains a scaling factor, between 0 and 1, around each axis (x, y and z axis) of the coordinate system. The scale factor will be multiplied by the tracked user location.

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="MarkerLibrary" type="MarkerLibrary" />

  <xs:complexType name="MarkerLibrary">
    <xs:sequence>
      <xs:element name="Marker" type="Marker" minOccurs="1" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Marker">
   <xs:sequence>
    <xs:element name="PatternFileName"          type="xs:string" />
    <xs:element name="LinkID"                   type="xs:string" />
    <xs:element name="Caption"                  type="xs:string" />
    <xs:element name="PatternSideDimension"     type="xs:double" />
    <xs:element name="AbosoluteLocationInWorld" type="3DPoint"   />
    <xs:element name="NormalVector"             type="3DVector"  />
    <xs:element name="UpDirection"              type="3DVector"  />
    <xs:element name="DirectionVector"          type="3DVector"  />
    <xs:element name="Translation"              type="3DPoint"   />
    <xs:element name="Rotation"                 type="3DPoint"   />
    <xs:element name="Scale"                    type="3DPoint"   />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="3DPoint">
    <xs:attribute name="x"                      type="xs:double" />
    <xs:attribute name="y"                      type="xs:double" />
    <xs:attribute name="z"                      type="xs:double" />
  </xs:complexType>

  <xs:complexType name="3DVector">
    <xs:attribute name="x"                      type="xs:double" />
    <xs:attribute name="y"                      type="xs:double" />
    <xs:attribute name="z"                      type="xs:double" />
  </xs:complexType>

  <xs:complexType name="3DRotation">
    <xs:attribute name="x"                      type="xs:double" />
    <xs:attribute name="y"                      type="xs:double" />
    <xs:attribute name="z"                      type="xs:double" />
  </xs:complexType>

  <xs:complexType name="3DScale">
    <xs:attribute name="x"                      type="xs:double" />
    <xs:attribute name="y"                      type="xs:double" />
    <xs:attribute name="z"                      type="xs:double" />
  </xs:complexType>

</xs:schema>
```

Figure 3.10: The XML schema for the marker library

3.5 SUMMARY AND CONCLUSIONS

In this chapter, a framework for AR applications has been proposed to allow the integration of four main components: (1) Large scale 3D virtual models, (2) Mobile devices, (3) Video tracking and (4) Interaction. The framework uses 3D virtual models of real world structures such as building and combines them with real world geographic data such GIS maps in one system that is based on the same coordinate system. In addition, a practical method has been proposed to simplify the configuration and the deployment of markers for the video-tracking. The proposed approach is characterized by the following:

- Integrates large scale 3D virtual models of buildings along with detailed interior 3D virtual models in one single system.
- All 3D virtual models are represented in one single coordinate system that represents the real world coordinate system based on GIS.
- It introduces different interaction methods for indoor AR applications.
- It provides a practical method, for the registration of the virtual world with the real world that covers both the configuration and the deployment stages of the markers.

CHAPTER 4 IMPLEMENTATION AND CASE STUDIES

4.1 INTRODUCTION

Based on the proposed framework, a prototype system has been developed to illustrate its feasibility and practicality. The prototype system has been implemented in Java programming languages, which is platform independent, and used Java bindings for the different tools that are not available in pure Java. Using the object oriented Java3D API, the system seamlessly integrates different 3D virtual models and 2D GIS maps in one single environments that represents the real world. The implemented system also uses Java Database Connectivity (JDBC) to connect a relational database to the different elements in the virtual models. This chapter explains how the prototype system has been implemented and presents three case studies to demonstrate its different implemented features and illustrate their usefulness.

4. 2 SELECTION OF DEVELOPMENT TOOLS

To implement the proposed system, a selection had to be made regarding the main programming language, the 3D graphics Application Programming Interface (API), the 3D virtual models import formats, the database format and the GIS maps API.

The different choices of programming languages were related to the choices of the 3D graphics APIs. The main 3D graphics APIs available are OpenGL and DirectX. The OpenGL API is known by its speed of rendering and by its availability on many platforms such as Unix, Linux and Windows. DirectX is also a popular 3D graphics API but available only on the Windows platform. The main programming languages considered were C++ and Java. Java is a multi-platform object-oriented programming language; it is known by its advanced capabilities such as servlets and applets that run on almost all available platforms without the need to recompile them. It is known also for its multithreading, network programming, simplicity, and ease of use. In addition, an important amount of Java reusable libraries are available in the World Wide Web. However, Java applets and applications do not run as fast as the applications written in other programming languages such as C++. On the other hand, C++ is also an object-oriented programming language; it is known by its run and compilation speed; but, C++ is not platform independent and is not as easy to learn as Java.

Based on the advantages of Java, it has been selected as the main programming language in implementing the proposed system. In addition to the aforementioned advantages of Java, an object-oriented 3D graphics API is also available in Java, known as Java3D. Java3D allows the programmer to describe a 3D scene using coarser-grained graphical objects and defining objects for elements such as appearances, transformations, materials, lights, etc. Compared to OpenGL, the code of Java3D is more readable, maintainable, reusable, and easier to write (Selman, 2002).

Building custom applications that incorporate GIS and mapping capabilities in Java was possible by using a Java edition of the GIS API MapObjects. It helps the programmers build applications that perform a variety of geography-based display, query, and data retrieval (MapObjects-Java, 2006).

The standard used for importing 3D VR models is VR Modeling Language (VRML). VRML is the most popular interoperability standard for describing interactive 3D objects and virtual worlds delivered across the Internet (Nadeau and Moreland, 1996).

The database of the 3D objects in the VE is designed with Microsoft Access to represent the information of all the objects, such as buildings. JDBC is used to access information stored in databases. The details about software requirements and installation guide of the prototype system are included in Appendix C.

Based on the selected programming language, Java, the software tools and APIs shown in Table 4.1 and Table 4.2 have been selected for the purpose of covering most of the development phases. In order to allow the project to be shared among different developers Microsoft SourceSafe has been selected to manage the database of the source code. JUnit has been selected to write and run different unit testing inside the same development environment.

Table 4.1: Software tools used in the implementation

Tool	**Purpose**
Borland JBuilder Enterprise version	Development environment
Javadoc	Inline documentation generator
JUnit	Unit testing
Microsoft SourceSafe	Source code version control
Microsoft Access	Database management

Table 4.2: Software APIs used in the implementation

API	Purpose
Java3D	3D graphics
JARToolkit	Marker detection
MapObjects Java (MOJ)	GIS Interface
JDBC	Database interface

4.3 BACKGROUND OF THE CASE STUDIES

Concordia University downtown campus (Sir George Williams Campus) in Montreal has been chosen as the subject of the case studies (Figure 4.1). Concordia is a large urban university in Montreal. The growth of Concordia's downtown campus has led to build two new buildings, the Integrated Engineering, Computer Science and Visual Arts Building and the new John Molson School of Business.

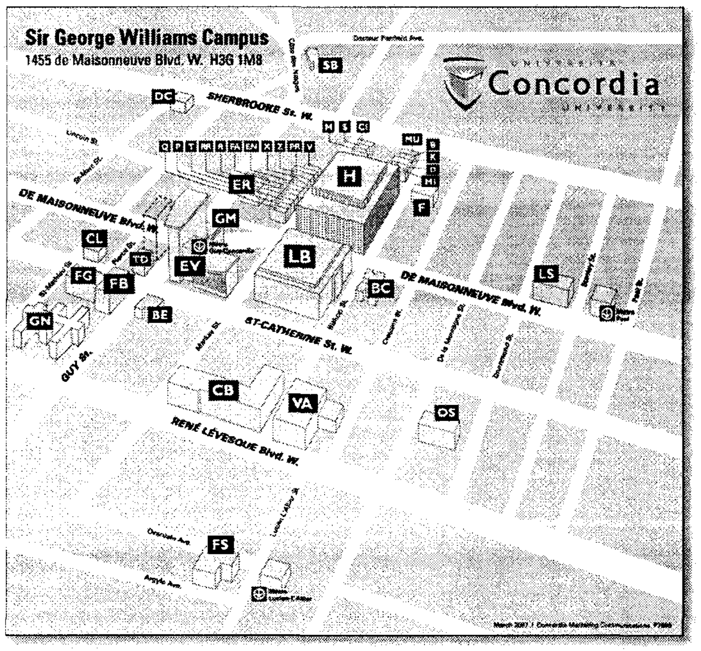


Figure 4.1: Concordia University campus map

4.4 HARDWARE COMPONENTS

The hardware components that have been used in the implementation of the framework and the different case studies are shown in the Table 4.3 and consist of a Tablet PC and a Laptop, an HMD and a digital video camera.

Table 4.3: Hardware components of the prototype system

Device Type	Brand	Specifications
HMD	MicroOptical	Resolution: 800x600 Transparency: opaque FOV: 16° horiz./20° diag.
Tablet PC	Panasonic ToughBook CF-18	CPU: 1 GHz RAM: 512 MB
	Toshiba Tecra M4	CPU: 1.86GHz RAM: 1 GB
Laptop	Dell XPS	CPU: 3.2 GHz RAM: 4 GB
Digital Video Camera	Logitech QuickCam for Notebooks Pro	Resolutions: 640x480 @ 30fps

4.5 IMPLEMENTING THE FRAMEWORK

4.5.1 Database model

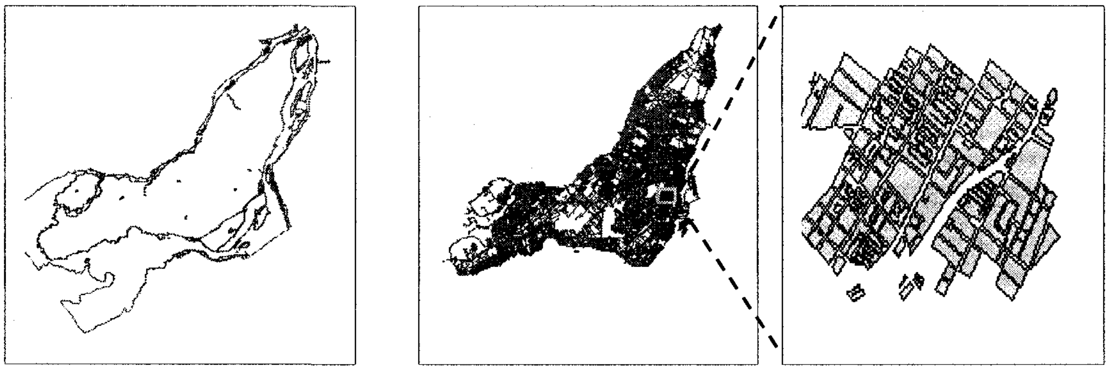
Object-relational data model has been used in the proposed framework in the design of the database. The data has been stored in a relational database while the operations on the database have been developed in the object-oriented language: Java. To be able to retrieve and store data from and into the relational database JDBCD has been used. Using

JDBC, data is retrieved by passing an SQL (Structured Query Language) query to the database. Thanks to the JDBC, the database does not have to be in a Microsoft Access file but rather it can be an Oracle database or any other format that can be accessed by JDBC since, just like Java, JDBC is a platform independent interface.

While most of the data have been stored in a relational database, the tracking configuration information has been stored in an XML file. The latter offers an ease of use and customization in addition to its popularity and its wide use in configuring applications. Examples of the used XML can be found in Appendix B.

4.5.2 GIS integration

A GIS sub-system is created using MapObjects Java Edition (MapObjects-Java, 2006). The map includes several layers (Figure 4.2) related to Montreal City, such as a border layer (Figure 4.2(a)) and other layers for roads (Figure 4.2(b)), blocks around the campus (Figure 4.2(c)) and Concordia buildings.



(a) Border layer (b) Road layer (c) Downtown campus

Figure 4.2: GIS information

4.5.3 Building the VR model of the campus

The 3D virtual model of Concordia downtown campus is developed using the following data: (1) 2D CAD drawings of the buildings obtained from the Facilities Management

Department of the university; (2) A digital map of the city of Montreal (roads and blocks) obtained from the municipality of Montreal; (3) VRML library of small objects developed to embed in the 3D model, such as traffic lights, fire hydrants and street furniture; and (4) Orthogonal digital images of the facades of the buildings collected using a digital camera.

The 2D CAD drawings of the buildings are imported as a layer in ArcView (ArcView, 1996) and edited to create the outline of buildings. The map of the area is imported in ArcView to create the block layer. The other layers including the image layer, object layer and tree layer are created using ArcView. The required attribute information of the layers is input in the attribute tables of the layers.

The GIS layers, images and 3D objects described above were integrated and translated into VRML. The translator application developed in Visual Basic uses a GIS library (MapObjects) to extrude the GIS shape files and create a number of VRML files that constitute the virtual 3D model.

4.5.4 Virtual Scene Visualization

Java 3D has been used as the main 3D graphics API. In Java 3D, the VE can be created in the form of a *scene graph*. Scene graphs are defined by a tree structure grouping different nodes of groups and sub-groups of virtual objects. Virtual objects define geometry, location, orientation, appearance, sound, light and many other compound types. *BranchGroup* nodes are used to form a tree structure based on parent-child relationships (Figure 4.3). *TransformGroup* nodes can be constructed to form a group of geometry objects on which a *Transform3D* object can be applied. *Transform3D* defines a geometric transformation for the objects grouped by the parent *TransformGroup*. Geometric transformation can be a combination of translation, rotation and scaling (Walesh and Gehringer, 2001).

In the AR mode, the scene graph properties, such as transparency and polygone mode, can be modified to decrease the occlusion of the real world. Using the user interface, the user can manually adjust, at run time, the different visual properties of the scene graph to make them convenient for the AR mode.

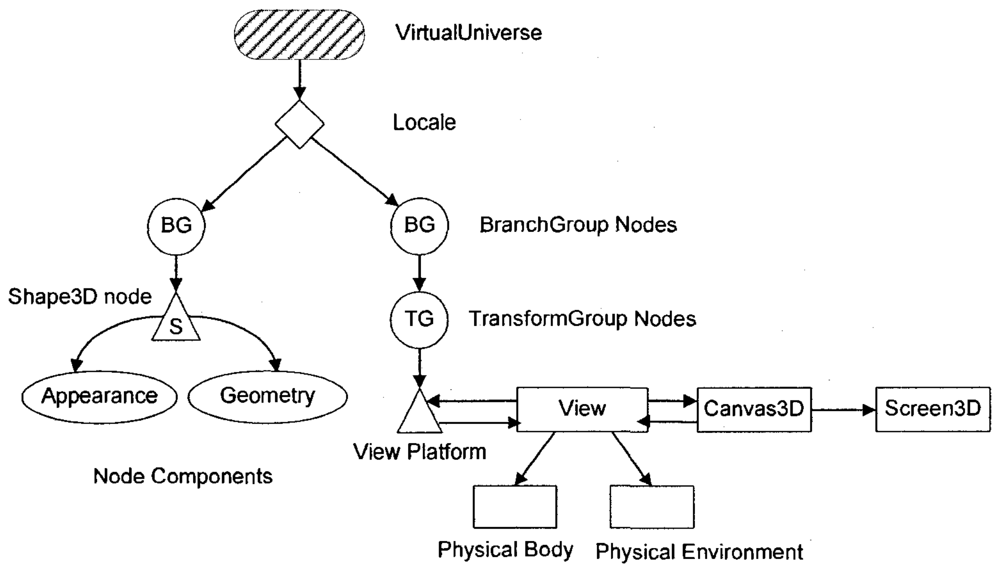


Figure 4.3: Scene graph (Walesh and Gehringer, 2001)

To visualize the VE, two main BranchGroups have been created: The Augmentation BranchGroup and the Virtual Model BranchGroup.

(1) Creating the Augmentation Branch: To be able to show the augmentation when other virtual objects of the model are hidden (e.g. the buildings are hidden but HVAC ducts are shown), the virtual augmentation needs to be created and attached to a separate BranchGroup called AugmentationBranchGroup. The augmentation is dynamically created and attached to the virtual scene graph. It can be of two types: a 3D Text or a 3D shape. The text content of the 3D Text Augmentation is loaded from the Caption field from the XML marker configuration. The 3D text augmentation is created using the 3DOrientedShape class of Java3D. This class creates a 3D text that is always facing the virtual user view point. The 3D shape augmentation can be a 3D directional arrow, a sphere, a cylinder, a cube or a 3D line. The 3D directional arrow is dynamically created when the user selects the direction option to get assistance in finding a target (see Case Study 1 for more details). The sphere, the cylinder, the cube and the 3D line can be created by the user by picking their 3D locations at run time (see Case Study 2 for an example of creating a sphere). The 3D line can, also, be automatically created, when the user selects the direction option, to highlight a path to a target.

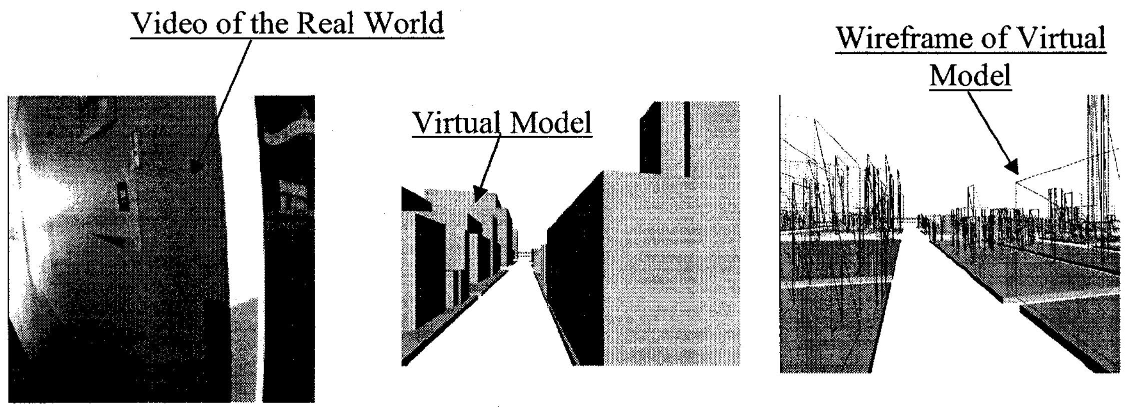
(2) Creating the Virtual Model BranchGroup: The virtual model is created then imported into the system in the form of VRML files. The VRML files are imported into the Java 3D scene graph using the VRML 97 API (J3d.org, 2006). The different objects loaded from the VRML files are assigned unique IDs to link them to a database that contains more information about them. The unique IDs are automatically loaded into the system and used to retrieve information about the objects at run time using the JDBC interface. The different virtual models loaded into the system can be both two dimensional and three dimensional:

(a) *Loading three dimensional CAD files*: As an example of 3D CAD model, one floor of a building (the fifth floor of the EV building) is prepared and translated into VRML. In order to locate the model of the floor in the virtual 3D model, two points that correspond to two corners on a diagonal of the building are picked. The coordinates of the points are used to calculate the transformation matrix, including rotation, scaling and translation, from the local coordinate system of the CAD model to the global coordinate system of the virtual model. One of the main issues encountered in positioning the virtual model of the floor is the difference in the coordinate system axes used in different 3D CAD Tools. For example, the axis representing the height direction is considered as the Y-axis in Java 3D and as the Z-axis in 3D Studio Max.

(b) Loading two dimensional CAD files: 2D drawings are loaded in the format of DWG and DXF files. DWG files are loaded and visualized using the DWGLoader library (MapObjects Java, 2003). The 2D DXF files are loaded and visualized using DXFLoader library (j3d.org, 2006). 2D drawings such as the floor plans can be loaded into the augmentation BranchGroup then superimposed on the real world to allow the user to get a realistic view of the plan before building the walls.

(3) Display Modes: Different display modes (Figure 4.4) can be selected at run time to allow the user manually adjust the virtual model according to his/her needs. The virtual model can be rendered in solid mode, wireframe mode, or transparent mode.

The transparent mode is used in the AR environment setting. In this case, the background of the virtual scene is filled with real-time video frames coming from the digital video camera and most of the virtual model is made transparent except the selected augmentation such as the directional 3D arrow. The wireframe mode can, also, be used in the AR environment setting since it does not completely occlude the video background of the real world. The solid mode is used during the configuration stage in VR setting to register the markers with the real world.



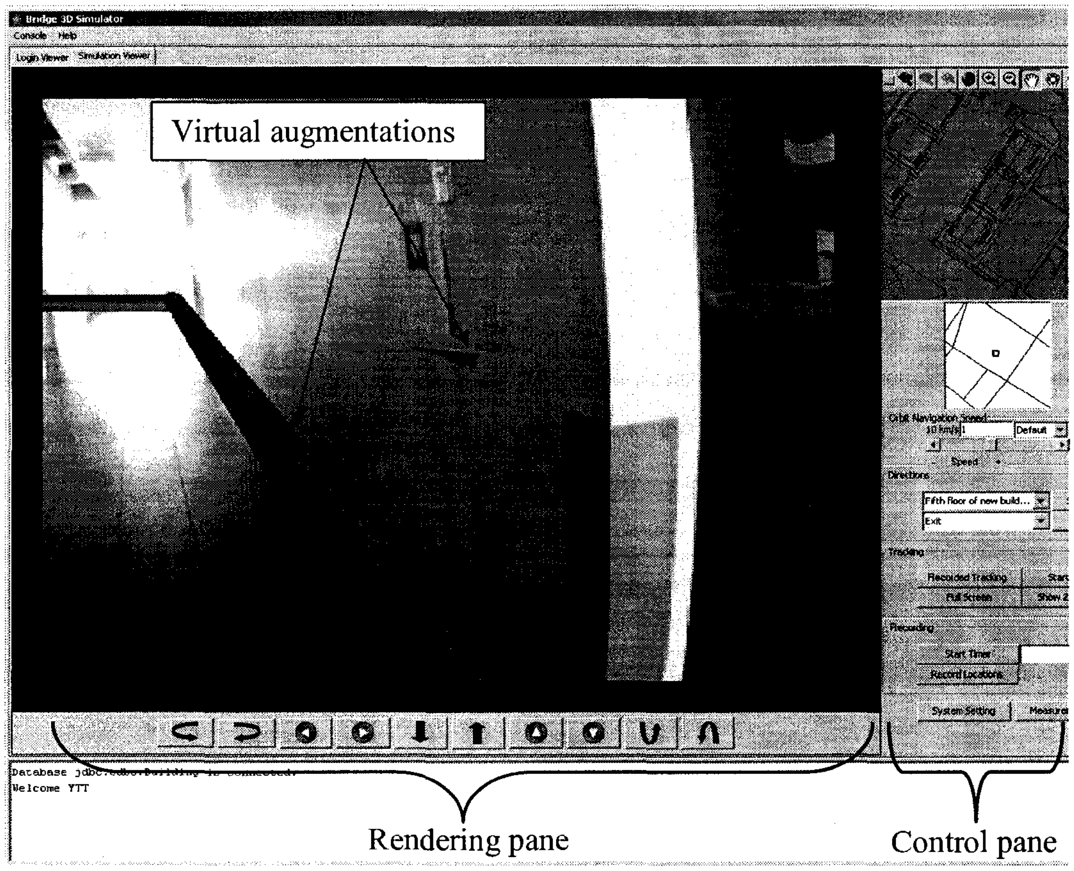
(a) Transparent mode (b) Solid mode (c) Wireframe mode

Figure 4.4: The different display modes

4.6 USER INTERFACE DEVELOPMENT

Since some of the VR operations cannot be carried out in the AR mode, the system automatically adjusts the user interface to leave only applicable operations based on the mode (VR or AR) selected by the user.

Figure 4.5 shows the main user interface of the system. It is made up of two main panes: a *Rendering Pane* and *Control Pane*. In the *Rendering Pane*, the video of the real world, retrieved from the digital video camera, is rendered. On top of the real world, the 3D Virtual objects making up the whole VE or just the augmentation are also rendered. The *Control Pane* is made up of a set of sub-panes: a *Tree sub-pane*, a *GIS map sub-pane,* a *Directions sub-pane,* a *Tracking sub-pane,* a *Recording sub-pane*, and a *System Setting and Measurements sub-pane.* The *Tree sub-pane* is for tree navigation of the VE. It contains a logical tree of a set of buildings that is automatically created by reading the names of the buildings from a database. Using the tree, the user can highlight a building. The *GIS map sub-pane* contains two maps: The first one is a 2D GIS map representing the streets; the second one is a detailed map representing one floor plan. The 2D maps show the current user location. Since, in AR mode, the location of the user is continuously tracked, the 2D maps are automatically updated with the new location each time a change is detected.



```
Database jdbc:...
Welcome YTT
```

Figure 4.5: Main User Interface

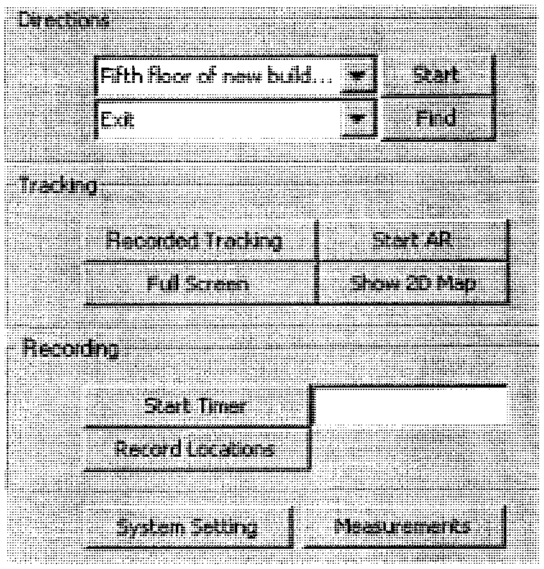


Figure 4.6: Control Pane

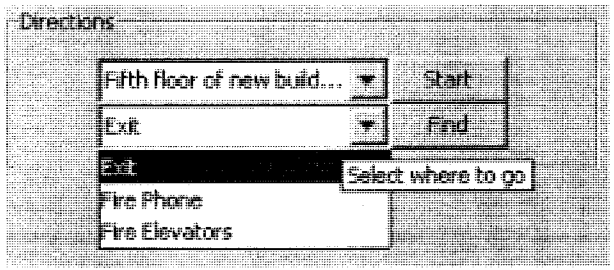


Figure 4.7: Direction assistance features

The *Directions sub-pane* allows the user to get real-time direction assistance to find a selected destination (as shown in Figure 4.7). The user selects a destination from a list of destinations that are preloaded into the system. For example, if the user selects *Exit,* the system displays a 3D path in the form of a thick colored line that goes to the closest exit in the floor. In addition, the system displays, as discussed in Section 4.8, an animated 3D arrow that is oriented to the path that takes to the closest exit.

The *Tracking sub-pane* allows the user to start AR mode by clicking on *Start AR* which automatically starts the video-tracking. The user can click on *Show 2D Map* to show or hide the *GIS map sub-pane*. The user can, also, play a prerecorded tracking path by clicking on the *Recorded Tracking* button which gives the user the option to browse to the file that contains the prerecorded path and play it. The *Full Screen* button allows the user to hide all panes to leave more space for the *Rendering Pane.*

When the video-tracking is started, the system connects to the available digital video camera; once the connection is established the system automatically switches the displayed 3D virtual models to a full transparency mode and starts rendering the video frames into the background of the *Rendering Pane* as shown in the left part of the Figure 4.5. The system, also, automatically displays the *AR Setting sub-pane* (Figure 4.8) in which the user can change the field of view of the 3D virtual scene, adjust its transparency level as well as the tracking scale. The field of view option allows the user to adjust the virtual world scene to the field of view of the video camera. On the other hand, the transparency option allows the user to manage the occlusion of the real world, during the AR mode, by making the virtual scene in different levels of transparency. The *Tracking Scale* option allows the user to modify the scale in which the coordinates retrieved from the tracking are considered. The tracking scale of 1 means that 1 meter in the real world corresponds to 1 meter in the virtual world since the virtual world is in the same coordinate system as the real world. Changing the scale to a number 'n' means that moving in the real world of 'n' meter corresponds to 1 meter in the virtual scene.

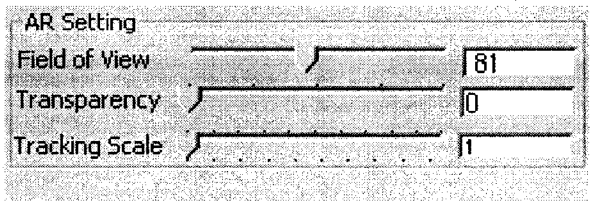
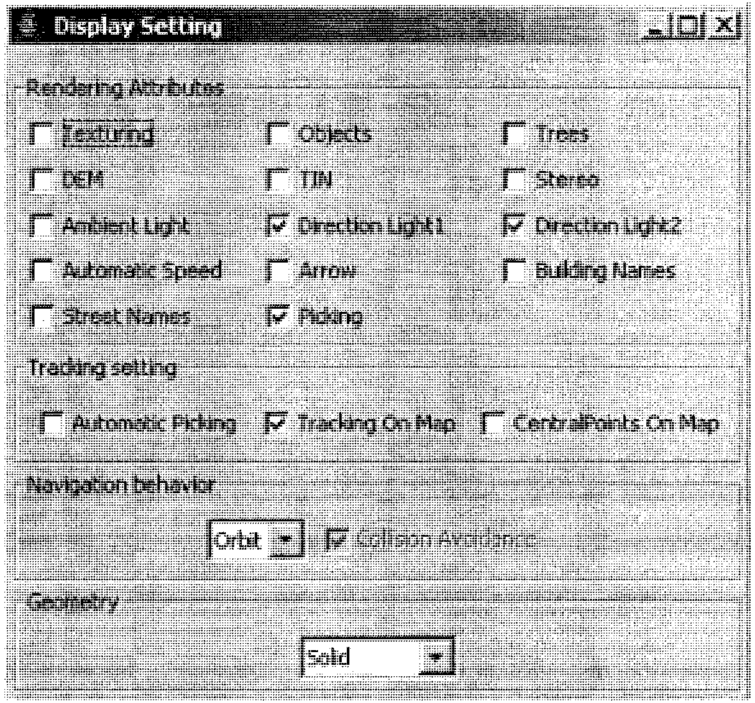

AR Setting
Field of View 81
Transparency 0
Tracking Scale 1

Figure 4.8: Setting options for AR mode

The *Recording sub-pane* allows the user to record his/her changing locations into a file that can be played using the *Recorded Tracking* option. In the same *sub-pane*, the user can get a timer for the recording to see how long it took when moving from one location to another.


Display Setting

Rendering Attributes

☐ Texturing ☐ Objects ☐ Trees

☐ DEM ☐ TIN ☐ Stereo

☐ Ambient Light ☑ Direction Light1 ☑ Direction Light2

☐ Automatic Speed ☐ Arrow ☐ Building Names

☐ Street Names ☑ Picking

Tracking setting

☐ Automatic Picking ☑ Tracking On Map ☐ CentralPoints On Map

Navigation behavior

Orbit ☑ Collision Avoidance

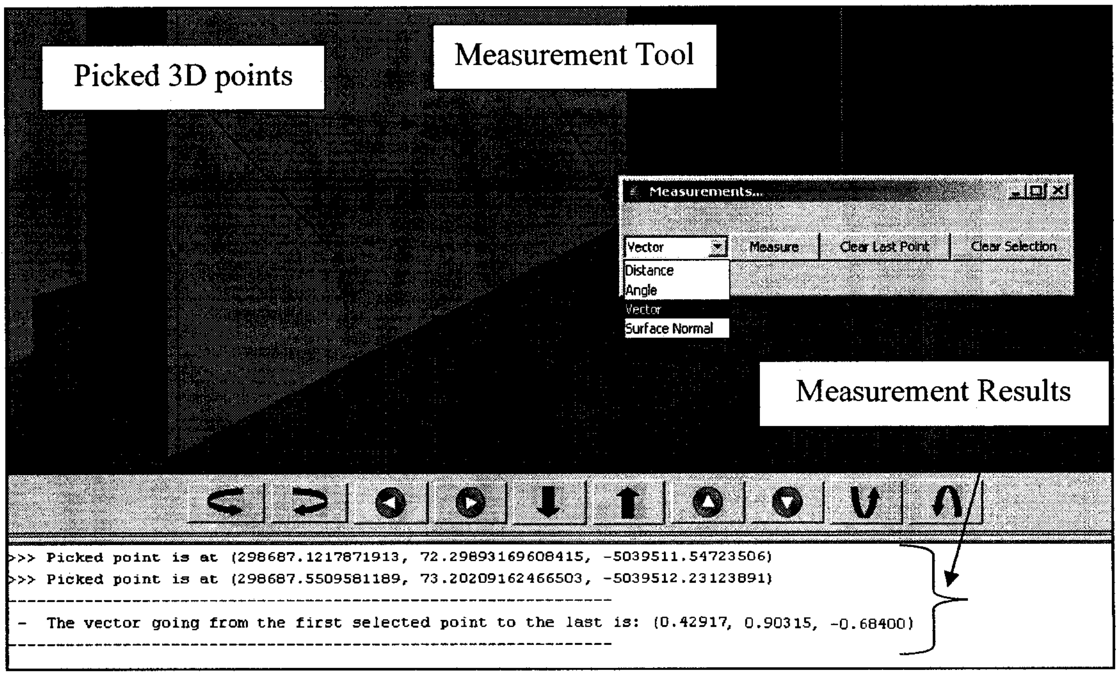
Geometry

Solid

Figure 4.9: System Display Setting options

Figure 4.9 shows the *System Display Setting* that can be displayed by clicking on *System Setting* in the *Control Pane*. The *System Setting* is designed to change the different parameters of the system. This sub-pane is made up of the *Rendering Attributes* group, *Tracking Setting group,* the *Navigation Behavior group* and the *Geometry group.* The *Rendering Attributes* group contains several check boxes that can be used to control the different rendering settings of the VR. The main settings are the visual cues, such as building names, the different 3D lights, the manual picking, and the stereoscopic rendering mode. The stereoscopic mode can be used to render the VE in a more realistic mode which helps in perceiving depth in the virtual augmentation. This option can be used only if the user wears an HMD capable of displaying two different images for each eye. The *Tracking Setting group* contains the *Automatic Picking* option that can be enabled or disabled to allow to user to pick an element from the VE by looking at it continuously for a certain short time (e.g. 2 sec). In the same group the user can select the *Tracking On Map* which allows the user to see his/her location highlighted in the *2D GIS Map sub-pane* while he/she is moving. The user can, also, have the 2D maps automatically zoomed-in to show more details around the his/her current location using the *CentralPoints On Map* option. The *Navigation Behavior group* can be used to change the navigation type in the VE when the VR mode is turned on. The different navigation types are: *Orbit*, *Fly* and *Drive*. The *Geometry group* allows the user to change the polygon mode for all objects in the VE to make them in different modes: wire-fame, point cloud or solid.

From *Control Pane*, the *Measurements* tool (Figure 4.10 and Figure 4.11) can be started to allow the user to perform different kind of measurement on the VE. This tool can be used to measure four type of entities (as shown in Figure 4.11): the distance between two or more points, the angle between two lines made up of three points, the vector going from one point to another and the surface normal at a point. This tool is used in combination with the picking tool; so the user needs to pick 3D point using a pointing device; the picked points would be highlighted using red spheres. Then, the user needs to select which entity to compute and click on *Measure*. The system performs the selected measurements and displays the results in the bottom of the *Rendering Pane* inside the *Message* window. The *Measurement tool* is used in configuring the markers for video-tracking as discussed in Section 3.4.



```
>>> Picked point is at (298687.1217871913, 72.29893169608415, -5039511.54723506)
>>> Picked point is at (298687.5509581189, 73.20209162466503, -5039512.23123891)
------------------------------------------------------------
 -  The vector going from the first selected point to the last is: (0.42917, 0.90315, -0.68400)
------------------------------------------------------------
```

Figure 4.10: Using the Measurement Tool

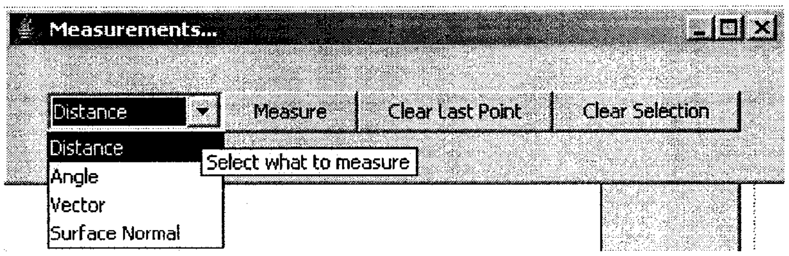


Figure 4.11: Different options in the Measurement Tool

4.7 UML CLASS DIAGRAMS

4.7.1 Description of the main classes

(1) **Class CurrentArrowBehavior** (Figure 4.12): This class is derived from behavior class in Java3d. The interaction with 3D scene in Java3D is provided through behaviors. This behavior wakes up every time the location of the viewpoint in the VE changes, gets the coordinates of the new location and places an arrow oriented from this location towards the next destination.

(2) **Class Marker** (Figure 4.13): This class holds all necessary information about a marker to which will be associated an absolute 3D location in the VE.

(3) **Class ARPatternTransformGroup** (Figure 4.14): This class extends the Java3D TransformGroup. It is used for retrieving the relative location and orientation of the user with respect to the camera and computing the final absolute location and orientation of the user. It is, also, used for displaying 3D text augmentation that is always facing the user, based on the computed location and orientation.

(4) **Class ARBehavior** (Figure 4.15): This is a Java3D behavior continuously displaying the video streams coming from the camera and scanning them for the markers. When a marker is detected, this behavior notifies ARPatternTransformGroups with the relative coordinates of the user with respect to the camera.

(5) Class PickHighLightBehavior (Figure 4.16**):** This is a class derived from the Java3D MouseBehavior which is derived from Behavior. This class is used for the manual 3D picking. When a mouse click is detected a pick ray is created and used by the PickIntersection class to compute the intersection points with the 3D model. The points are then return the PickHightLightBehavior class which computes the closest one to the user location then creates a sphere to show it.

(6) **Class MainConsole**: This is a Java frame window that holds, in a tab structure, JcbSimulationViewer panel and the Login panel.

(7) **Class Arrow** (Figure 4.12): Creates a 3D arrow that consists of a Cylinder and a Cone.

(8) **Class ModelPane** (Figure 4.12 and Figure 4.14): This class is an applet contained inside a JPanel that is attached to the JcbSimulationViewer. It runs the 3D rendering of the Java3D canvas.

(9) **Class JARToolkit3D_Main** (Figure 4.15): This is a class responsible for initializing the AR mode, configuring the markers and attaching the AR tracking behavior to the 3D canvas (ModelPane).

(10) **Classs JcbSimulationViewer** (Figure 4.16): This class is Java panel designed to contain all the 2D user interface components (e.g. buttons, slider bars, list, etc.) and the 3D canvas.

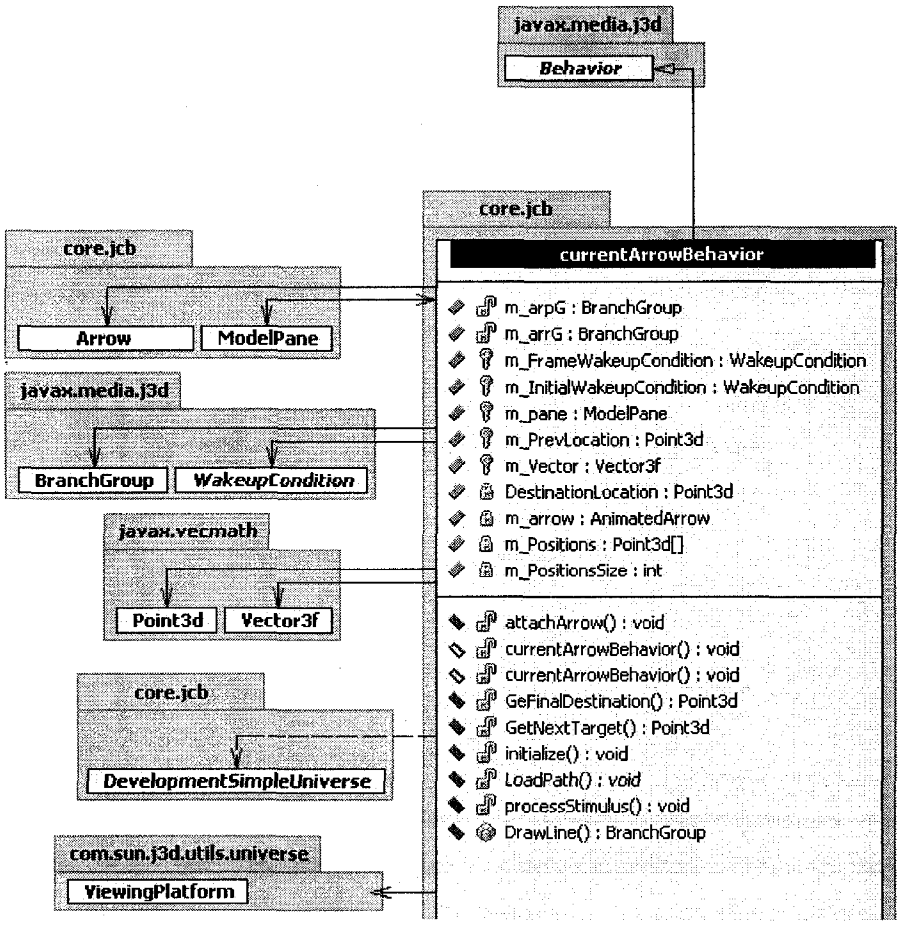


Figure 4.12: Class diagram for CurrentArrowBehavior

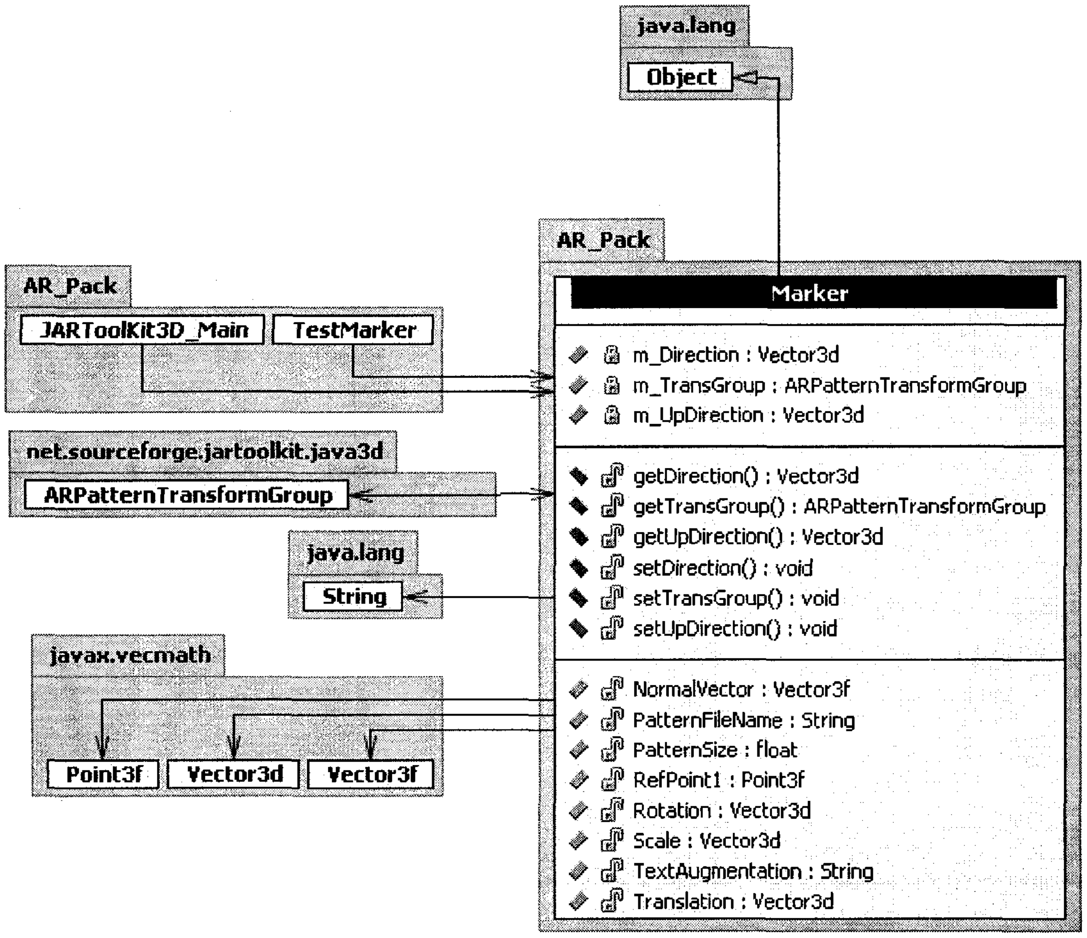


Figure 4.13: Class diagram for Marker

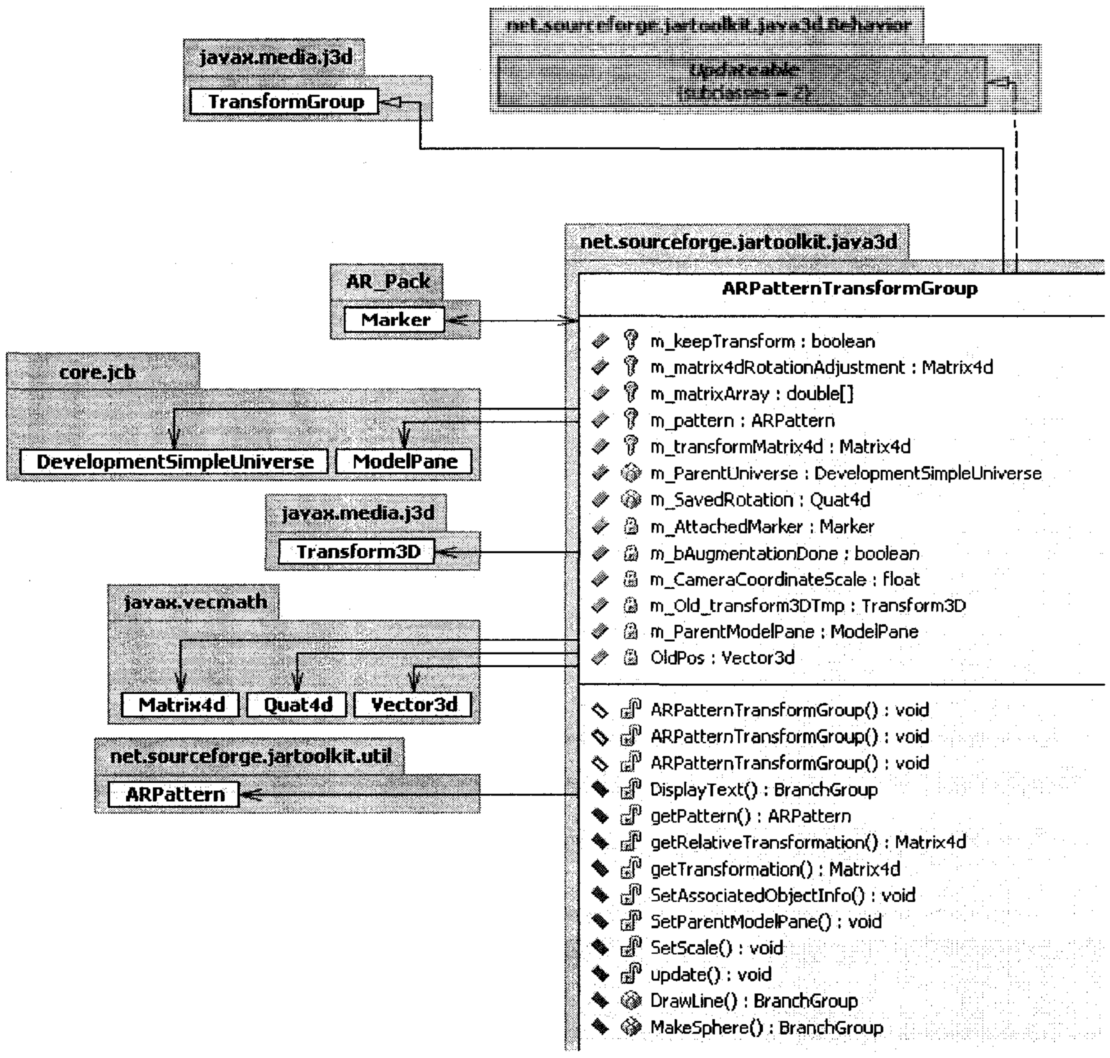

net.sourceforge.jartoolkit.java3d.Behavior
Updateable
(subclasses = 2)
javax.media.j3d
TransformGroup
net.sourceforge.jartoolkit.java3d
ARPatternTransformGroup
AR_Pack
Marker
core.jcb
DevelopmentSimpleUniverse
ModelPane
javax.media.j3d
Transform3D
javax.vecmath
Matrix4d
Quat4d
Vector3d
net.sourceforge.jartoolkit.util
ARPattern
m_keepTransform : boolean
m_matrix4dRotationAdjustment : Matrix4d
m_matrixArray : double[]
m_pattern : ARPattern
m_transformMatrix4d : Matrix4d
m_ParentUniverse : DevelopmentSimpleUniverse
m_SavedRotation : Quat4d
m_AttachedMarker : Marker
m_bAugmentationDone : boolean
m_CameraCoordinateScale : float
m_Old_transform3DTmp : Transform3D
m_ParentModelPane : ModelPane
OldPos : Vector3d
ARPatternTransformGroup() : void
ARPatternTransformGroup() : void
ARPatternTransformGroup() : void
DisplayText() : BranchGroup
getPattern() : ARPattern
getRelativeTransformation() : Matrix4d
getTransformation() : Matrix4d
SetAssociatedObjectInfo() : void
SetParentModelPane() : void
SetScale() : void
update() : void
DrawLine() : BranchGroup
MakeSphere() : BranchGroup

Figure 4.14: Class diagram for ARPatternTransformGroup

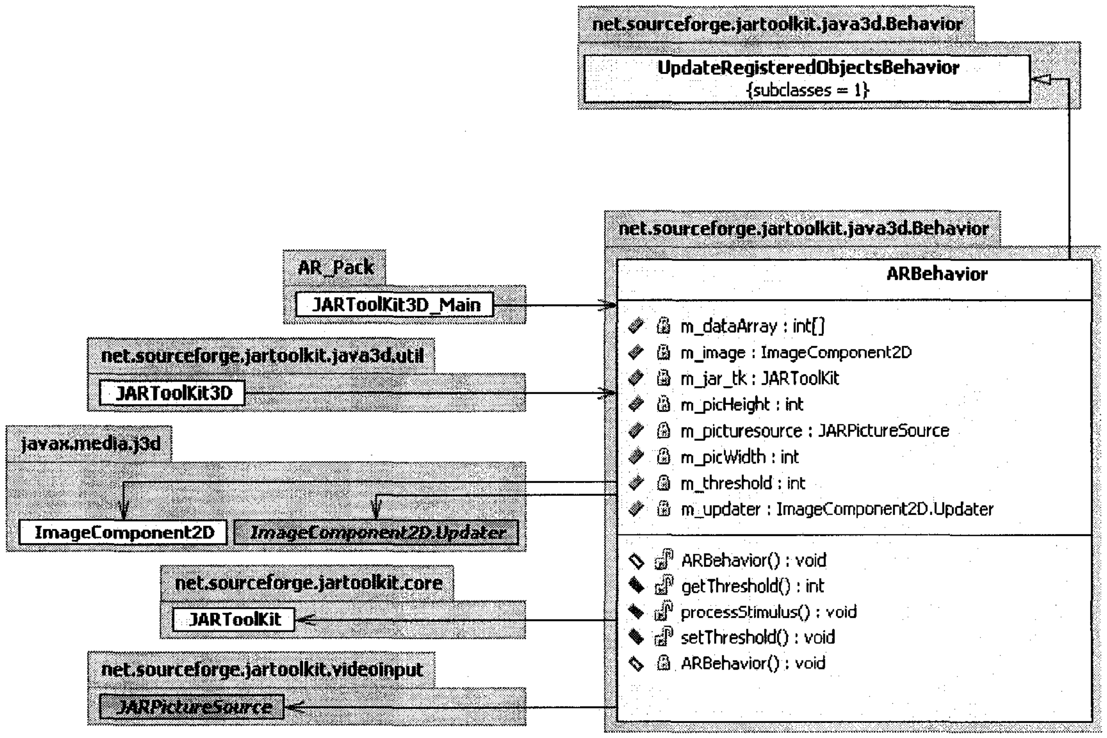


Figure 4.15: Class diagram for ARBehavior

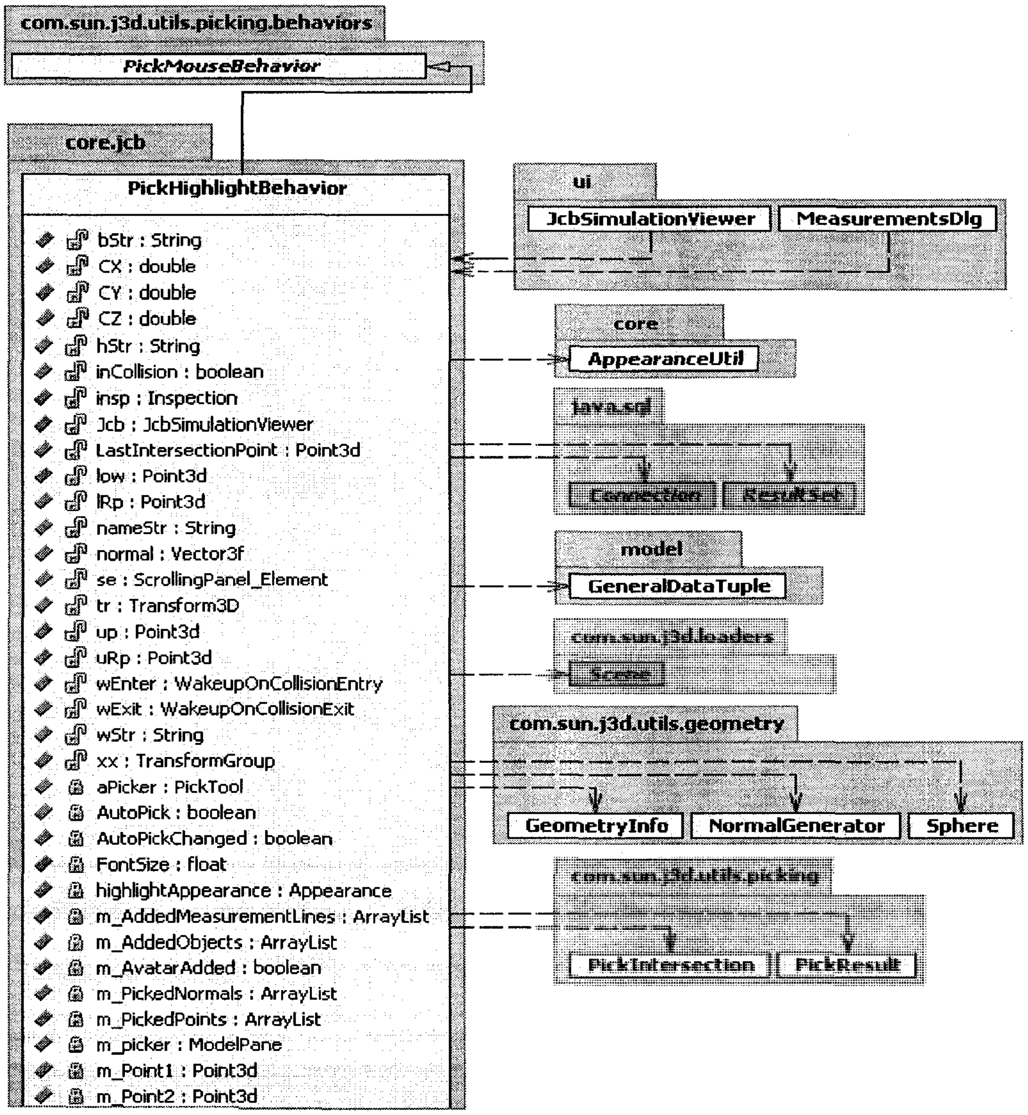


Figure 4.16: Class diagram for PickhighlightBehavior

4.7.2 Design Patterns Used in the Implementation

(1) The Singleton pattern: One of the requirements is that only one sub window for 3D rendering is allowed. For this purpose, the singleton pattern has been used to make the instantiation of the class ModelPane, which creates the 3D canvas, occur only once. The other requirement is that only one instance of ARToolkit is allowed, so the singleton pattern was used to ensure that one instance is created the first time then returned each a request for an instance is received.

(2) The Factory pattern: In the VE mode, the user can select from a list the type of 3D navigation to be used with the mouse. Three types of navigation behavior are provided: *drive*, *fly*, and *orbit*. These types of behavior use the mouse to control the viewpoint motion. Each button on the mouse generates a different type of motion while the button is pressed. The distance of the pointing device location from the center of the display area controls the speed of motion. Since there are common functions that are called for all the behaviors, a parent class (MouseBehavior) has been created from which all mouse navigation behaviors are derived and which contains all common behavior functions. Then a factory class has been created which, in turn, creates one MouseBehavior based on the user selection; and common functions can be called from the returned instance. For example, for all the behaviors a call must be done to the method SetSchedulingBounds which sets the maximum 3D bounding sphere in which the behavior is applicable.

The factory pattern has been created as a decision making class that returns instance of one of the several navigation behavior classes based on the selection provided by the user at run time.

4.8 CASE STUDIES

Case Study 1: Navigation guidance

This case study consists of a situation where a user needs guidance to get to a selected destination and needs information about the rooms in a floor of building. In this case, two scenarios are tested. The first scenario is that a student tries to find the way to emergency exit stairs that are closest to his/her current location. The second scenario is that a student tries to find a room in a building. The case study has been carried out on the fifth floor of the EV building of Concordia University. The student is equipped with a tablet PC, in a back bag, a digital video camera attached to a helmet and connected to the tabled PC, and a video see-through HMD also connected to the tablet PC on which the system runs.

In the first scenario, different paths leading to the different emergency exits in the floor are pre-loaded into the application. Figure 4.17 shows the floor plan and the exit path selected by the system. Also, two locations of the user during the navigation are shown by bold arrows. Two markers have been pasted on the walls of the corridor near the two locations. The edge length of each of the markers is 4 inches. The system starts tracking the movements of the user as soon as a first marker is detected. A path to an exit is defined by a set of predefined 3D point in the world coordinate system. The system computes the closest 3D point to the current user location. While the user is moving towards the exit, the system continuously tracks his/her location. Based on the current user location, the system computes the closest path to an exit. Then, the system displays a 3D directional arrow showing the user the direction to the closest exit. The orientation of 3D directional arrow is updated every time a change in the user location is detected.

Figure 4.18 and Figure 4.19 show examples of the video that the user sees during the navigation. Figure 4.18 (a) shows the view as seen by the user through the HMD when he/she is at Location 1. At this location, the closest exit is on the right of the user; so the directional arrow points towards the right. Figure 4.18(b) shows the view of the virtual scene around the same location with the transparency set to opaque. When the user moves to the right around the Location 2, he/she is in the corridor that leads to the exit, so the directional arrow points ahead to tell the user to keep moving in the same direction as shown in Figure 4.19.

In the second scenario, the student walks in the corridor and looks for the room 9-210. When the student walks near the room, the room number is displayed as a 3D text augmentation is always facing him/her. Figure 4.20 shows two different views when the student is near the door of the room. The first view (a) shows the augmentation in AR mode. The second view (b) shows the same augmentation in VR mode.

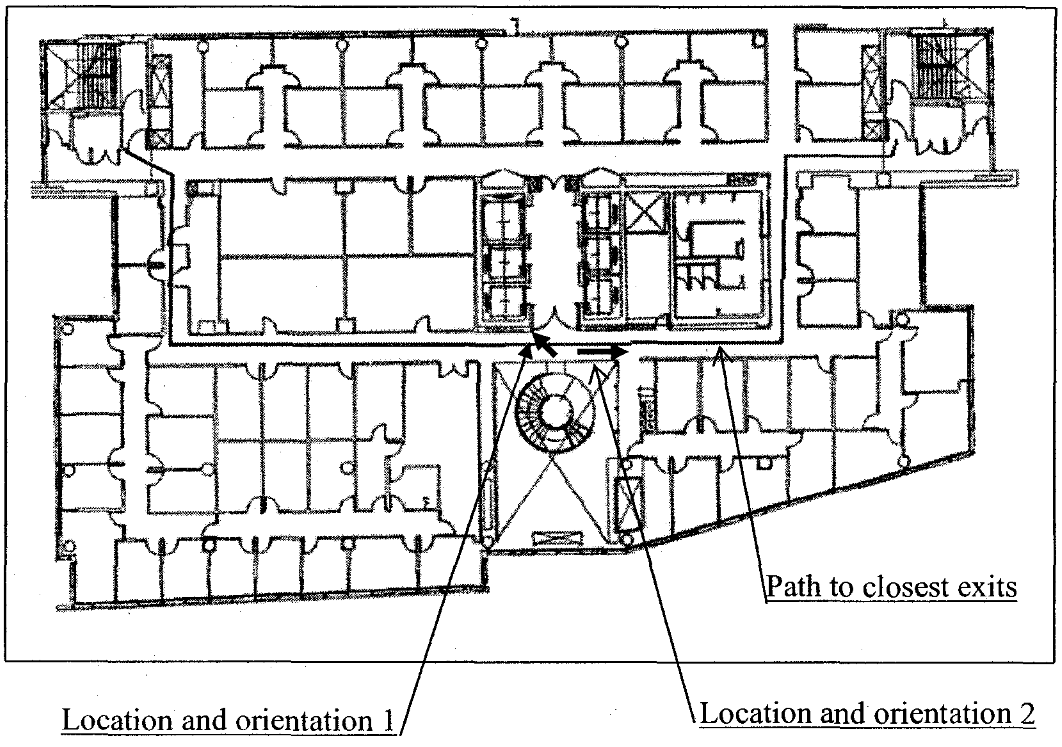
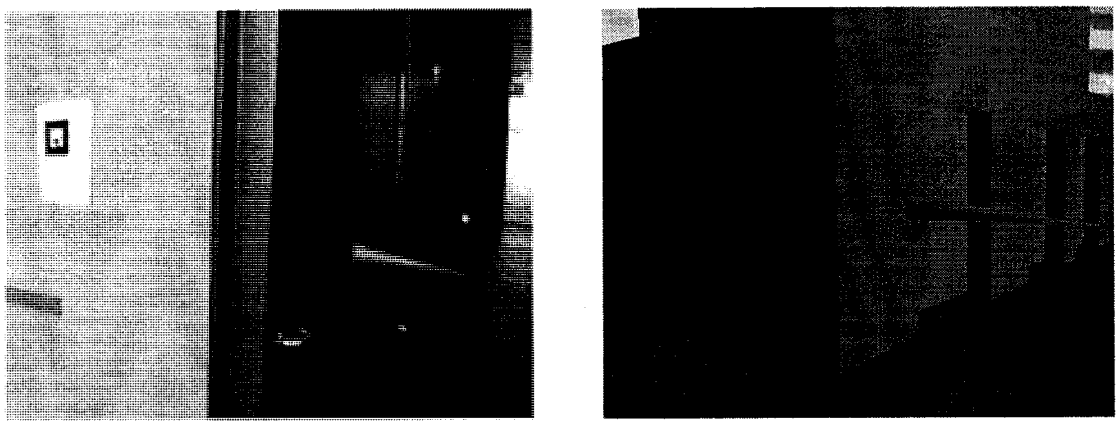


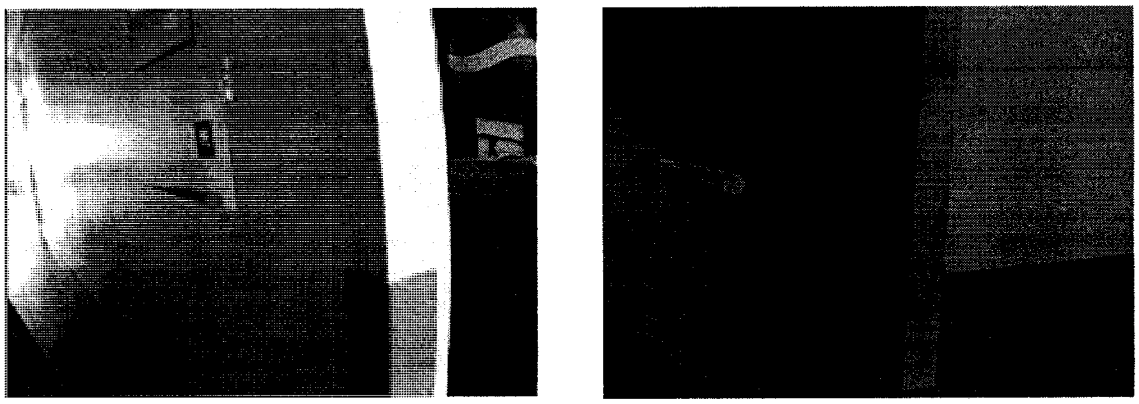
Figure 4.17: Two locations and orientation of the user during navigation



(a) View in AR mode at Location 1

(b) View in VR mode at Location 1

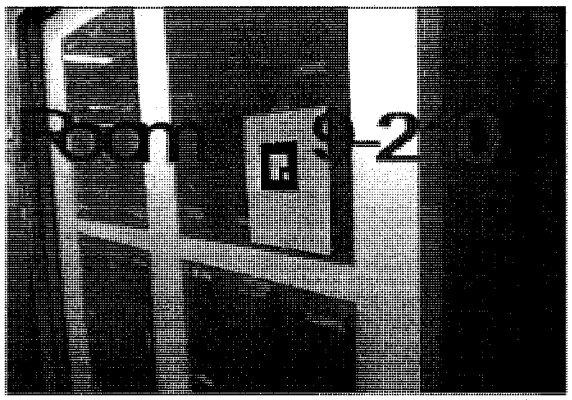
Figure 4.18: AR and VR views as seen by the user at Location 1



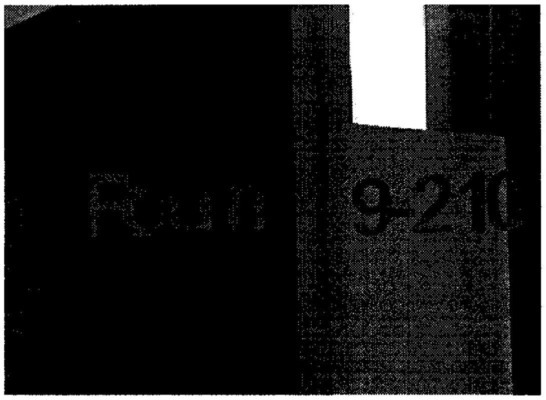
(a) View in AR mode at Location 2

(b) View in VR mode at Location 2

Figure 4.19: AR and VR views as seen by the user at Location 2



(a) Text augmentation in AR mode



(b) Text augmentation in VR mode

Figure 4.20: Text augmentation in AR and VR modes

Case Study 2: AR for HVAC inspection

The 3D virtual model of the EV building in Concordia downtown campus is used as case study, for indoor video-based AR applications. The fifth floor and the HVAC (Heating, Ventilation and Air-conditioning) ducts installed on it have been modelled and loaded into the virtual model of the building.

In this case study, the system is used to conduct an inspection for the HVAC ducts. The inspector uses the system to locate a duct of the HVAC system based on his/her tracked location and his/her task. The system is used to facilitate and speed up the process of finding the HVAC duct that is hidden behind the false ceiling. Instead of guessing or approximating the location of the duct, the inspector just needs to look towards the ceiling to have his/her location detected, and have the ceiling augmented with the 3D virtual model of the HVAC ducts.

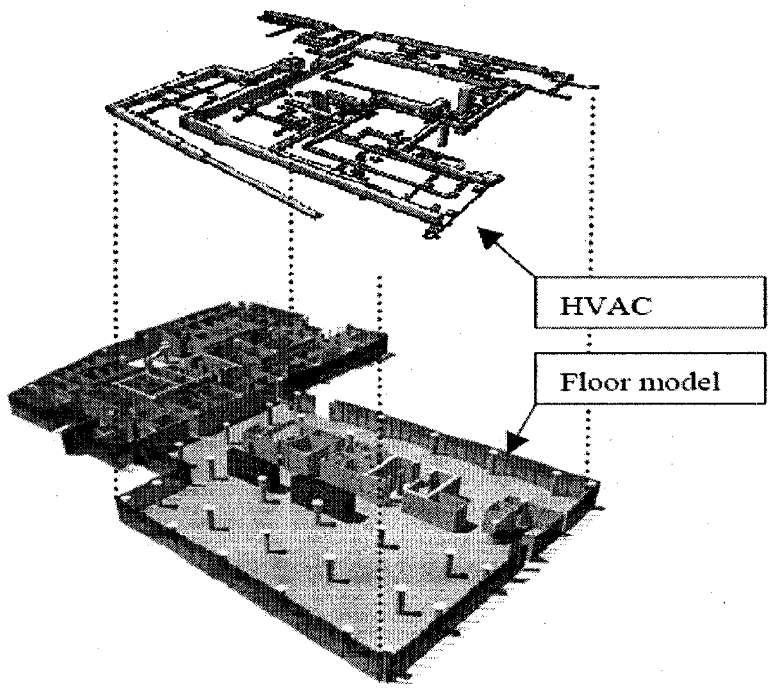


Figure 4.21: Detailed 3D model of one floor and the HVAC ducts

Figure 4.21 shows the virtual models of the fifth floor and the HVAC ducts. Both models have been imported into the system in the format of VRML files. Each element of the models, such as columns and ducts, is associated with an ID that links it to a database that contains more detailed information about that element. The facilities management inspector equipped with the AR devices can perform a routine inspection task for the HVAC system. The 3D HVAC elements (e.g., ducts) are seen through the HMD and the inspector's position and orientation are tracked and used to update the 3D view.

Figure 4.22 shows an example of the results of the AR application used in a facility management project on the campus. This figure shows one virtual HVAC duct in the fifth floor and one marker attached to the ceiling. The figure simulates the view that the inspector can see when the real ceiling is augmented with the virtual model of the HVAC ducts. The inspector will be able to assign defects on the virtual model and retrieve duct element information to be displayed.

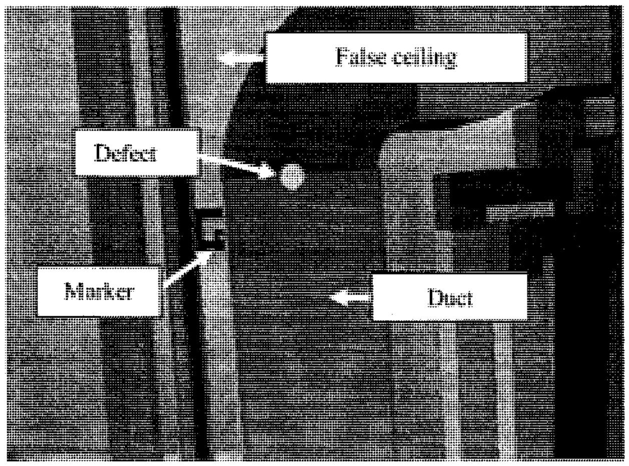


Figure 4.22: The augmentation of the HVAC model as seen by the user

Two types of HMDs were used in this case study: Microvision Nomad ND2000 (2005) and MicroOptical SV-6 (2005). More details about the used hardware can be found in Section 4.9.

Case Study 3: Map augmentation:

The map augmentation is an example of Augmented Virtuality (AV) where a real map augments a 3D model of a city. The map (shown in Figure 4.23) is a paper-based representation of the real world. Since the area covered by the application in this case is small, only one marker has been used.

This augmentation can be used in an urban planning application where several planners and engineers can be sitting around a table and looking at a map laid on it. On top of the map will be displayed a 3D model of the city. The engineers can interactively hide and show objects and buildings in the 3D virtual model.

The main feature that differentiates this case study from the other two cases is the scaling. Since the real world representation on the paper map is scaled, the movement of the user while looking at the map had to be scaled too.

Figure 4.24 shows the different views that the user can see through the HMD while looking at the map. The first view (a) shows a far view of the virtual model of the campus area and the second view (b) shows a closer view that the user can see when looking closer to the map.

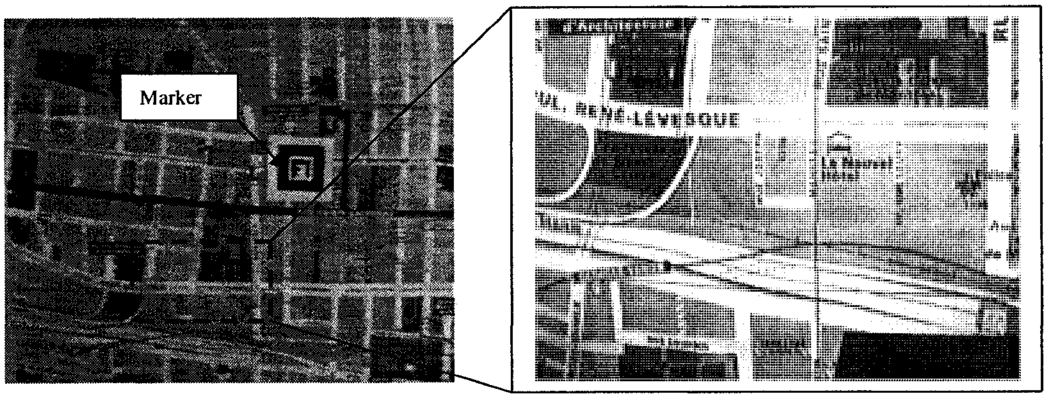
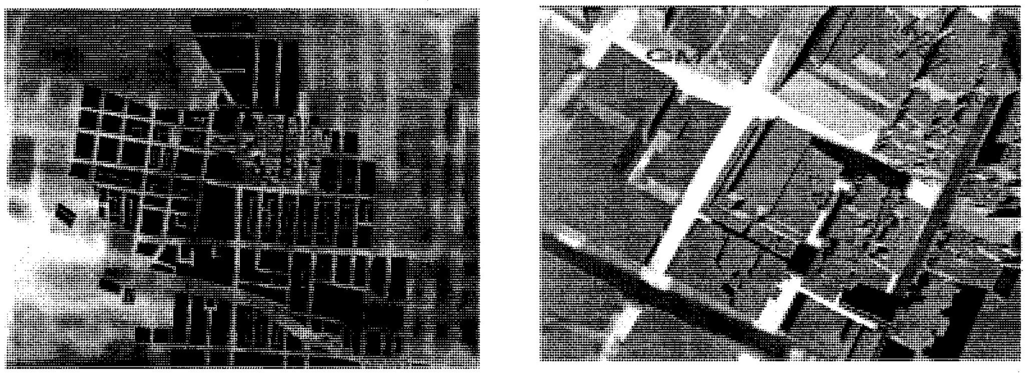


Figure 4.23: Paper map of the campus area



(a) Far AV view (b) Near AV view

Figure 4.24: Example of different AV views as seen by the user

4.9 HARDWARE TESTING

Four digital video cameras were satisfactorily tested with ARToolKit: Logitech QuickCam, Logitech QuickCam Pro, Creative Ultra for Notebooks, and IO-Data USB CCD Camera (Table 4.4). Three types of HMDs were tested: Microvision Nomad ND2000 (Microvision, 2005), MicroOptical SV-6 (MicroOptical, 2005) and I-glasses Video 3D Pro (IO Display Systems, 2006). Microvision Nomad ND2000 has a rugged, monochrome red display (32 gray levels) readable in all lighting conditions with automatic brightness adjustment. MicroOptical SV-6 is smaller, less rugged and has a one-eye color display. I-glasses Video 3D Pro is also less rugged than the first one; it has a two-eye color display and earphones, and is 3D stereo capability. All displays support resolution of 800x600 pixels. Table 4.5 shows the basic specifications of both displays. After testing these displays under different conditions, it was found that the I-glasses is more suitable because of its overall superior visibility using colours and its 3D stereo capability. As for the size of the text displayed with the above resolution, it was found that a minimum font size of 25 points is required for comfortable readability.

Table 4.4: Specifications of test digital video cameras

Make	**Model**	**Satisfactory Resolution and Frame Rate**
Creative	Ulra for Notebooks	640x480 at 30 fps
Logitech	Quickcam	640x480 at 30 fps
Logitech	Quickcam Pro	640x480 at 30 fps
Sony	IO Data USB CCD	320x240 at 30 fps

Table 4.5: Specifications of the HMDs

Make	**Model**	**Transpa r-ency**	**Resolu -tion (pixels)**	**FOV**	**Color**	**Weight (g)**	**Signal**
Microvision	Nomad ND2000	Trans-parent	800x 600	17° horiz. 23° diag.	Mono-chrom e red	128	SVGA
MicroOptical	SV-6	Opaque	800x 600	16° horiz. 20° diag.	Color	35	SVGA
IO Display Systems	I-glasses Video 3D Pro	Opaque	800x 600	27° diag.	Color	226	SVGA

A test has been conducted to check the relationship between the size of a marker and the distance from the video camera (Logitech Quickcam: video resolution of 640x480 at 30 fps) where the marker can be detected and tracked. These ranges were found to be 1.5 m and 3 m for markers with edge lengths of 20 cm and 50 cm, respectively. The effects of light conditions on the tracking were also measured. It was found that the camera can detect the marker even under dark lighting conditions. For example, the marker with the edge length of 50 cm can be detected under luminance equal to or greater than 4 Lux within the range of 3 m.

Table 4.6 shows the detection range tests carried out on two different markers having edge sizes of 20 cm and 50 cm in the same lighting conditions.

Table 4.6: Detection range for different marker sizes

Marker Size	**Distance from Marker**	**Detected (Yes/No)**
20 cm	Less than 1.5 m	Yes
20 cm	Less than 3.0 m	No
50 cm	Less than 1.5 m	Yes
50 cm	Less than 3.0 m	Yes
50 cm	More than 3.0 m	No

4.10 SOFTWARE TESTING

Preliminary tests for the prototype system have been carried out to cover the main functionalities and the main components. The first testing was the Unit testing. It consisted of writing test cases that are carried out inside the development environment to ensure that different methods of the main classes behave correctly and return the right results. Next, requirement tests have been carried out to test the main features of the system and make sure they are correctly implemented.

On the other hand, usability testing can be also carried out on the system but since the system is still a prototype, no usability testing has been performed at this stage. Nevertheless, Appendix F contains some usability testing guidelines and heuristics for AR applications that can be used for future testing of the system.

4.10.1 Unit Testing

The unit testing has been carried using Junit, which is a testing tool integrated inside the selected development environment (JBuilder). Several test cases have been built and executed to make sure the main classes in the implemented system are working properly. The detailed description of the different test cases can be found in Appendix E.

Class AnimatedArrow (Table C.1)

- **testCreateArrow():** This test method ensures that there is always an arrow object created when the function CreateArrow is called.

  Reason for testing: Because visual testing of the arrow cannot be performed unless the arrow object exists.

- **testOrientationAlgorithm():** This test method ensures that the algorithm used to orient the arrow to the destination location is correct.

  Reason for testing: Because the orientation of the arrow is crucial for getting the right direction to the destination point.

Class Marker (Table C.2)

All of the following test methods ensure that the properties saved for a paper marker after being loaded from a marker configuration file are correctly retrieved when they are needed.

Reason for testing: Because if any of the marker properties are not correctly retrieved, the tracking of the user in AR mode will not be reliable.

- testGetDirection()
- testGetNormalVector ()
- testGetPatternFileName()
- testGetPatternSize()
- testGetRefPoint()
- testGetRotation()
- testGetScale()
- testGetTextAugmentation()
- testGetTranslation()
- testSetUpDirection()

Class JARToolKit3D_Main (Table C.3)

- testGetInstance(): This test method ensures that there is always an instance of the class returned.

  Reason for testing: Because it is required by the used toolkit "ARToolkit" to create a singleton instance of it.

- testGetInstance_Null_ModelPane_Hanlding(): This method tests exception handling in case the parent window is passed to the class method (GetInstance).

  Reason for testing: Because the application must not crash or cause any serious problem when it fails to return an instance.

- **testMarkersLoadingFromNullFile():** This method tests exception handling when an empty file name is passed to the class method LoadMarkersFromFile

  Reason for testing: Because the application must not crash or cause any serious problem when no file name is provided.

- **testMarkersLoadingFromFile():** This method ensures that all the markers in the marker configuration file are loaded.

  Reason for testing: Because it is necessary to have all the markers in the configuration file loaded before starting the tracking in AR Mode.

- **testMarkerStoredInfo():** This method ensures that the markers' properties stored in the configuration file are corrected loaded into memory and stored in the instances of the class Marker.

  Reason for testing: Because if any of the properties stored in the configuration file is not properly loaded into memory, the tracking in AR mode will not be correct and reliable.

- **testMarkerOutOfRangeIndex():** This method tests the exception handling in case a wrong index of the marker is passed. The marker index has to be within the range of the markers loaded from the configuration file.

  Reason for testing: Because if a wrong index of marker is passed the application must not crash and cause any serious problem for the user.

- **testSetTrackingScale():** This method makes sure the tracking scale is set for all the loaded markers.

  Reason for testing: Because if the tracking scale is not set for one marker, the tracking in AR mode will be inconsistent.

Class JARToolKit3D (Table C.4)

Reason for testing: Because if any of the marker properties are not corrected retrieved the tracking of the user in AR mode will not be reliable.

- **testGetInstance()**
- **testCreate()**
- **testInitializeCamera()**
- **testInitializeCameraWrongFileName()**
- **testInitializeCameraWrongPath()**
- **testCreateBackground()**

4.10.2 Requirements Testing

<u>**Feature 1:**</u> Develop animated 3D arrow oriented from current user's location to the object of interest.

Requirments: These requirment testing had to be performed visually following system usage scenerios (Table D.1).

- The orientation of the arrow should be always adjusted based on the user location: It is necessary for the user who is navigating in the virtual and real environment to be able to locate the destination from anywhere in the space.
- The arrow should be visible in AR mode: The system is designed to be used in AR mode in addition to the VR mode.

<u>**Feature 2**</u>: Track and record the current user locations in VR mode by developping a behavior that repeatedly calulates the abosolute user location (Table D.2).

Since the application is intended to be use in mobile and location-based situations in AR mode, it was necessary to, first, test the tracking of the user location in VR mode before AR mode. To test this feature, testing method has been used based on Geographic Information System (GIS) since the VE is based on real locations. In this test, the location of the users at every time step t_i while navigating are recorded as points in a GIS layer. Later this layer can be visualized to make sure his/her navigation in the was tracked correctly.

Requirments:

- The tracking technology should be provided in VR Mode: It is necessary to track the user navigation to be able to provide guidance through the 3D oriented arrow.

- A behavior should be developed to repeatedly move the arrow to current user location. It is necessary to get updated guidance while the user is moving.

Feature 3: Track the current user location in AR mode using the video tracking (Table D.3).

Requirements:

- Detect the user's movement with respect to the paper markers: This is necessary because the relative user location with respect the marker will be added to the stored absolute location of the marker.
- Calculate the 3D coordinate representing the user's location in the real world: This is needed to compute the final absolute location of the user in real world thus being able to track him/her.

Feature 4: Overlay the 3D animated arrow on the video of real object displayed using the HMD (Table D.4).

Since our application provides the user guidance using an animated arrow in AR mode, this feature should be tested. We tested this feature visually by using our testing system. A sphere is added by default to the location of a marker. If the feature works coorectly, the sphere should be visible when the marker is detected by camera. In addition, an oriented arrow should be added to location of the sphere.

Requirements:

- Add an arrow to current location pointing to object of interest: The system is designed to guide the user in AR mode as well as in VR mode.

Feature 5: Overlay a text augmentation on the video of real objects showing the recognized room number in the building (Table D.5).

This feature is usefull to give information about the environment to the user in AR mode so this feature should be tested. The testing process of this feature is similar to the previous one. However, a text showing the room number should be added to the same location of the sphere.

Requirements:

- Overlay a 3D text on the video showing the room number: This is needed to ensure that the system shows correct information about the environment.

Feature 6: Control the field of view of VE in AR Mode (Table D.6).

Requirements: In order to test this feature, the actual field of view can be compared wih the video field of view. It is important to test this feature because it adjusts the field of view of the virtual scene so that the user can match it with the field of view of the digital video camera.

Feature 7: Control the transparency of the virtual objects in the virtual scene (Table D.7).

Requirements: The user should be able to change the transparency level of the virtual objects in the scene to be able to completely hide them or show them partially transparent of completely opaque on top of the video frames.

Feature 8: Control the video-tracking scale (Table D.8).

Requirements: The user needs to be able to move the camera that is tracking the paper marker and change the value of the tracking scale. The movement should be scaled based on the selected scale.

4.11 SUMMARY AND CONCLUSIONS

In this chapter a prototype system has been implemented to demonstrate the feasibility of the proposed approach. The system has been intended to be platform independent so all of its components were implemented in the Java programming language and for some of the tools Java binding to APIs were used. Different interaction methods have been implemented to allow for a direct and location-based access to information about elements of different structures, such building rooms, through the 3D picking tool. The prototype system implemented a hands-free navigation through the video tracking of geo-referenced markers. In addition, a way-finding assistance has been implemented to help the user find the path to a selected destination while moving in the real and virtual world. On the other hand, the video tracking configuration and deployment method has been implemented and verified through cases studies. These case studies have also been used to verify the applicability and usefulness of the proposed approach. While the system has been proven applicable and useful, due to the limitation in the different hardware capabilities such as the video resolution, the frame rate and the field of view, the virtual objects rendered as augmentation appear with a slight mis-alignment with the real world. This mis-alignment error is a combination of the inaccuracy of the virtual model, tracking errors caused by ARToolkit, and the hardware limitations.

CHAPTER 5 SUMMARY, CONCLUSIONS, AND FUTURE WORK

5.1 SUMMARY

AR has been proven to be useful in many fields as mentioned in the Chapter 2, and several tools and systems have been reviewed and examined. Taking into consideration the different shortcomings of the available AR systems, a framework for indoor video-based AR application has been proposed to integrate four main components of AR applications, which are large scale VEs, mobile devices, interaction methods and video-tracking, in one system. The proposed framework benefits from the 3D virtual modeling by combining GIS maps and models of buildings and building interiors in one system. Different interaction methods for AR applications have been introduced, such as the automatic 3D picking which is used for a location-based data access. In addition, a practical method has been proposed for the configuration and the deployment of the video tracking. This method made use of the XML language to allow for future extensions and simplified interoperability. An implementation of the proposed framework has been developed to demonstrate its feasibility. Different case studies have been carried out to validate the applicability of the system and identify its benefits and limitations. The first case study consisted of an AR navigation assistance using a directional 3D arrow and 3D text to help the user find a selected destination inside a building. The second case study demonstrated how the system can be used to help an HVAC inspector find one of the ducts and mark on it a 3D shape to indicate the location of a problem. The last case study illustrated the use of the system in adding scaled augmentation that fits on a paper map of the Concordia Campus area. It also demonstrated the use of scaling during video tracking to make it fit the map scale.

5.2 CONCLUSIONS AND CONTRIBUTIONS

The proposed framework is characterized by its applicability to several fields, which is demonstrated by the case studies for navigation assistance, building inspection, and urban planning. The framework has the advantage of using real data, such as GIS maps, and integrating them in the same coordinate system as 3D virtual models defined in a standard file format (VRML) which allows for an easy exchange of data and more extensibility. In addition, the proposed video-tracking system is characterized by its affordability and ease of use. Moreover, using Java and Java3D object-oriented API, other tracking capabilities and tools can be easily integrated into the system thanks to the built-in Virtual Universe concept and Behavior, which is an object-oriented and extended type of multi-threading.

The contributions of this research are grouped into the following areas:

(1) A GIS-based approach for building VR models for AR applications has been developed. The approach integrates large scale 3D virtual models of buildings along with detailed interior 3D virtual models in one single system. All 3D virtual models are represented in one single coordinate system that represents the real world coordinate system based on GIS.

(2) A framework has been developed for indoor video-based AR applications to integrate four components: (1) Large scale 3D virtual models, (2) Mobile devices, (3) Video tracking, and (4) Interaction.

(3) A practical and simplified method has been developed and implemented for the configuration and deployment of video tracking in indoor AR applications.

(4) Different interaction methods for indoor AR applications have been developed, such as the automatic picking behavior.

(5) A prototype system has been developed and implemented in Java, and three case studies have been used to demonstrate the feasibility of the above mentioned approaches and methods. The first case study demonstrated navigation assistance; the second case study demonstrated HVAC inspection; and the third case study demonstrated urban planning.

On the other hand, the implemented prototype of the system is limited by the field of view of the used video cameras; this limitation can be overcome, in the future, as mentioned in Section 5.3, using spherical cameras. The system is also limited by the possible discontinuity of the video-tracking due to the limited visibility range of the different markers; this limitation can be reduced by using motion-tracking devices such as gyroscopes. The combination of the hardware limitations, the inaccuracy of the virtual model, and the tracking errors caused by ARToolkit, produced slight mis-alignment of the virtual augmentation with the real world. Reduced environment awareness of the users is another limitation imposed by the use of video-based HMD.

5.3 FUTURE WORK

(1) Since the field of view of digital video cameras is crucial in the video-based detection of markers, spherical cameras, such as the Ladybug (Point Grey website, 2008), can be used in future works to make the system more aware of the surroundings and get a robust tracking which makes the system safer for the user. Appendix F explains the possible extension of the system to seamlessly link it with the Ladybug spherical camera.

(2) Through the networking capability of the system, the user can be lively notified of an emergency. The system starts guiding the user to the closest exit and keeps him/her updated with the latest information about the emergency. The user can also dynamically change the exit plan based on the changes in the circumstances such as the collapse of a floor that makes the selected exit impossible or more risky. A similar capability has been introduced by the Chicago Fire Department to help the firemen collaborate during firefighting operations and get live notification about the changes in the surrounding conditions, such as room temperature (Fire Project website, 2008).

(3) The tracking component of the system can be extended by other types of tracking such the Radio Frequency IDs (RFID to detect different hazardous places in a building and warn the user before entering in their perimeter.

(4) A new path finding technique inside buildings can be used to dynamically select the closest path to the target.

(5) Head tracking devices can be used to extend the indoor tracking by filling the gaps between the visibility ranges of the different markers.

(6) Developing methods to reduce the errors discussed in the previous section.

REFERENCES

AMIRE project website (2007). <www.amire.net>, (accessed July, 2007).

ARPE: Augmented Reality Prototyping for Entertainment Group (2003). Carnegie Mellon University - Entertainment Technology Center, <www.etc.cmu.edu/projects/ar/argt.html>, (accessed in 2006).

ARToolkit website (2005). <www.hitl.washington.edu/artoolkit>, (accessed 2005).

Azuma, R. (1997). Survey of Augmented Reality, Presence: Teleoperators and Virtual Reality, 6(4), pp. 355-386.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). Recent Advances in Augmented Reality, IEEE Computer Graphics and Animation, Nov/Dec 2001, pp. 34-47.

Behzadan, A., and Kamat, V. (2006). Animation of Construction Activities in Outdoor Augmented Reality, Joint International Conference on Computing and Decision Making in Civil and Building Engineering 2006 – Montreal, Canada.

Brian, G. (2004). Techniques for Assessing and Improving Performance in Navigation and Wayfinding using Mobile Augmented Reality, Ph.D. thesis, University of Central Florida, pp. 2-60.

Brunner, S., Bimber, O., and Mader, S. (2003). Report on Tracking Technology, Virtual Showcases Project, The European Commission – Information Society Technologies, pp. 10-15.

Charles O., Fan X., and Paul M. (2002). What is the best fiducial? The First IEEE International Augmented Reality Toolkit Workshop, September 2002.

Chung, K. (2002). Application of Augmented Reality to Dimensional and Geometric Inspection, Ph.D. thesis, Virginia Polytechnic Institute, pp. 35-45.

Christoph, K., and Bludau, HB. (2002). Applicability of Handheld Computers in Clinical Information Systems: Comparison of Evaluation Methods. Second Conference on Mobile Computing in Medicine, 2002, pp. 51-61.

Coelho, E. (2005). Spatially Adaptive Augmented Reality, Ph.D. Thesis, Georgia Institute of Technology, pp. 7-9.

Drasic, D. and Milgram, P. (1996). Perceptual Issues in Augmented Reality, SPIE Volume 2653: Stereoscopic Displays and Virtual Reality Systems III, San Jose, California, USA, January/February 1996, pp 123-134.

Dunston, P.; Wang, X., Billinghusrt, M., and Hampson, B. (2002). Mixed Reality Benefits for Design Perception. 19th International Symposium on Automation and Robotics Construction (ISARC 2002), NIST, Gaithersburg, MD, 2002, pp. 191-196.

Fiala, M. (2004). ARTag Website, <www.artag.net>, (accessed June, 2006).

Fire Project website (2008). < fire.me.berkeley.edu>, (accessed in 2008).

Frund, J., Gausemeier J., Matysczok, C., and Radkowski, R. (2003). Application Areas of AR-Technology within Automobile Advance Development, International Workshop on Potential Industrial Applications of Mixed and Augmented Reality, 2003.

Gabbard, J. (2001). Researching Usability Design and Evaluation Guidelines for Augmented Reality Systemswww.sv.vt.edu/classes/ESM4714/Student_Proj/class00/gabbard>, (accessed in 2005).Gediga, G., Hamborg, K., and Duntsch, I. (1999). The IsoMetrics Usability Inventory: an operationalization of ISO 9241-10 supporting summative and formative evaluation of software systems.

Gerhard R., G., and Schmalstieg, D. (2004). Collaborative Augmented Reality for Outdoor Navigation and Information Browsing. Proceedings of the Symposium on Location Based Services and TeleCartography, Vienna, Austria, pp. 31-41.

Ph.D. Thesis,Halden Virtual Reality Center website (2006). <www2.hrp.no/vr>, (accessed in 2006).

Hammd, A., Khabeer, B., Mozaffari, E., Devarakonda, P., and Bauchkar, P. (2005). Augmented Reality Interaction Model for Mobile Infrastructure Management Systems, 1st CSCE Specialty Conference on Infrastructure Technologies, Management and Policy.

Hoffmann, E. (2001). The 3d Studio file format library, <lib3ds.sourceforge.net>, (accessed in 2005)

IO Display Systems Website (2006), </www.i-glassesstore.com>, (accessed in 2006).

ISO 9241-11, (1998). Guidance on Usability.

Kaiser, E., Olwal, A., McGee, D., Benko, H., Corradini, A., Li, X., Cohen, P., and Feiner, S. (2003). Mutual Disambiguation of 3D Multimodal Interaction in Augmented and Virtual Reality. Proceedings of the International Conference on Multimodal Interfaces'03, November 2003.

Kato, H., Billinghurst, M., and Poupyrev, I. (1999). ARToolKit version 2.33: A Software Library for AR Applications, <www.hitl.washington.edu/artoolkit>, (accessed December, 2004).

Ledermann, F., Reitmayr, G., and Schmalstieg, D. (2002). Dynamically Shared Optical Tracking, The First IEEE International Augmented Reality Toolkit Workshop, September 2002.

Liarokapis, F. (2006). An Exploration from Virtual to Augmented Reality Gaming, Simulation & Gaming, Vol. 37, pp. 507-533.

Liarokapis, F., White, M., and Lister, P. (2004). Augmented Reality Interface Toolkit, Proceedings of the International Symposium on Augmented and Virtual Reality, pp. 761-767.

Livingston, M., Rosenblum, L., Julier, S., Brown, D., Baillot, Y., Swan, E., Gabbard J., and Hix, D. (2002). An Augmented Reality System for Military Operations in Urban Terrain. Proceedings of the Interservice / Industry Training, Simulation, & Education Conference (I/ITSEC '02), December 2002.

Livingston, M. (1998). Video-based Tracking with Dynamic Structured Light for Video See-through Augmented Reality, Ph.D. Thesis, University of North Carolina, pp. 30-31 .

Macchiearella, N. (2004). Effectiveness of Video-Based Augmented Reality as a Learning Paradigm for Aerospace Maintenance Training, Digital Avionics Systems Conference, 2004, pp. 5.1-9.

MicroOptical website (2005). <www.microopticalcorp.com>, (accessed in 2005).

Middlin, P. (2002). Vision-based Tracking of Fiducials for Augmented Reality, Master of Science Thesis, Michigan State University, pp. 6-7.

Milgram, P., and Kishino, F. (1994). Taxonomy of Mixed Reality Visual Displays. IEICE Trans. on Information and Systems, Vol. E77-D, pp. 1321-1329.

Milgram, P., Rastogi and A., Grodski, J. (1995). Telerobotic Control Using Augmented Reality, University of Toronto - Ergonomics in Teleoperation and Control Lab, <vered.rose.utoronto.ca/people/paul_dir/RO-MAN95/roman95.html>, (accessed April, 2007)

Mozaffari, E. (2006). Creating and Testing Urban Virtual Reality Models for Engineering Applications, Master of Science Thesis, Concordia University, pp. 28-29.

MXRToolkit website (2005), Singapore Mixed Reality Lab, <mxrtoolkit.sourceforge.net>, (accessed in 2006).

Nielsen, J. (1994). Usability Engineering, 1st edition, San Francisco, CA: Morgan Kaufmann, pp. 24-25.

Nielson, J. (1995). www.useit.com/papers>, (accessed 2007).

Nomad Display Systems (2005). Microvision website, <www.microvision.com>, (accessed in 2005).

Pierre, M., Wayne, P., and Bruce, HT. (2002). Measuring ARToolKit Accuracy in Long Distance Tracking Experiments, The First IEEE International Augmented Reality Toolkit Workshop, September 2002.

Point Grey Research website (2008). < www.ptgrey.com>, (accessed in 2008).

Romao, T., Correia, N., Dias, E., Danado, J., Trabucob, A., Santosb, C., Santosb, R., Camara, A., and Nobre, E. (2004). ANTS-Augmented Environments, Computers & Graphics, pp. 625–633.

Ravden, S., and Johnson, G. (1989). Evaluating usability of human-computer interfaces – a practical method, First Edition, Halsted Press, pp.19-30.

Rekimoto, J. (1998). Matrix: A Realtime Object Identification and Registration Method for Augmented Reality, Proceedings of Asia Pacific Computer Human Interaction , 1998, p. 63.

Rekimoto, J., and Ayatsuka, Y. (2000). CyberCode: Designing Augmented Reality Environments with Visual Tags, Proceedings of Designing Augmented Reality Environments, 2000. pp. 1-10.

Sands, J., Lawson, S., and Benyon, D. (2004). Do we need Stereoscopic displays for 3D Augmented Reality Target Selection Tasks? , Proceeding of the Eighth International Conference on Information Visualisation, 2004, pp. 633-638.

Shiratuddin, M., Thabet, W., and Bowman, D. (2004). Evaluating the Effectiveness of Virtual Environment Displays for reviewing Construction 3D Models, Virginia Tech University, < http://research.cs.vt.edu/3di/>, (accessed July 2007).

Slay, H., Phillips, M., Vernik, R., and Thomas, B. (2001). Interaction Modes for Augmented Reality Visualization, Proceedings of the Asia-Pacific symposium on Information visualization, 2001, pp. 71-75.

Tatham, E. (1997). Depth Cueing for Augmented Reality, Proceedings of IEEE Conference on Information Visualization, 1997, pp. 348-349.

The Usability Company (2004). What's Usability, <www.theusabilitycompany.com>, (accessed in 2006).

Vallino, J. (1998). Interactive Augmented Reality, Ph.D. Thesis, University of Rochester, pp. 16-17.

Virtual Reality Laboratory website (2006). Virtual Reality: a Short Introduction, Virtual Reality Laboratory, University of Michigan, <www-VRL.umich.edu>, (accessed 2007).

Wang, X., and Dunston, P. (2004). Application of Mixed Reality to A/E/C (Architecture/Engineering/Construction): Taxonomy, Prototype, and Evaluation. Proceedings of ASCE Construction Research Council Research Symposium, American Society of Civil Engineers (ASCE), November 2004, pp. 133-140.

Appendix A: XML files for describing markers

1. Example of a Marker Library XML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<MarkerLibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\\BridgeResources\\Markers.xsd">

<Marker>
<PatternFileName>C:/BridgeResources/JARToolkitTestFolder/data/pattern/4
x4_384_47.patt</PatternFileName>
            <LinkID>5125</>
            <PatternSideDimension> 115 </>
            <AbosoluteLocationInWorld>
                  <x> 298700.412</x>
                  <y> 72.689</y>
                  <z> -5039511.305</z>
            </AbosoluteLocationInWorld>
            <NormalVector>
                  <x> -0.836 </x>
                  <y> 0.0 </y>
                  <z> -0.548 </z>
            </NormalVector>
            <UpDirection>
                  <x> 0 </x>
                  <y> 1 </y>
                  <z> 0 </z>
            </UpDirection>
            <DirectionVector>
                  <x> 0.0</x>
                  <y> 1.0</y>
                  <z> 0.0</z>
            </DirectionVector>
            <Translation>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Translation>
            <Rotation>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Rotation>
            <Scale>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Scale>
</Marker>
```

<Marker>
<PatternFileName>C:/BridgeResources/JARToolkitTestFolder/data/pattern/4
x4_384_91.patt</PatternFileName>
        <LinkID>ElevatorCorner</>
        <PatternSideDimension> 115 </>
        <AbosoluteLocationInWorld>
                <x> 298696.60836248304 </x>
                <y> 73.15966358270978 </y>
                <z> -5039515.23124213 </z>
        </AbosoluteLocationInWorld>
        <NormalVector>
                <x> 0.0</x>
                <y> 0.0</y>
                <z> 1.0</z>
        </NormalVector>
        <UpDirection>
                <x> 0.0 </x>
                <y> 1.0 </y>
                <z> 0.0 </z>
        </UpDirection>
        <DirectionVector>
                <x> 0.0 </x>
                <y> 1.0 </y>
                <z> 0.0 </z>
        </DirectionVector>
        <Translation>
                <x> 0.0 </x>
                <y> 0.0 </y>
                <z> 0.0 </z>
        </Translation>
        <Rotation>
                <x> 0.0 </x>
                <y> 0.0 </y>
                <z> 0.0 </z>
        </Rotation>
        <Scale>
                <x> 0.0 </x>
                <y> 0.0 </y>
                <z> 0.0 </z>
        </Scale>
</Marker>

<Marker>
<PatternFileName>C:/BridgeResources/JARToolkitTestFolder/data/pattern/4
x4_384_83.patt</PatternFileName>
                <LinkID>5125</>
                <PatternSideDimension> 115 </>
                <AbosoluteLocationInWorld>
                        <x> 298702.487</x>
                        <y> 72.767</y>
                        <z> -5039506.637</z>
                </AbosoluteLocationInWorld>
                <NormalVector>
                        <x> 0.546 </x>
                        <y> 0.0 </y>
                        <z> -0.837</z>
                </NormalVector>

```
            <UpDirection>
                  <x> 0 </x>
                  <y> 1 </y>
                  <z> 0 </z>
            </UpDirection>
            <DirectionVector>
                  <x> 0.0</x>
                  <y> 1.0</y>
                  <z> 0.0</z>
            </DirectionVector>
            <Translation>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Translation>
            <Rotation>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Rotation>
            <Scale>
                  <x> 0.0 </x>
                  <y> 0.0 </y>
                  <z> 0.0 </z>
            </Scale>
```

</Marker>
</MarkerLibrary>

Appendix B: Software requirements and installation guide of the prototype system

Software requirements:

(1) Borland JBuilder 2006 Enterprise: used to develop the prototype system;

(2) MS Access (MS Access XP): used to store the data of the buildings;

(3) ArcGIS (ESRI 2004): used to develop GIS application;

(4) Windows XP: used as the operation system.

Installation guide:

1. Copy four folders to corresponding driver and change the associated code in the project to match the driver path. The contents in these folders include:

   - *Currproject* or *infra_project* (The folder includes all codes of our project)
   - *Javasoft* (The folder includes all libraries which are required in our project)
   - *Bridgere* (The folder includes all 2d information)
   - *BridgeResources* (The folder includes all 3d information and models)

2. Click Start->Control Panel-> Administrative Tools->Database Source.

   And add ODBC data source as below:

   - Microsoft Access Driver: Name: bridge, Location: C:\ BridgeResources\db1.mdb

   (No password for data source is required, so just leave the password as blank.)

3. Launch Jbuilder, open the project.jpx file. Then click the menu of Jbuilder: Project->Project Properties. Click the tab "Required Libraries". Then edit or add the path of libraries as below:

- *VRML97 (VRML File Loader API)*

  Download the library from: https://j3d-vrml97.dev.java.net

  D:/javasoft/loaders/vrml97.jarVRML97 library

- *Jdk3D (Java development Kit 3D) - Version 1.3.1*

  Download from http://java.sun.com/products/java-media/3D/downloads/index.htmlJRE/1.3.1_09/lib/ext

- *DXFLoader (DXF file loader API) - Version 1.0* is needed.

  Download from: http://www.johannes-raida.de/index.htm?cadviewer

- JARToolkit Dlls: The dlls are located at D:\JavaSoft\JARToolkit Dlls

  Make sure you put these dll files in your JARToolkit Dlls directory, then click start-> Control Panel->System->Advanced->Environment Variables, please add the path of JARToolkit Dlls directory to the environment variable "path" of User Variables.

- *MOJ (MapObject Java API) - Version 2.1*

  Download from: http://www.esri.com/software/mojava/

  Put the files in: C:/ESRI/MOJ21/lib add the tutsource.jar and tutorial.jar files that are available at the directory MOJ21\Samples\Tutorial.

- *JARToolkit (Java ARToolkit API) - Version 2.0*

  Download from: http://jerry.c-lab.de/jartoolkit/

  The needed files are: JARFrameGrabber.dll, JARToolkit.dll, JARVideo.dll, libARvideo.dll, libARvideod.dll, and msvcr70.dll

Table B.1: Summary of libraries used in the prototype system

Library	Description	Source	Version
VRML97	VRML File Loader API	**https://j3d-vrml97.dev.java.net/** **D:/javasoft/loaders/vrml97.jar**	
JDK 3D	Java 3D API	**http://java.sun.com/products/javamedia/3D/downloads/index.html** **D:/javasoft/ JRE/1.3.1_09/lib/ext**	1.3.1
DXFLoader	DXF File Loader API	**http://www.johannes-raida.de/index.htm?cadviewer** **D:/Javasoft/DxFloader/**	1.0
MOJ	MapObject Java API	**http://www.esri.com/software/mojava/**	2.1
ARToolkit	ARToolkit API	**http://www.hitl.washington.edu/artoolkit/**	
JARToolKit	Java Binding for ARToolKit API	**http://jerry.c-lab.de/jartoolkit/** **D:\JavaSoft\JARToolkit Dlls**	2.0

Appendix C: Unit Testing

The following tables list the different test cases done in this study. Each table shows the class name and the methods that have been tested.

Table C.1: Test cases for AnimatedArrow

Test Case	**Test Data**	**Expected Result**	**Traceability**
testCreateArrow()	Null	Arrow object	TR1
testOrientationAlgorithem()	Point3f(0,0,0), Point3f(0,0,0)	Angle= 0.493	TR2

Table C.2: Test cases for Marker

Test Case	**Test Data**	**Expected Result**	**Trace-ability**
testGetDirection()	Vector3f(10.0f, 0.56f, 0.3f)	Vector3f(10.0f, 0.56f, 0.3f)	TR3
testGetNormalVector()	Vector3f(0.0f, 0.5f, 0.9f)	Vector3f(0.0f, 0.5f, 0.9f)	TR4
testGetPatternFileName()	"C:\\BridgesResources\\marker.txt"	C:\\BridgesResources\\marker.txt"	TR5
testGetPatternSize()	Marker.Size=40	40	TR6
testGetRefPoint()	Point3f(298692.1306951222f, 73.8795095f, -5039515.30027)	Point3f(298692.1306922f, 73.87950951054f, -5039515.30027f)	TR7

Table C.3: Test cases for Marker (continued)

Test Case	Test Data	Expected Result	Trace-ability
testGetRotation()	Vector3d(0.0f, 0.5f, 0.9f)	Vector3d(0.0f, 0.5f, 0.9f)	TR8
testGetScale()	Vector3d(0.0f, 0.5f, 0.9f)	Vector3d(0.0f, 0.5f, 0.9f)	TR9
testGetTextAugmentation()	"Room 9215"	"Room 9215"	TR10
testGetTranslation()	Vector3d(200.0f, 10.5f, 0.1f)	Vector3d(200.0f, 10.5f, 0.1f)	TR11
testSetUpDirection()	Vector3d(2.0f, 1.0f, 0.0f)	Vector3d(2.0,1.0,0.0)	TR12

Table C.4: Test cases for JARToolKit3D_Main

Test Case	Test Data	Expected Result	Trace-ability
testGetInstance()	Null	Exception raised	TR13
testGetInstance_Null_ModelPane_Hanlding()	Null	Exception raised	TR14
testMarkersLoadingFromNullFile()	Null	Exception raised	TR15

Table C.5: Test cases for JARToolKit3D_Main (continued)

Test Case	**Test Data**	**Expected Result**	**Trace-ability**
testMarkersLoadingFromFile()	Path: "C:/Resources/ testmarkers.cfg"	Path: "C:/Resources/ testmarkers.cfg"	TR16
testMarkerStoredInfo()	Path: "C:/Resources/ testmarkers.cfg"	Path: "C:/Resources / testmarkers.cfg"	TR17
testMarkerOutOfRangeIndex()	Path="C:/Resources/ testmarkers.cfg"	Path="C:/Resources/ testmarkers.cfg"	TR18
testSetTrackingScale()	Path=" C:/Resources/ testmarkers.cfg"	Path=" C:/Resources/ testmarkers.cfg"	TR19

Table C.6: Test cases for JARToolKit3D

Test Case	Test Data	Expected Result	Trace-ability
testGetInstance ()	Null	Exception raised	TR20
testCreate ()	Null	Exception raised	TR21
testInitializeCamera ()	Path= "C:/Resources /JARToolkitTestFolder/data/camera_para.dat"	Path= "C:/Resources /JARToolkitTest Folder/da/camera_para.dat"	TR22
testInitializeCameraWrongFileName ()	Path="C:/Resources/JARToolkitTestFolder/data/camera_para.vat"	Path="C:/Resources/JARToolkitTestFolder/data/camera_para.vat"	TR23
testInitializeCameraWrongPath ()	Path= "C:/camera_para.dat"	Path= "C:/camera_para.dat"	TR24
testCreateBackground ()	Null	Exception raised	TR25

Traceability Messages:

TR1: Failed to create Arrow

TR2: Wrong transformation detected!

TR3: GetDirection returned wrong direction vector

TR4: GetNormalVector returned wrong Normal Vector

TR5: GetPatternFileName returned wrong Pattern file name!

TR6: GetPatternSize returned wrong Pattern size!

TR7: GetRefPoint1 returned wrong Reference Point coordinates

TR8: GetRotation returned wrong rotation angle!

TR9: GetScale returned wrong scaling!

TR10: GetTextAugmentation returned wrong Text augmentation!

TR11: getTranslation returned wrong translation adjustment!

TR12: Marker.SetUpDirection must not allow wrong direction vector.

TR13: Failed to get a singleton instanc of JARToolKit3D_Main

TR14: Failed to throw exception when an null ModelPane is passed

TR15: Failed to throw exception in LoadMarkersFromFile!

TR16: Wrong number of loaded markers from file!

TR17: Wrong loaded Marker Coordinate

TR18: Failed to throw exception in GetMarker when passed index is negative!

TR19: Wrong scaling

TR20: Failed to raise an exception

TR21: Failed to create a singleton instance of the ARToolkit3D

TR22: Failed to create an instance of JARToolKit3D

Appendix D: Requirement Testing

Table D.7: Requirement testing for Feature 1

Feature 1: Develop animated 3D arrow oriented from current user's location to the object of interest.		
Requirements	**Scenario**	**Success Criteria**
1. The orientation of the arrow should be adjusted based on the user location.	1.Launch the application 2.Click on the Settings button. 3.Select "Arrow" 4.A blue arrow oriented shows the location of the elevators of the fifth floor.	The orientation of the arrow will be adjusted correctly.
2. The arrow should be visible in AR mode.	1.Launch the application 2.Select AR mode. 3.Click on the Settings button. 4.Select "Arrow" 5.A blue arrow oriented shows the location of the elevators of the fifth floor.	The arrow should be always visible.

Table D.8: Requirement testing for Feature 2

Feature 2: Track and record the current user location in VR mode		
Requirement	**Scenario**	**Success Criteria**
1. The tracking technology should be provided in VE.	1.Launch the application. 2.Add the arrow. 3.Press *Start Record Location* button. 4.Input file name. 5.Navigate in the VE start from EV building to Hall building. 6.Press S*topRrecordLocation* button.	The location of the user are recorded in the GIS file that the name is specified by user.
2. A behavior should be developed to add the arrow to current user location repeatedly.	1.Launch the application. 2.Add the arrow. 3.Navigate in the VE. 4.An animated arrow points to the destination while navigating.	The arrow is always visible.

Table D.9: Requirement testing for Feature 3

Feature 3: Track the current user location in AR mode		
Requirement	**Scenario**	**Success Criteria**
1. Detect the user's movement with respect to the paper markers.	1.Launch the application. 2.Select AR mode. 3.Select the camera parameters. 4. Navigate in the real world.	See video background of real world in the VE.
2. Calculate the 3D coordinate representing the user's location in the real world.	1. Launch the application. 2. Select AR mode. 3. Select the camera parameters. 4. Move in the real world. 5. Go in front of the room 9.250	See the room umber overlaid on the video while moving in front of the room

Table D.10: Requirement testing for Feature 4

Feature 4: Overlay the 3D animated arrow on the video of real object displayed using the HMD.		
Requirement	**Scenario**	**Success Criteria**
Add an arrow to current location pointing to object of interest.	1.Launch the application. 2.Select AR mode. 3.Select the camera parameters. 4.Click on setting button. 5.Select Arrow. 6.Move in the real world.	See the arrow pointing to the object of interest while moving.

Table D.11: Requirement testing for Feature 5

Feature 5: Overlay a text augmentation on the video of real objects showing the recognized room number in a building.		
Requirement	**Scenario**	**Success Criteria**
Overlay a 3D text on the video showing the room number.	1. Launch the application. 2. Select AR mode. 3. Select the camera parameters. 4. Move in the real world. 5. Go in front of the room 5.125	See the room umber overlaid on the video while moving in front of the room

Table D.12: Requirement testing for Feature 6

Feature 6: Control the field of view of VE in AR Mode.		
Requirement	**Scenario**	**Success Criteria**
Adjust the field of view of virtual scene so that the user can match it with the field of view of the digital video camera.	1- Launch the application. 2 - Select AR mode. 3- Select the camera parameters. 4- Navigate in the real world. 5 – Change the field of view	See video the VE matching the real world.

Table D.13: Requirement testing for Feature 7

Feature 7: Control the transparency of the virtual objects in the virtual scene.		
Requirement	**Scenario**	**Success Criteria**
Change the transparency ratio of the virtual objects in the scene	1- Launch the application. 2 - Select AR mode. 3- Select the camera parameters. 4- Navigate in the real world. 5 – Change the transparency level using the transparency slider	See video the transparency of VE changing to different levels from opaque to a complete transparency.

Table D.14: Requirement testing for Feature 8

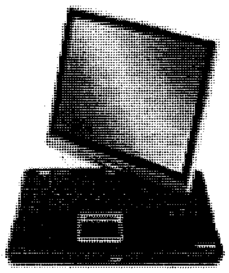
Feature 8: Control the video-tracking scale.		
Requirement	**Scenario**	**Success Criteria**
The movement of the user while the video-tracing is running should be scaled based on the selected scale.	1- Launch the application. 2 - Select AR mode. 3- Select the camera parameters. 4- Navigate in the real world. 5 – Change the tracking scale using the scale slider to 2.	See video the movement in VE matching twice the movement in the real world.

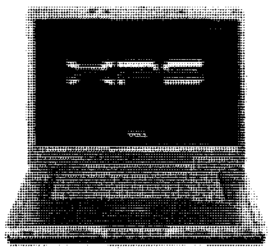
Table D. 15: Stress Testing

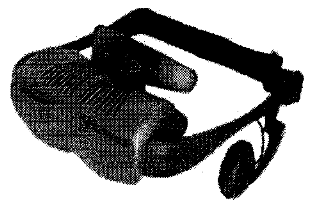
Extreme situation	**Scenario of system usages**	**System reaction**
When the marker is detected by camera but user moves the camera very frequently.	1. Launch the application. 2. Select AR mode. 3. Select the camera parameters. 4. Navigate in the real world. 5. Stand In front of marker. 6. Move the camera very frequently.	There should not too much flickering in showing virtual object on the video of real world.

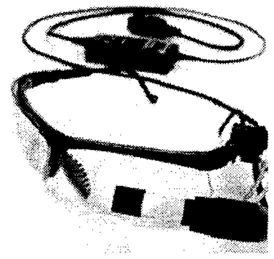
Appendix E: Equipment used in the prototype system

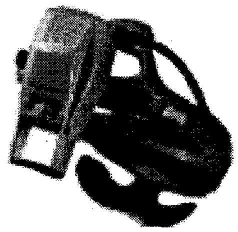
(1)


(2)


(3)


(4)


(5)


(6)

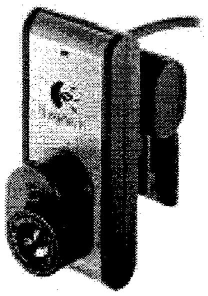

(7)

Table E.1: Equipment specifications

Device Type	Number	Brand	Specifications
Tablet PC	(1)	Panasonic ToughBook CF-18	Processor: Intel® Pentium® M Processor ULV 753, 1.20GHz Memory: 512MB+256MB SDRAM standard Hard Disk Driver: 40GB Display: (XGA) transmissive daylight readable TFT Active Matrix Color LCD, Pointing Device: Pressure sensitive touchpad with vertical scrolling support Battery Life: 7.0 hours
	(2)	Toshiba Tecra M4	Processor: Mobile Intel® Pentium® M1.86GHz, Memory: 1GB DDR2, Hard Disk Driver: 80GB, Display: TFT Active Matrix colour LCD display w digitizer, Pointing Device: Touchpad + Accupoint, Toshiba Tablet Pen, Battery Life: 3.5 hours
Laptop	(3)	Dell XPS	Processor: Intel® Core 2 Duo 2.5GHz Memory: 3GB Hard Disk Driver: 200GB Display: Standard LCD Display with 2MP Camera Pointing Device: Pressure sensitive touchpad Battery Life: 2.5 hours
HMD	(4)	I-glasses 3D Pro	Resolution: 800 x 600 1.44 Million Pixels per Display True Black Background Field of View: 26 Degrees Diagonal Virtual Image Size: 70" at 13' Color Depth: 256 Levels per Color (True 24 Bit) Weight: < 226 gram

Table E.1: Equipment specifications (continued)

Device Type	Number	Brand	Specifications
HMD	(5)	MicroOptical SV-6	Display Format: 1280 x 1024 pixels, 60 Hz refresh rate Display Color: 24-Bit. Field of View: Approximately 48° horizontal, 60° diagonal, Input Signal: SXGA format Brightness (nits): 30 fL max
	(6)	Microvision Nomad ND2000	Display Format: SVGA 800x600 pixels, 60 Hz refresh rate Display Color: Monochrome Red Field of View: 17.25° horizontal, 23° diagonal Input Signal: SVGA format Focus Range: Adjustable from 1 feet to infinity. Continuous Operation: 8 hours
Digital Video Camera	(7)	Logitech QuickCam for Notebooks Pro	Resolutions: 640x480 @ 30fps Features: Built-in microphone Connectivity: USB

Appendix F: Usability Testing

1. Usability and usefulness

Usability and usefulness are related but different. Nielsen (1994) differentiates between usefulness and usability. "Usefulness is the issue of whether the system can be used to achieve some desired goal. Usability applies to all aspects of a system with which a human might interact..."

The usability testing is about usability not usefulness. For example, a system can be amazingly easy to use but of no use. Finding out if a feature is useful can be during user-needs assessment stage of the development process. "Usability testing is a poor way to assess utility" (Dillon, 2003).

2. Usability Definition

"Usability is the broad discipline of applying scientific principles to ensure that the system/site designed is easy to learn, easy to use, easy to remember, error tolerant, and subjectively pleasing" (The Usability Company, 2004).

Usability evaluation is a process that aims to identify usability problems in user interface design (Mack and Nielsen, 1994).

The usability is also defined as "the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use" (ISO 9241-11, 1998)

3. Usability Guidelines for AR Systems

Joseph L. Gabbard (Gabbard, 2001) from Virginia Tech's Systems Research Center presented a list of usability guidelines as preliminary results of a research on VE and AR usability engineering. The main guidelines relevant to AR systems are summarized as below:

(1) Multi-user mode

- Information about other collaborative users should be always available even if they are physically occluded or located in a remote place.
- In case of multi-user collaborative environment, users should be allowed to share tracking information.
- In collaborative environment, users should be able to control the type and the extent of information to share or to keep private.

(2) Navigation

- Interaction techniques (e.g. navigation) should not require noticeable portion of the user's attention.
- Spatial labels, signs, landmarks, and a compass that shows the North direction should be included when appropriate.
- The user should be always able to find his/her location, altitude and orientation. Also, he/she should be able to know how to get to a certain location.
- Non-direct manipulation should be possible (such as query-based selection) in case of temporal, descriptive, or relational selection criteria.
- High frame rates and low latency are necessary for 3D target acquisition.
- Accurate representation of location and orientation of graphics and text is recommended.

- For large environments, navigational grid and/or navigational map should be included.
- Multiple degrees of freedom input is well suited for coarse positioning tasks, but not for tasks which require precision.
- The number of degrees of freedom of the navigation technique and device should match the nature of the task. For example, menu selection should not require an interaction technique or device with more than two degrees of freedom.
- The trackers should provide enough accuracy for small fractions of degrees in orientation and for few millimeters in position

(3) Speech Recognition and Natural Language

- The user should be able to record and playback annotations in a quick, efficient and unobtrusive way. Annotations should be seamlessly integrated.
- The user should be able to edit, remove or save annotations.

(4) Visual Feedback

- The AR system should be responsive and prompt. Timing and response delay affects user performance.
- The user should be allowed to adjust the visual display (e.g. illumination and contrast levels).
- The wearable display should be sufficiently comfortable and optically transparent for the user.

(5) Visual clarity

Information displayed on the screen should be clear, well organized, unambiguous and easy to read.

4. The Evaluation Checklist

The evaluation checklist consists of a set of specific questions intended for usability assessment. The checklist provides standardized and systematic way of identifying problem as well as improvement areas and good aspects of the system (Ravden et al., 1989).

Using this method, the evaluator will be asked to carry out tasks that have been designed to be performed as part of the evaluation. The tasks should represent the work the system is designed to perform, thus should test as many of the system functions as possible.

The evaluation checklist is based on nine criteria that a well-designed user interface should meet (Ravden et al., 1989). These criteria are Visual clarity, Consistency, Compatibility, Informative feedback, Explicitness, Appropriate functionality, Flexibility and control, Error prevention and correction and User guidance and support.

5. IsoMetrics

IsoMetrics is a usability inventory developed at the University of Osnabruck (Gediga et al., 1999), based on seven usability principles from norm ISO-9241 Part 10, being: suitability for the task, self-descriptiveness, controllability, conformity with user expectations, error tolerance, suitability for individualization and suitability for learning. IsoMetrics is available in two different versions; the first is IsoMetrics short, designed for the evaluation of an existing product, known as summative evaluation, and the second is

IsoMetrics long, designed for the evaluation during the development process and known as formative evaluation.

6. Evaluation heuristics for VE Displays

Recently, a research team from Virginia Polytechnic Institute and State University (Shiratuddin et al., 2004) has carried out an experiment to compare and evaluate the effectiveness of five VE displays. The comparison has been based on six VE features. The effectiveness of these features can be considered as evaluation heuristics.

Quality of visual presentation of the model

Some VE displays provide better quality of visual presentation of the model by allowing the user to correctly and easily identify different elements and objects of the model and their sizes. The colors used in the 3D model should be distinctive and the textures should look as good as expected. Also, the colors should not be too shady or too dark. In addition, the images resolution should be high enough and evenly distributed.

Physical comfort

Fatigue is one of the main causes of discomfort. The weight of the HMD can be one of the reasons of fatigue. Also, using glasses can be bothering for the users and hence causes discomfort. In addition, losing focus on the 3D model can cause frustration and discomfort.

Level of realism

The VE display should provide realistic views and images of the environment. 3D models should be close to the real world (e.g. life-size). The system should provide wide and surrounding field of view and ease of seeing details and of keeping from walking through walls. Moreover, walkthrough mode can considerably add to the realism of the system.

Ease of navigation

The simplicity of moving around in the VE is one of the factors that affect the ease of navigation. Similarity with videogames and existing widely used systems can substantially improve the easiness of navigation. Walkthrough mode makes the navigation easier as well as realistic. Navigation techniques should be easy to learn.

Ability to keep one from getting lost

The user should not feel disoriented or lost. Making the user think for a while on his/her movement and on how to navigate in the environment is one of the factors of disorientation and lost.

Suitability for making decisions and performing tasks

The ability to easily identify elements of the model, the large field of view and the walkthrough mode can significantly help in the group decision-making. Also, accommodating for more than one user at a time added a high sense of involvement and presence in the VE are considered very suitable for group decision.

Appendix G: Ladybug Spherical Camera Interface

Since the video tracking relies on the field of view of the digital video cameras, a new type of digital camera, called Ladybug spherical digital camera, can be used to get a higher coverage of the real world environment. A configuration and interfacing method has been proposed, as shown in Figure G.1, to allow the system to use this type of camera and keep the same tracking toolkit (ARToolkit and its Java Binding JARToolkit). The spherical digital camera feeds the application with video frames from five cameras from the left, right, front, back and top sides. The proposed interface retrieves video frames from the five cameras, one video frame from each camera at a time through a Java Native Interface (JNI) that communicates with the cameras through the Ladybug Application Programming Interface (API). The video frame is then sent to the tracking toolkit (ARToolkit) through the JNI using JARToolkit to be checked for marker detection. When a marker is detected, ARToolkit returns a transformation matrix for the detected markers that defines its position and orientation with respect to the marker detected in the video frame and returns the identification number of the marker. Knowing the camera which detected the video frame containing the marker, the position and orientation of the marker and its identification are then sent to the main application to compute the user location and orientation. Based on the camera number, the marker identification and the predefined angles physically separating the cameras, the final user location and orientation is then computed and the virtual augmentation is displayed.

The video frames from the front camera are rendered as the video of the real world on which the augmentation is added.

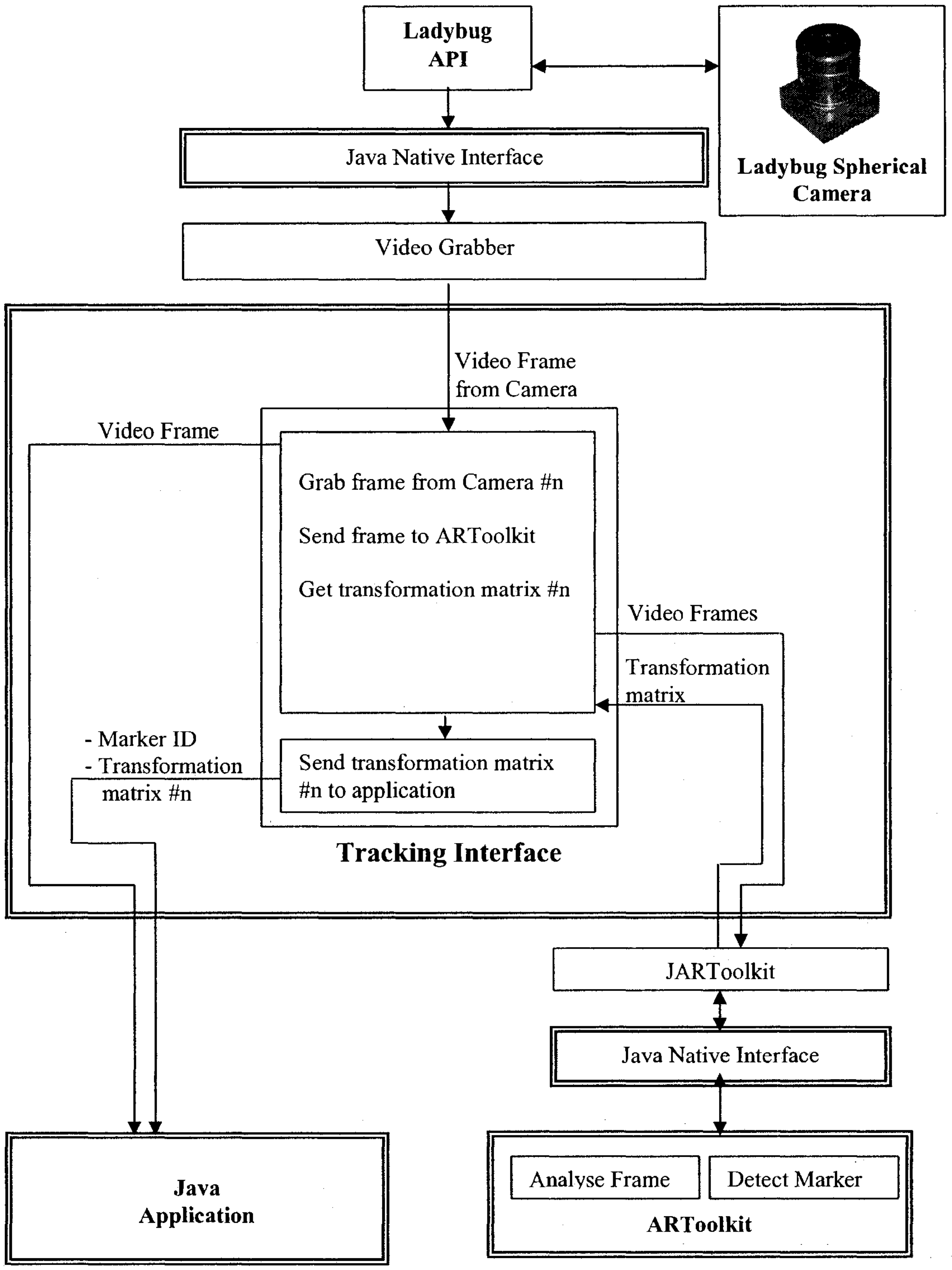


Figure G.1: Structure of a Ladybug API Interface Extension

Note:
- #n: Camera number from 1 to 5
- Transformation Matrix #n: location and orientation of Camera #n with respect to a marker.

Appendix H: List of publications

Conference papers:

(1) Mozaffari, E., **Khabir, B.**, Zhang, C., Devarakonda, P., Bauchkar, P., and Hammad, A. (2005). Interaction Models for Infrastructure Management Systems Using Virtual and Augmented Realities, International Congress of Urbistics, Montreal, Canada.

(2) Hammad, A., Mozaffari, E., **Khabeer. B.**, and EL-Ammari, K. (2006). Framework for Virtual and Mixed Reality Applications in Civil Engineering, Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada.

(3) Hammad, A., **Khabeer, B.**, Mozaffari, E., Devarakonda, P., and Bauchkar, P. (2005). Augmented Reality Interaction Model for Mobile Infrastructure Management Systems, 1st CSCE Specialty Conference on Infrastructure Technologies, Management and Policy Toronto, Ontario, Canada.

Journal papers:

(1) Hammad, A., Mozaffari, E. and **Khabeer, B. (**Submitted in 2006). Framework for mixed reality applications in civil engineering, Journal of Computer Animation and Virtual World.