

AUGMENTED REALITY BASED INDOOR POSITIONING NAVIGATION TOOL

by

LOW CHEE HUEY

DISSERTATION

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

© Copyright 2010

by

Low Chee Huey, 2010

CERTIFICATION OF APPROVAL

**AUGMENTED REALITY BASED INDOOR POSITIONING
NAVIGATION TOOL**

by

LOW CHEE HUEY

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONIC ENGINEERING)

Approved by,

(Mr. Patrick Sebastian)

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

DECEMBER 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(LOW CHEE HUEY)

ABSTRACT

Indoor positioning gained popularity recently due to its potential to be used in the increasing complexity of indoor environment. Unfortunately GPS signals are restricted from indoor purposes. The main objective of this project is to design a new method to develop indoor positioning navigation system without using wireless technology through image processing. Apart from that, the project aims to develop an interactive indoor navigation system. Augmented reality is being used to superimpose the directional signage on the real view of the indoor environment in 3D form. Along with the 3D guides, voice guidance will be output from the system to assist users in identifying their locations easily. Overall scope of study mainly revolves the research on augmented reality, audio API, and other additional techniques that could improve the program in computing the route. As development phase, laptop has been used as computing device. In the future, this idea could be broadly applied to mobile devices such as mobile hand phones and PDA as added indoor navigation functionality without using GPS and wireless communication. The system has been tested at ground floor in Information Resource Centre (IRC) and the results show the flexibility of the system in navigating 12 locations and handling up to 30 possible routes.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to persons who helped me a lot in completing the project. First and foremost I want to show my appreciation to my project supervisor, Mr. Patrick Sebastian for his guidance and advice on the project. Mr. Patrick has contributed a lot of constructive ideas for this project.

Secondly, I would like to thank my colleagues including a post graduate student from IT department for supporting and guiding me on the knowledge of using ARtoolkit and 3D modeling software.

Last but not least, I would like to show my gratitude to my family for supporting me in my academic so that I could do the best for my final year project.

TABLE OF CONTENT

ABSTRACT	iv
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Scope of Study	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Augmented Reality	5
2.2 ARtoolkit.....	8
2.2.1 <i>Introduction of ARtoolkit</i>	8
2.2.2 <i>Image Processing of ARtoolkit</i>	10

2.2.3 <i>Analysis of ARtoolkit's Accuracy</i>	13
2.3 OpenAL.....	15
CHAPTER 3 METHODOLOGY	16
3.1 Procedure of Project Identification	16
3.2 Project Methodology.....	17
3.2.1 <i>Research</i>	17
3.2.2 <i>Basic Software Development</i>	17
3.2.3 <i>Demo Development</i>	17
3.3 Tools and Required Equipment	19
3.4 Project Operation	20
CHAPTER 4 RESULT & DISCUSSION	22
4.1 Program Initialization	22
4.2 ARtoolkit.....	23
4.2.1 <i>ARtoolkit: Initialization</i>	24
4.2.2 <i>ARtoolkit: Frame Acquisition</i>	25
4.2.3 <i>ARtoolkit: Marker Detection</i>	26
4.3 Route Planner Algorithm	26
4.4 ARtoolkit: VRML Model Rendering	28
4.5 Audio Module	29

4.6 Overall Summary of the Result.....	31
CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS.....	33
5.1 Conclusions.....	33
5.2 Recommendations.....	34
REFERENCES	35
APPENDICES.....	37
APPENDIX A :Floor Plance of IRC Ground Floor	38
APPENDIX B :Source Code of Simple VRML Program	39
APPENDIX C :Patterns Used to Evaluate Maker Matching Algorithm	56
APPENDIX D :Results of Maker Matching Algorithm Evaluation Under Fluorescent (White) Lighting	59
APPENDIX E :Results of Maker Matching Algorithm Evaluation Under Tungsten-Halogen (Yellow) Lighting	62

LIST OF FIGURES

Figure 1 Real desk with virtual lamp and two virtual chairs [4]	5
Figure 2: Optical see-through HMD by Hughes Electronics [4].....	6
Figure 3 Video see-through HMD by Hughes Electronics [4]	6
Figure 4 A Mini augmented reality ads hit newsstands [5].....	7
Figure 5 Main ARtoolkit pipeline [6].....	9
Figure 6 Computer vision and image processing in ARtoolkit [6]	11
Figure 7 Working flow in ARtoolkit [7]	11
Figure 8 Pattern normalization and template matching in ARtoolkit [12].....	12
Figure 9 VRML object being overlay on the marker	12
Figure 10 Plot of systematic error of ARtoolkit at different camera distance [14]	14
Figure 11 OpenAL logo [8].....	15
Figure 12 Project activities.....	16
Figure 13 Project Overview.....	21
Figure 14 Console output window	22
Figure 15 Flowchart of ARtoolkit algorithm.....	23
Figure 16 Camera parameter setting window.....	24
Figure 17 Marker dimensions [9]	25

Figure 18 Output display	28
Figure 19 Flowchart of OpenAL algorithm.....	29
Figure 20 Marker is placed in IRC	31
Figure 21 Updated map with current route.....	32

LIST OF TABLES

Table 1 Software and hardware used in the project	19
Table 2 Lookup table that defines the route between the check points in IRC ground floor.....	28
Table 3 Indication of the code	28

LIST OF ABBREVIATIONS

API:	application programming interface
AR:	augmented reality
GSM:	Global System for Mobil Communication
GPS:	global positioning signal
HMD:	head mounted device
OpenCV:	Intel open source computer vision
RFID:	radio frequency identification
USB:	universal serial bus
VE:	virtual environment
WLAN:	wireless local area network
WPF:	Window Presentation Foundation
3D:	three-dimensional

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Indoor Positioning technology is to locate and track objects in an indoor area. Currently GPS technology fail to perform positioning in indoor area as the GPS signal get attenuated when it pass through the obstacles and walls of the building. Experiments[1] reveal that the attenuation of Global positioning system (GPS) that is higher than 1dB per meter of structure is estimated to drop down to roughly -160dBW to 200dBW. However the lowest accepted working range based on GPS coarse acquisition code it is designed to reach on the ground not less than -150dBW power level. Hence GPS is not suitable for enclosed areas.

Nevertheless there are organizations that develop indoor positioning based on wireless technology such as Bluetooth, GSM (Global System for Mobile Communication), RFID, and WLAN. Nokia Research Centre has explored the research field of indoor positioning technology that using wireless technology. Their mobile phone S60 named as Kamppi Trial [2] is equipped with the indoor positioning functionality. Other than that there are researchers are having great passion to further explore the indoor positioning field by taking cooperative efforts such as organizing conference and publishing papers. The 2010 International Conference on Indoor Positioning and Indoor Navigation [3] organized by Indoor Positioning and Indoor Navigation organization, IEEE and other related bodies serves the mentioned purpose.

Augmented Reality (AR) [4] is a visual enhanceive technology whereby the computer-generated graphics being superimposed on the user's real view to create the mixture of virtual and reality effects. AR technology has been invented since 1950s' and for current few years it has been explored rapidly in more domestic fields such as gaming fields and personal research projects from previous industrial application e.g. medical visualization, industrial manufacturing, military aircraft navigation and etc. The motivation force that driven behind the technology is the enhancement of the visual effect that could assists users in presuming certain object more easily. Therefore the interactive 3D image being superimposed on the real scene may improve the efficiency in training based application such as assisting doctors in surgery demonstration, robotic design and etc.

1.2 Problem Statements

Indoor positioning application appears increasingly important as the result of the development of constructions around the world. However GPS signals will be attenuated down to undetectable level in enclosed areas, it cannot be used in navigating in the indoor environment.

Although there are few newly invented indoor positioning applications that are based on wireless technology such as WLAN and Bluetooth, they are very dependent on the signal coverage provided to that particular area. Hence the performance of the indoor positioning system will be very fragile to the coverage of the wireless signal in the indoor environment.

The conventional navigation guidance provided for indoor purpose such as map and signage around the building are not directing in straight forward manner since they take some time for the users to figure out their exact location and find the route to their wanted locations. Such condition gets worse in the buildings with very complicated internal layout design.

1.3 Objectives

- i. To develop an interactive indoor positioning system with
 - ✓ 3D view (AR) of navigation information display
 - ✓ Navigation information with voice guidance
- ii. To design new method for navigating and locating without using wireless technology through
 - ✓ Concept of matrix (Lookup table) to relate locations

1.4 Scope of Study

The scopes of study include the field of computer vision and image processing; field of audio programming and field of 3D modeling. The main scope of study is revolving computer vision and image processing which is mainly applied in ARtoolkit. Several image processing techniques have been applied to process the incoming frame to detect markers. Other than that ARtoolkit has applied several computer graphics API includes OpenGL and GLUT. They are the cross-platform for writing application to produce computer graphics.

Audio programming is another scope of study in this project which is using the OpenAL library to generate 3D audio output. This involves the programming techniques to manipulate the effect or sounds and functionality to load the sound tracks.

The project involves 3D modeling to create the VRML model for augmented reality rendering.

CHAPTER 2

LITERATURE REVIEW

2.1 Augmented Reality

Augmented Reality (AR) [4] is a term of a live direct or indirect view of physical real-world environment whose element are merged with (or augmented by) virtual computer-generated imagery creating a mixed reality. For example, Figure 1 is showing the result of AR. The table lamp and chairs are computer generated image which have been superimposed on the real scene (the table with the telephone on it). It creates the experience of mixed reality to the users. However it is not only limited to sight but also involve other sense such as sound. But current most of the research works are focus on sight. The general concepts of the AR require it to be mixture of reality and virtual objects; real-time running; interactive and immerse users in 3Dscene.



Figure 1 Real desk with virtual lamp and two virtual chairs [4]

Usually the AR requires the use of Head-Mounted Display (HMDs) in order to immerse the user in a mixed reality world. Both optical and video technologies (Figure 2 and Figure 3) are being applied to accomplish the augmented reality effects. Video see-through augmented reality enables user to view virtual images overlay on the live video of the real world. While optical see-through augmented reality superimposes virtual images directly on the view of real world.



Figure 2 Optical see-through HMD by Hughes Electronics [4]



Figure 3 Video see-through HMD [4]

However recently the AR technology are being explored become being able to be accomplished in much convenient way, for example [5] the augmented reality hit newsstands which create a truly interactive media piece out of a 2-dimensional pre-recognized image symbol which is illustrated in Figure 4 below.



Figure 4: Mini augmented reality ads hit newsstands [5]

2.2 ARtoolkit

2.2.1 Introduction of ARtoolkit

ARtoolkit 2.72.1 is a C and C++ language open source software library which is released by HIT Lab web site [6] to provide programmers the foundation to build project with AR applications. In general ARtoolkit is mainly to overlay a predesigned 3D object on the detected marker. The great part of ARtoolkit is that it is able to real-time precisely track the view point of the user by using computer vision techniques to calculate the camera position and orientation relative to the marker orientation. ARtoolkit has to be supported by the other computer vision libraries and drivers. The prerequisites that need to be installed to run ARtoolkit include:

- Microsoft Visual Studio .NET 2003

The development environment required to build and debug the project.

- DSVideoLib-0.0.8b-Win32

The library used to handle the interface of the Web Cam.

- OpenGL

It is a cross-platform APIs that is mainly for writing application to produce 2D and 3D computer graphics. It is used to render the virtual object.

- GLUT (The OpenGL Utility Toolkit)

It implements a simple windowing application programming interface (API) for OpenGL. It is used by ARtoolkit for windows and event handler.

- Video input device (USB Web Cam)

Web Cam is needed to capture live view and send the frame to be processed.

The working flow of ARtoolkit is based on a global pipeline metaphor which is shown in Figure 5.

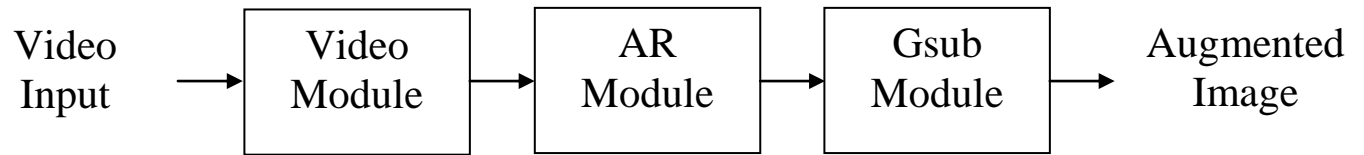


Figure 5 Main ARtoolkit pipeline [6]

The video input is grabbed from camera by the video module which acts as the interface between the driver of the camera and ARtoolkit. AR module includes the processing over the video frames such i.e. viewpoint tracking. Finally the virtual image is drawn by the Gsub module and output as augmented image.

2.2.2 *Image Processing of ARtoolkit*

Figure 6 and 7 shows the image processing process of ARtoolkit. First a frame from video stream is grabbed from the web cam. The image will be converted into binary image (black and white) based on the threshold value. It is represented by the binarization and thresholding [11] [12] block. Next the program will look for square regions by using image labeling technique where those connected components and the size which is almost to accommodate a fiduciary marker will be extracted. After that, the outlines of the contours that can fit four line segments will be recognized as square regions. The corner will then be detected from the mentioned contour. The square region can be in any orientations therefore it must be normalized to the initial orientation as shown in Figure 9. The pattern inside the normalized square region will then be compared with the pre-trained markers which have been stored as binary data. If there is a match, confidence value which is the percentage of matching will be computed.

When there is a marker being detected, the position and orientation of the marker relative to the camera will be computed. After the view point being computed OpenGL API will be used to overlay the VRML model on the marker based on the computed view point (camera's coordinates). As the result the virtual object will appeared exactly aligned with the marker (Figure 7). Same process is repeated for real time processing.

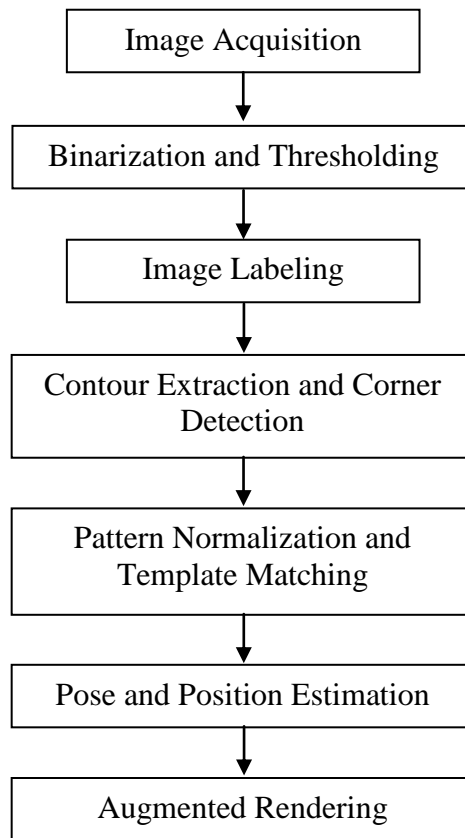


Figure 6 Computer vision and image processing in ARtoolkit [6]

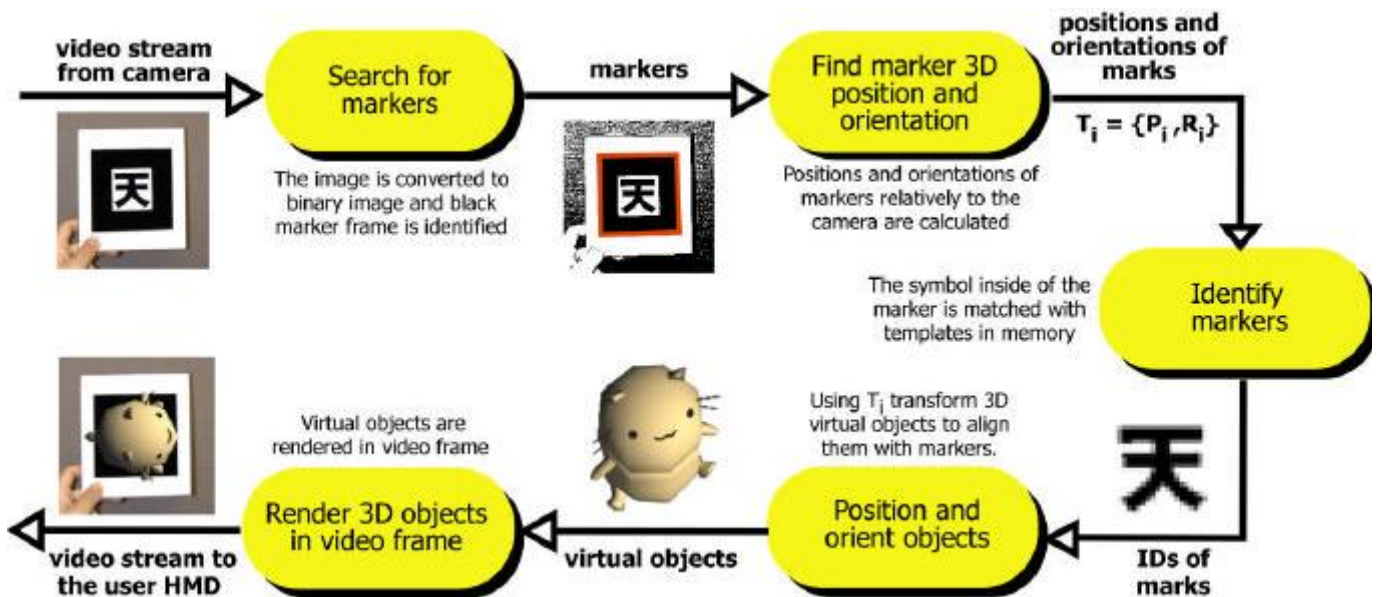


Figure 7 Working flow in ARtoolkit [7]

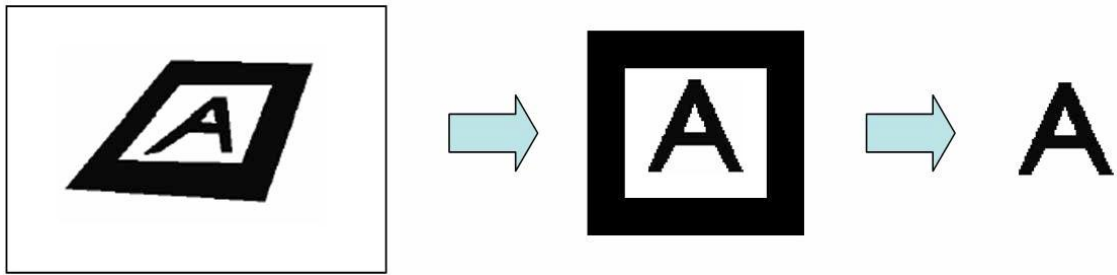


Figure 8 Pattern normalization and template matching in ARtoolkit [11]

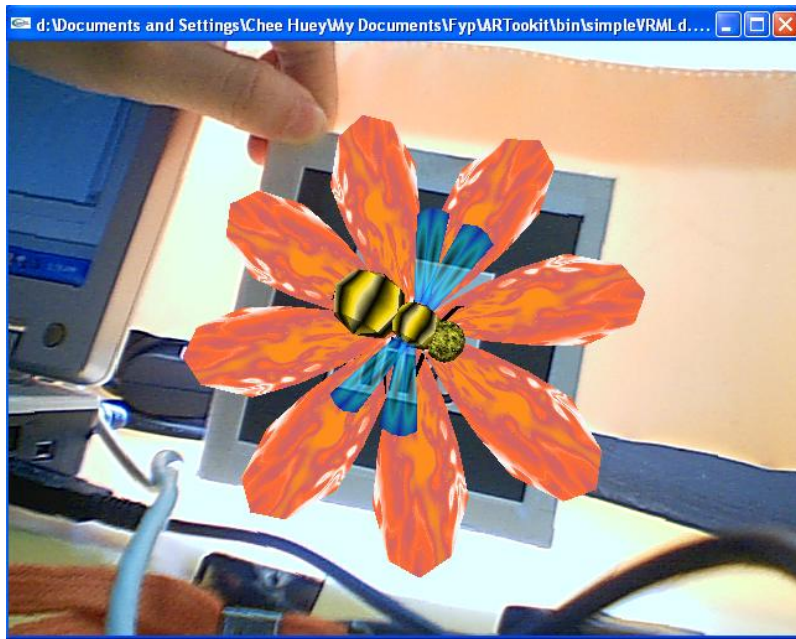


Figure 9 VRML object being overlay on the marker

2.2.3 Analysis of ARtoolkit's Accuracy

The performance of ARtoolkit plays an important role in determining the performance of the entire indoor positioning system as the 3D arrows and all navigation guides are depends on the detection input from ARtoolkit. The subsequence modules in the program are more stable compare to the detection module of ARtoolkit.

The performance of ARtoolkit can be evaluated from its ability of detecting a marker and the tracking accuracy upon the detection of marker. Confidence value that represents the matching percentage of the detected marker with the pre-trained marker will be calculated by ARtoolkit once it detects a marker.

The accuracy of ARtoolkit to detect a marker is very dependent on the lighting condition from where it detects the marker. If the lighting condition is similar to the lighting condition where the marker being trained then the threshold value is appropriate to easily detect the marker else it cannot. However the threshold value is adjustable to suit different lighting condition in the indoor environment.

On the other hand, the tracking accuracy of the ARtoolkit upon the detection is shown in Figure 10. The experiment [13] is carried out with the conditions that a 55mmx55mm marker is being fixed at x- and z-directions while changing y-direction (20cm to 100cm) from the camera. The camera used is a web cam with resolution 640 x 490 pixels and frame rate of 15 just as the configuration of this project.

The result shows that the systematic error of ARtoolkit to continuously track a marker is low for the estimated distance from 20cm up to 70cm from the camera to the marker. This is considered acceptable as the expected working range from the marker to the user will be fixed within the mentioned range.

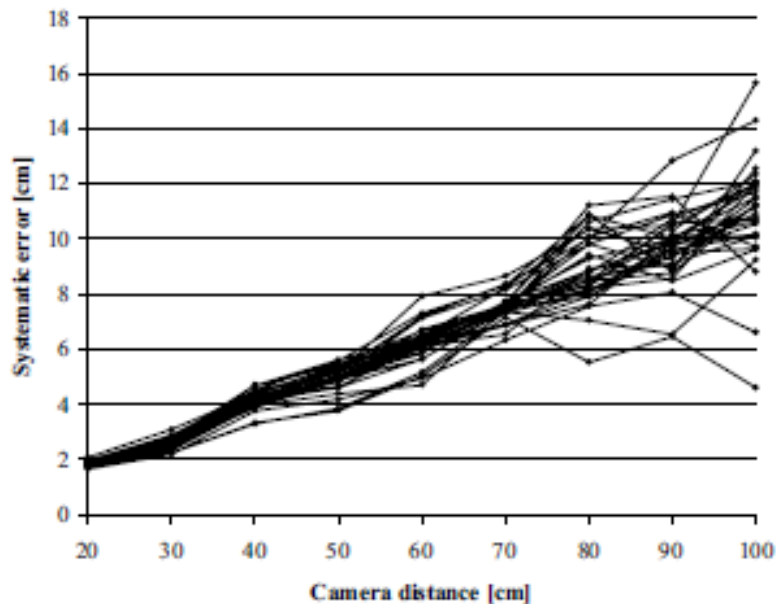


Figure 10 Plot of systematic error of ARtoolkit at different camera distance. [13]

2.3 OpenAL

OpenAL [8] is a 3D audio API that is used to handle audio in cross-platform application. It enables the developer to set the position and velocity of the source and listener to achieve the 3D audio effect. The OpenAL algorithm will be discussed further in the result and discussion. OpenAL SDK for windows has to be installed before it is applied in programming.



Figure 11 OpenAL logo [8]

CHAPTER 3

METHODOLOGY

3.1 Procedure of Project Identification

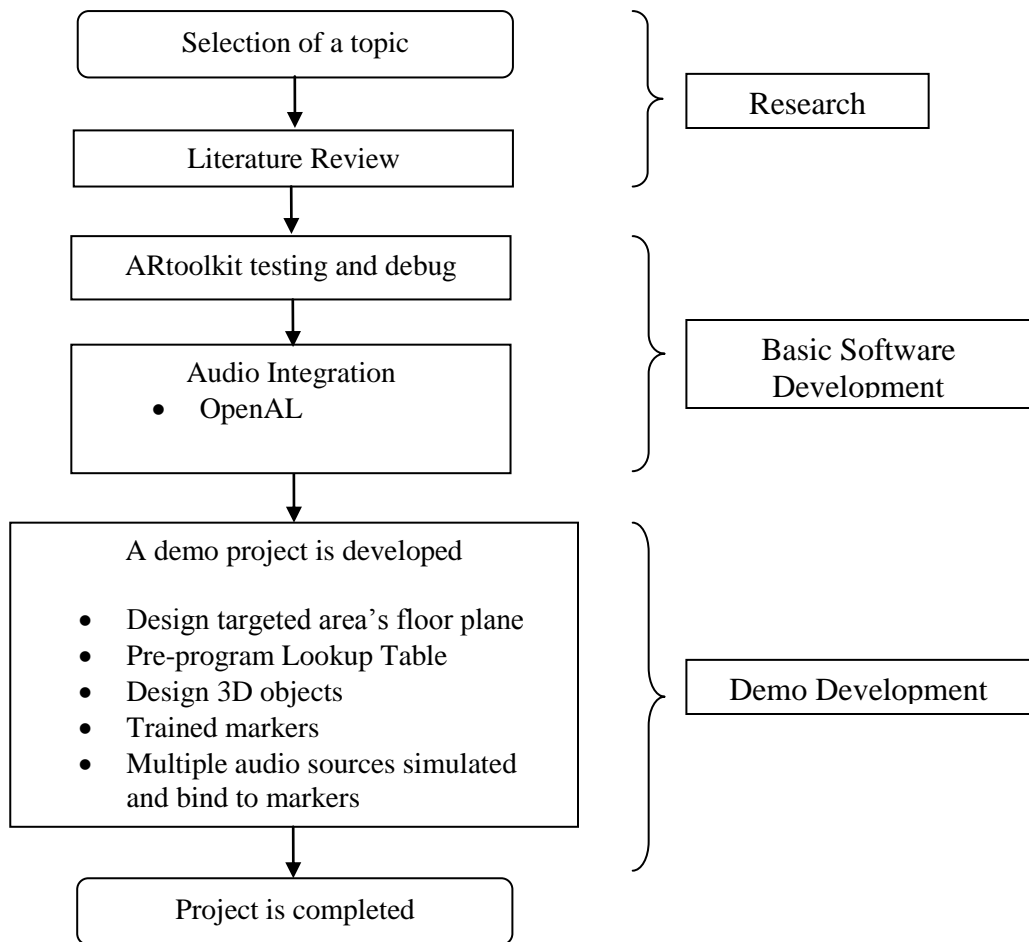


Figure 12 Flow chart for project activities

3.2 Project Methodology

The project can be divided into mainly three phases i.e. research phase; basic software development; and demo development.

3.2.1 Research

At the research phase, research methodology is started from the selection of a topic followed by a literature review. Augmented reality field is the main focus of the research at the beginning of the project. Many applications of augmented reality have been explored include initial study on AR applications journals, AR working concept journals and review of AR application projects from internet. Several version of open source augmented reality projects are available from internet so it is necessary to screen through the software and projects.

3.2.2 Basic Software Development

Several AR software being downloaded and tested then come to the final selection-ARtoolkit due to its suitability for further development. To debug ARtoolkit some prerequisites such as Windows APIs, drivers and integrated development environment (IDE) needed to be installed. Similar to OpenAL, the SDK is needed to be installed before it is applied in the programming. The algorithm of the ARtoolkit and OpenAL is discussed further in result and discussion.

3.2.3 Demo Development

In order to develop a complete demo navigation project, the directional information needed to be input to the database therefore a collection of background information of the targeted indoor area is necessary. The targeted indoor area for the demonstration of this project is the ground floor of UTP's Information Resource Center (IRC). The required background information includes floor plan and introductory information on certain checkpoints. The navigation information then will be converted

into VRML models (such as 3D arrows) 3D modeling software while the corresponding sound tracks will be created by text-to-speech software.

For pre-processing stage, the floor plane of IRC (Appendix) is designed using Edraw Max 5. The check points have to be pre-defined. They are chosen around the junction and corner areas. After the checkpoints being indentified, markers are needed to be trained and each of them will be placed at their respective checkpoint. Each checkpoint shall have only a unique marker. Then certain locations will be selected to be included into the list of destination for user to select from at the beginning of the program. Then a matrix (lookup table) that contains the routes between the locations and the check points will be programmed into the indoor positioning program together with the floor plane. At the same time, the sound tracks that corresponding to different navigation guidance will be created using the AT&T Labs Natural Voices® Text-to-Speech Demo program and being stored into the program.

The project is completed when the demo navigation project is tested and proved to be working in handling up to 12 locations and more than 30 routes. IRC ground floor as the demonstration area has been divided into 12 important locations.

3.3 Tools and Required Equipment

The project mainly revolves programming while the hardware involved are only computing device (e.g. laptop) and USB Web Cam. The table below shows the list of software and hardware that required implementing the project.

Table 1 Software and hardware used in the project

Software	Hardware
Microsoft Visual Studio 2003	USB Web Cam
ARtoolkit	
GLUT	
OpenAL SDK	Laptop (acts as processing and display unit)
AT&T Labs Natural Voices® Text-to-Speech Demo	
Autodesk 3D Studio Max 9	
Edraw Max 5.1	

3.4 Project Operation

A complete set of hardware and software needed for this project is a computing device (for example laptop which also acts as a displaying device) and a USB web cam.

The web cam will keep capturing the live image frames and send to the computing device. ARtoolkit will process the frame and recognize if there is at least one trained marker (pattern) being detected. If no marker is detected then the displaying device (laptop) will only shows the usual view of the live image captured from web cam. However if marker is detected, then the marker detection module in ARtoolkit will pass the detected marker ID to the route planner module.

The route planner module will compute the route based on the detected marker (which represents user's current checkpoint) and the desired location selected by the user. The route computation is based on the lookup table that pre-programmed to the system. The output from the route planner is the navigation information which will be sent to the VRML object rendering unit and audio module.

The rendering unit of ARtoolkit will overlay the corresponding 3D virtual image on the live view of the web cam based on the navigation information received from route planner. Other than textual information, sound track that corresponds to the navigation information will be played. Next, the image of the user's location on the map will be updated with the route being specified.

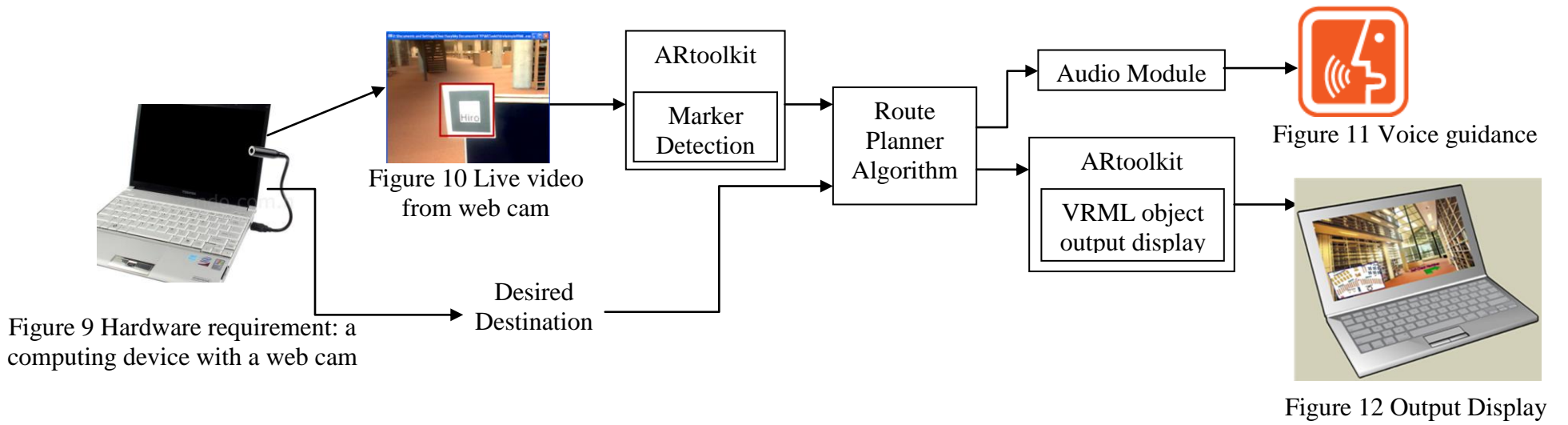


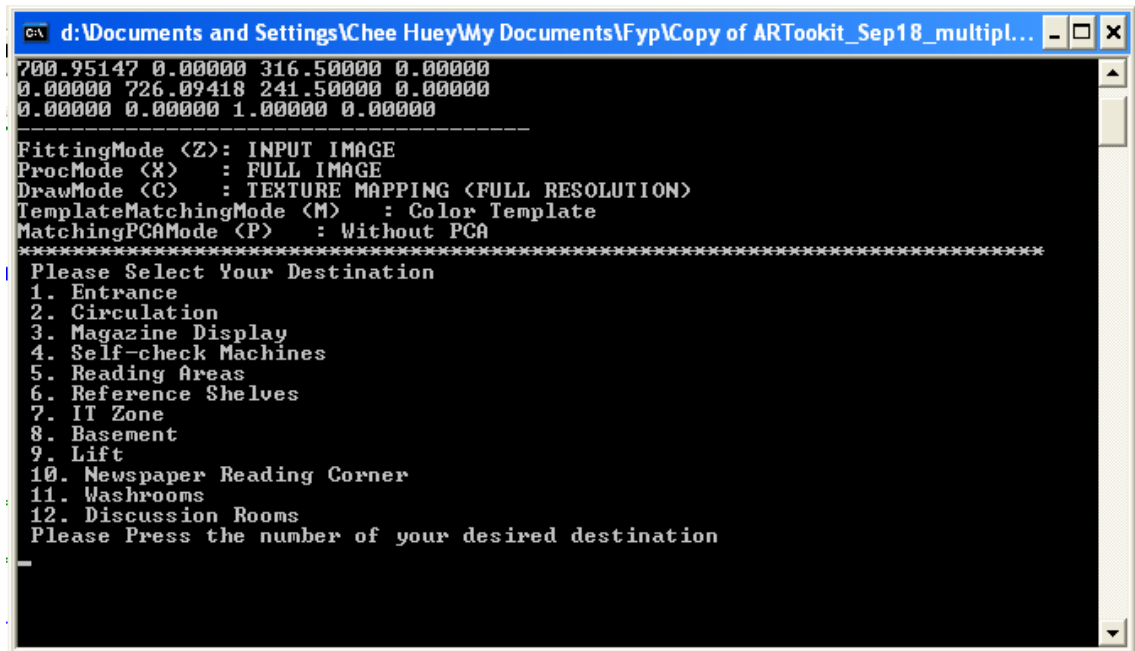
Figure 13 Project overview

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Program Initialization

User is required to select a targeted location at the beginning of the program. A console window prompted to show the list of destinations. Then the program will store the location that input by the user.



```
C:\ d:\Documents and Settings\Chee Huey\My Documents\Fyp\Copy of ART toolkit_Sep18_multipl... - □ ×
700.95147 0.00000 316.50000 0.00000
0.00000 726.09418 241.50000 0.00000
0.00000 0.00000 1.00000 0.00000
-----
FittingMode <Z>: INPUT IMAGE
ProcMode <X> : FULL IMAGE
DrawMode <C> : TEXTURE MAPPING <FULL RESOLUTION>
TemplateMatchingMode <M> : Color Template
MatchingPCAMode <P> : Without PCA
*****
Please Select Your Destination
1. Entrance
2. Circulation
3. Magazine Display
4. Self-check Machines
5. Reading Areas
6. Reference Shelves
7. IT Zone
8. Basement
9. Lift
10. Newspaper Reading Corner
11. Washrooms
12. Discussion Rooms
Please Press the number of your desired destination
_
```

Figure 14 Console output window

4.2 ARtoolkit

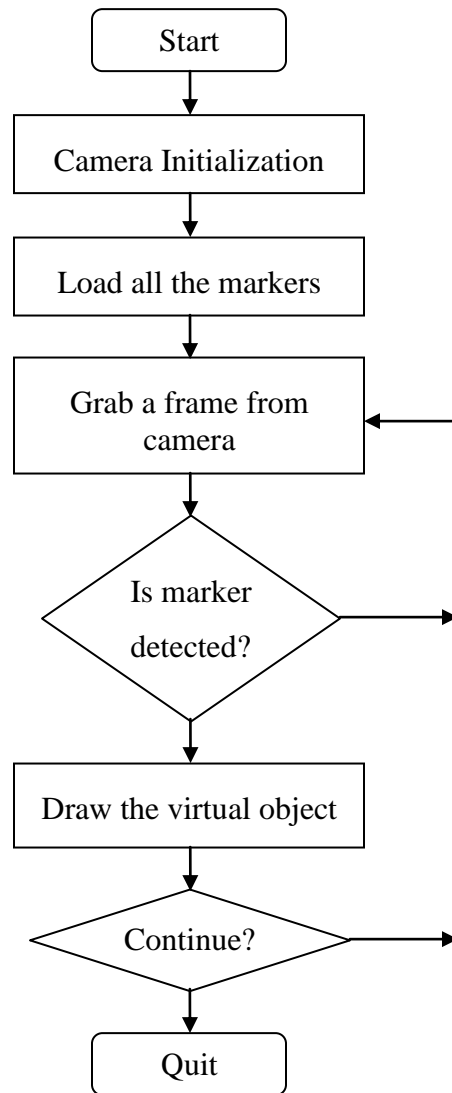


Figure 15 Flowchart of ARtoolkit algorithm

Figure 16 shows the processing flow of ARtoolkit. Some other modules such as route planner algorithm and audio module has been integrated into ARtoolkit to run the indoor positioning navigation.

4.2.1 ARtoolkit: Initialization

Camera Initialization

The program is started by detecting USB connected camera. After the camera calibration, the parameter of the camera will be saved as a file then it will be loaded during the camera initialization. Figure 17 shows the window prompt to ask user to select several setting of the camera. Output size is referring to the size of output console window.

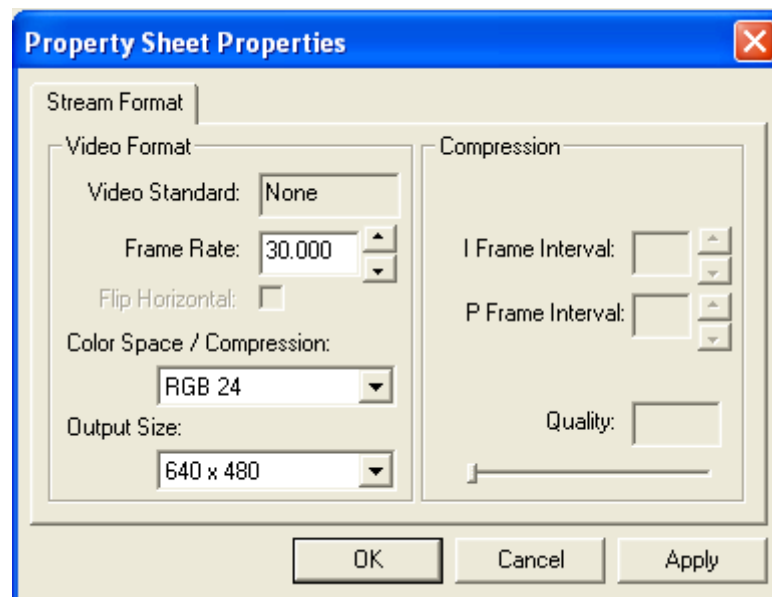


Figure 16 Camera parameter setting window

Load all the markers

The markers have to follow certain criteria such as black and white, rectangular and the ratio must follow as shown in Figure 17 to be recognized. Trained markers are kept in the database as binary data and they must be listed out in object_data_vrml file located in data directory. The program is continued by loading marker pattern file to load those markers and their assigned VRML models (virtual objects) once for validation purpose. The program will be halted if any error found in identifying marker of locating the VRML model.

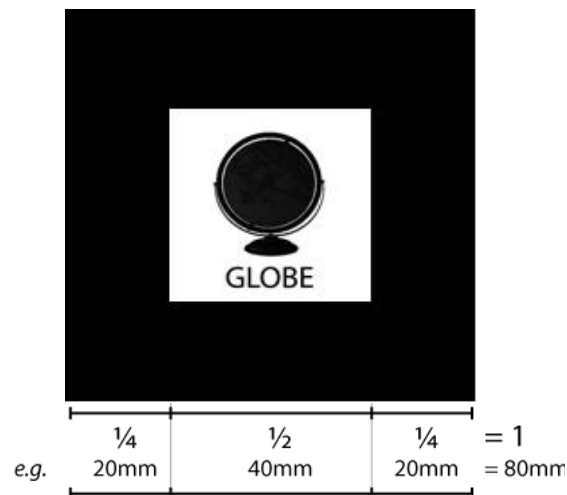


Figure 17 Marker dimensions [9]

4.2.2 ARtoolkit: Frame acquisition

OpenGL will be initialized at this section to load a window from GLUT this is where ARtoolkit video capture interacts with the windows graphic display API. After a frame is grabbed from the camera it will be output at the window loaded. A timer is set to keep track of the frame rates that determined by the user at the initialization stage.

4.2.3 ARtoolkit: Marker Detection

The live video images are grabbed from the camera for real time tracking. The image will be converted into binary image (black and white). The program will start matching the square regions in the image with the pre-trained markers. The program can detect and handle more than one marker at a time. The output from marker detection module includes the number of the detected marker, the recognition confidence value and the detected marker ID. Having the information of marker ID, the program can track the current location of the user. If there is no marker detected the process continue to grab a new frame.

Appendix D and E show the results of evaluating the marker template matching algorithm from ARtoolkit. Twenty similar patterns (Appendix C) are used to test whether the ARtoolkit is sensitive to differentiate the slight differences among all. The experiment is carried out under the different lighting conditions, white and yellow lighting condition to ensure the ability of marker detection being adaptive to different lighting conditions. From the result table, shaded area represents a “hit” (match) by the input pattern to the certain pattern. If the input and detected pattern are same which means the marker detection algorithm correctly recognize the marker. The results positively verify the sensitivity of marker detection although there is slight error is detected.

4.3 Route Planner Algorithm

The desired location selected by the user and the detected marker ID will pass to route planner. Route planner will then generate the route to link the targeted location and the current location of the user. The route planner will work according to the lookup table as well as the floor plane (Appendix) of the internal layout of the building. Route planner will search for the matrix stored in the lookup table (Table 2 and Table 3) for

the link between two locations. The corresponding route will refer to a particular VRML model ID and certain voice commands that already pre-defined. The VRML model will be the 3D navigation information that being displayed to guide the user.

Table 2 Lookup table that defines the routes between the check points in IRC ground floor

	Check Point 1	VRML ID	Check Point 2	VRML ID	Check Point 3	VRML ID
Entrance	0 (S)	5	1 (S)	9	3, 1 (S)	15
Circulation	0 (N)	0	1 (S)	9	3, 1 (S)	15
Magazine Display	1 (N)	1	1, 3 (S)	11	3, 1, 3 (S)	17
Self-check Machines	0 (N)	0	1, 3 (S)	11	3, 1(S)	15
Reading Areas	1, 2 (N)	2	2 (N)	7	2 (N)	13
Reference Shelves	1, 3 (N)	3	3 (N)	8	0 (N)	12
IT Zone	1, 3 (N)	3	3 (N)	8	0 (N)	12
To Basement	3 (N)	4	1, 2 (S)	10	3, 1, 2 (S)	16
Lift	3 (N)	4	1, 2 (S)	10	3, 1, 2 (S)	16
Newspaper Reading Corner	1, 2 (N)	2	2 (N)	7	2 (N)	13
Toilet	1 (N)	1	1 (N)	6	2 (N)	13
Discussion Rooms	1, 3 (N)	3	3 (N)	8	3 (N)	14

Table 3 Indication of the code

Code	Indication
0	Exactly at that particular point
1	Go straight
2	Turn left
3	Turn right
N	Direction shown with respect to North indicated in the map
S	Direction shown with respect to South indicated in the map

4.4 ARtoolkit: VRML Model Rendering

OpenGL and GLUT libraries are the APIs that being used for VRML model rendering, the camera transformation matrix which corresponds to the position and orientation of camera viewpoint is input to the 3D rendering module. ARtoolkit will compute the viewpoint of the camera relative to the marker. As the result the virtual object will be draw exactly align with the orientation of the detected marker. The result is shown in Figure 18. The map that is showing user's current location and undertaking route will be shown at the corner of the display as well.



Figure 18 Output display

4.5 Audio Module

AT&T Labs Natural Voices® Text-to-Speech Demo [10] is text to speech demo software from AT&T Labs. It converts the directional information in text form into WAV format to be saved as sound tracks.

OpenAL as the audio API used in the application to play the sound tracks. Number of sound track is depending on the application for example the number of the check points or depth of the information required. For the demonstration of this project, the number of sound track almost similar to the number of routes.

The algorithms of OpenAL can be summarized as the Figure 19.

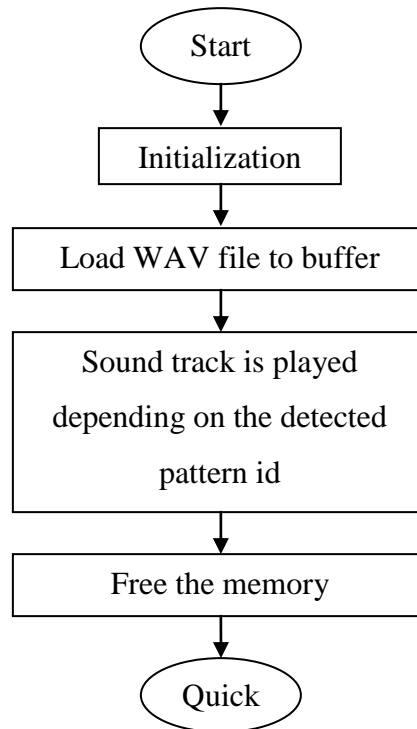


Figure 19 Flowchart of OpenAL algorithms

In initialization stage buffer and sources are being declared as well as location and velocity of the sources and listener are being set. This is how the 3D audio effect is created. Memory has to be created to hold the buffer and sources where multiple sources can be hold by a single buffer. If ARtoolkit rendering a virtual object it will continuously reading the input from the keyboard. When user press 'A' or 'a' button, the sound track that correspond to the output navigation information will be played and it will stop when 'S' or 's' is pressed. It is possible to play multiple sound tracks if more than one marker is detected. However it will create unwanted result therefore users can refresh by stopping all the sound tracks.

4.6 Overall Summary of the Result

For performance evaluation, the project has been tested at the ground floor in the Information Resource Center (IRC) of Universiti Teknologi PETRONAS.

For pre-processing phase, floor plane (Appendix) of the targeted indoor environment should be created and a matrix (Table 2 & 3) connecting the route between the checkpoints in the map must be input to the system. The route planner module will look for the route from lookup table based on the detected location ID and the destination location ID input from the user.

Figure 20 shows the marker placed at the check point 3 in IRC. The required setup for the system is a laptop and web cam. The web cam captures the live video of the condition in IRC when the user walks in IRC. The program is stated when user input a destination (Basement) at the initialization phase. Figure 21 shows the program recognizes the marker when the user reach checkpoint 3 and the route planner module computes the route from checkpoint 3 to basement. Figure 22 shows the updated map display with the route suggested by the route planner module.

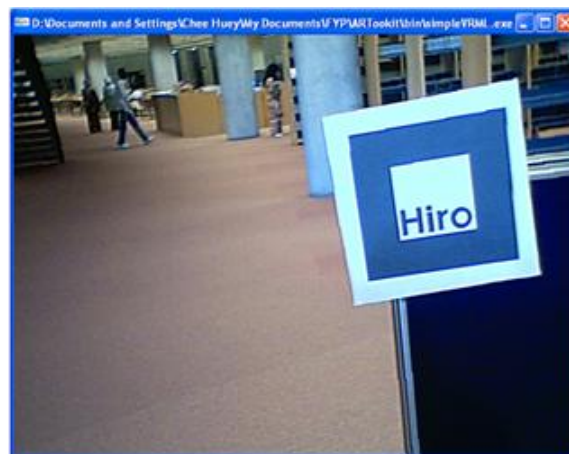


Figure 20 Marker is placed in IRC



Figure 21 Directional information is overlay on the marker

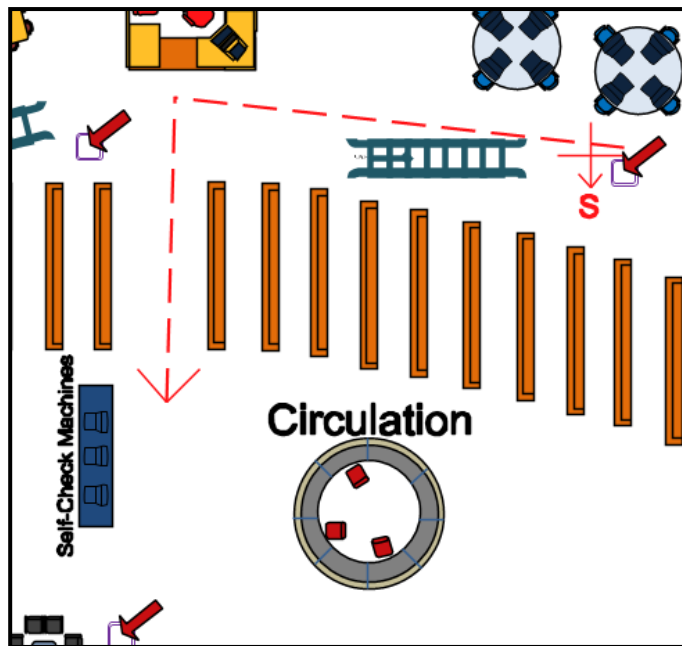


Figure 22 Updated map with current route

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The indoor positioning system that is based on augmented reality aims to deepen user's perception in perceiving the information conveyed from the map and to ease the user in identifying the route that leads to the destination. The project has been tested in IRC, and it has shown its flexibility in working as indoor positioning to navigate 12 locations which handles more than 30 possible routes. However it has room for improvement in terms of system's flexibility in handling more complicated indoor layout before it could be applied as an alternative manner of current indoor navigation solutions.

The concept of this project serves great potential for future development as it incorporate the visual enhancive technology in designing the indoor positioning technology which is also another growing technology.

5.2 Recommendations

For future enhancement, more research may focus on improving the intelligence of route planner such as shortest-distance-route computation; alternative routes based on the criteria input from the user. For example user may like to choose the routes which concentrated with restaurants in the shopping mall. However it is very dependent on the application. Therefore the flexibility of the system is very important.

Another desired improvement may focus on the accuracy of detecting and tracking a marker in the indoor environment as lighting condition in certain area may be different definitely it will affect the performance of ARtoolkit. This may be achieved by having self-adjust threshold value based on the lighting condition so that the marker could be easily being detected and tracked.

REFERENCES

- [1] B. B. Peterson, D. Bruckner, and S. Heye, “Measuring GPS Signals Indoors”, Proceedings of the Institute of Navigation’s ION GPS-2001, September, 2001.
- [2] Tim Stevens, Nokia’s Kamppi Trial succeeds at indoor positioning. <http://www.engadget.com/2009/12/22/nokias-kamppi-trial-succeeds-at-indoor-positioning-gets-shelve/> Consulted in Feb 2010.
- [3] 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN) <http://www.geometh.ethz.ch/ipin/> Consulted in Feb 2010.
- [4] Ronald T. Azum, August 1997, “A Survey of Augmented Reality”, Hughes Research Laboratories, CA 90265
- [5] Augmented Reality Project website <http://technabob.com/blog/2008/12/17/mini-augmented-reality-ads-hit-newstands/> Consulted 26 April 2010.
- [6] H. Kato, HIT Lab of University of Washington, HIT Lab NZ of University of Canterbury, New Zealand, ARtoolkit Website. Available at <http://www.hitl.washington.edu/artoolkit/> Consulted in Jan 2010.
- [7] H. Kato, M. Billinghurst, I. Poupyrev, Documentation of “ARtoolkit version 2.33”, HIT Lab of University of Washington, November 2000.
- [8] OpenAL Website. Available at

<http://connect.creativelabs.com/openal/default.aspx> Consulted in Feb 2010.

- [9] H. Kato and M. Billinghurst, Inside ARToolKit. Available at <http://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf>. Consulted in April 2010.

- [10] AT&T Labs Natural Voices® Text-to-Speech Demo. Available at <http://www2.research.att.com/~ttsweb/tts/demo.php> Consulted in June 2010.

- [11] Dr. Peter Scott, Animate Vision Principles for 3D Image Sequences, <http://www.cse.buffalo.edu/faculty/peter/cse668/> Consulted in June 2010.

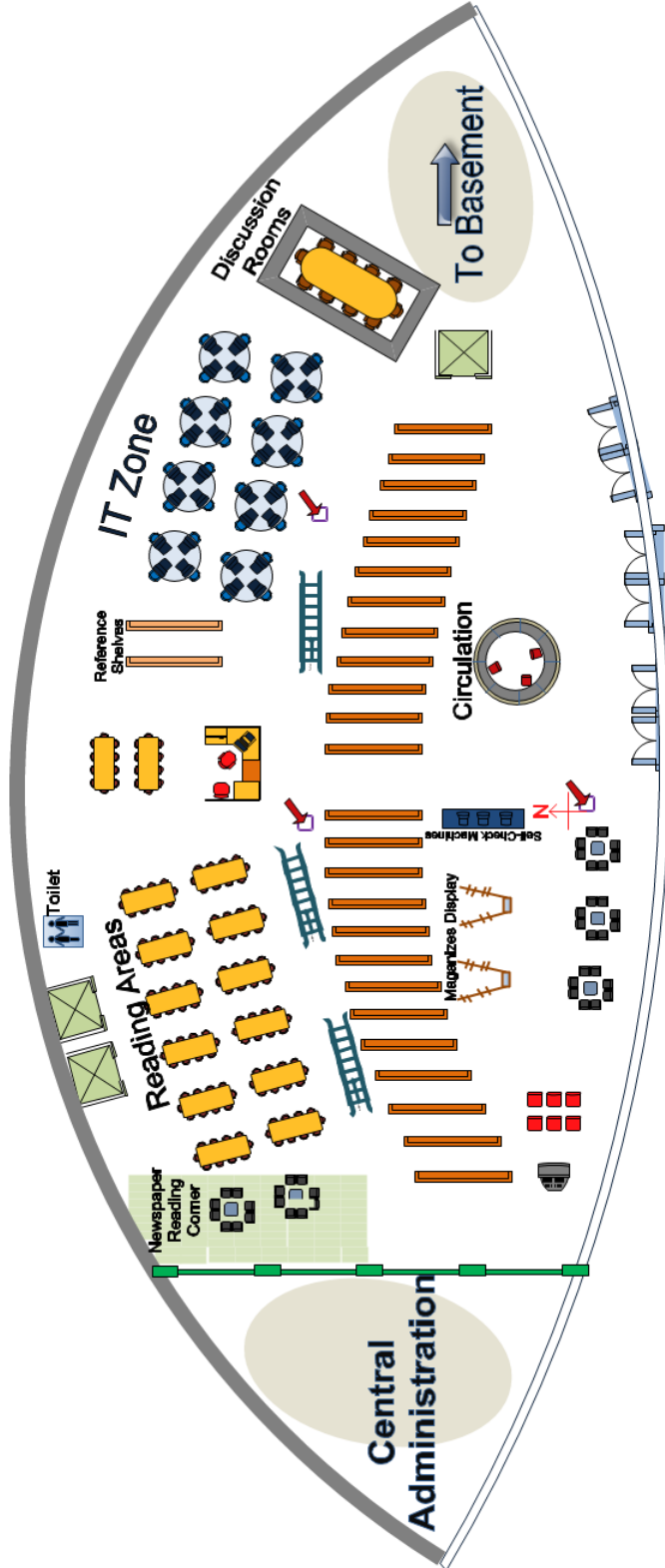
- [12] Dr. RameshRaskar, MIT Media Lab http://www.media.mit.edu/people/bio_raskar.html Consulted in June 2010.

- [13] Daniel F. Abawi, Joachim Bienwald, and Ralf Doerner, “Accuracy in optical tracking with fiducial markers: An accuracy function for ARToolkit”, In ISMAR, 2004.

APPENDICES

APPENDIX A

FLOOR PLANE OF IRC GROUND FLOOR



APPENDIX B

SOURCE CODE OF SIMPLE VRML PROGRAM

```
/*
 *   simpleVRML.c
 *
 *   Demonstration of ARToolKit with models rendered in VRML.
 *
 *   Press '?' while running for help on available key commands.
 *
 *   Copyright (c) 2002 Mark Billinghurst (MB)
grof@hitl.washington.edu
 *   Copyright (c) 2004 Raphael Grasset (RG)
raphael.grasset@hitlabnz.org.
 *   Copyright (c) 2004-2006 Philip Lamb (PRL) phil@eden.net.nz.
 *
 *   Rev          Date          Who          Changes
 *   1.0.0  ?????-??-??  MB          Original from ARToolKit
 *   1.0.1  2004-10-29  RG          Fix for ARToolKit 2.69.
 *   1.0.2  2004-11-30  PRL         Various fixes.
 * This file is part of ARToolKit.
 *
 * ARToolKit is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published
by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * ARToolKit is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with ARToolKit; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-
1307 USA
 */
//      Include
#ifdef _WIN32
#include <windows.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef __APPLE__
# include <GLUT/glut.h>
#else
# include <GL/glut.h>
#endif
```

```

#include <AR/config.h>
#include <AR/video.h>
#include <AR/param.h>           // arParamDisp()
#include <AR/ar.h>
#include <AR/gsub_lite.h>
#include <AR/arvrml.h>

#include "object.h"

#include <AL/alut.h>
#include <iostream>
#include <conio.h>
#include <AL/al.h>
#include <AL/alc.h>
//#include <AL/alut.c>
#include <time.h>
#include "bitmap.h"

// define _MT so that _beginthread( ) is available
#ifndef _MT
#define _MT
#endif
#include <process.h>
#include "resource.h"

int Dest_no;
#define MAX_LOCATION 12 //total of 12 locations in route planner LUP

/* Create checkerboard texture */
#define checkImageWidth 64
#define checkImageHeight 64
static GLubyte checkImage[checkImageHeight][checkImageWidth][4];

static int nTextureWidth;
static int nTextureHeight;
static GLuint texName;
//BYTE *map1, *map2, *map3, *CurrentMap, *map0;
BYTE* map[4]; //an array of 4 BYTE pointers
BYTE *CurrentMap, *Image;

//
=====
// Constants
//
=====

#define VIEW_SCALEFACTOR          0.025          // 1.0 ARToolkit unit
becomes 0.025 of my OpenGL units.
#define VIEW_SCALEFACTOR_1        1.0           // 1.0 ARToolkit
unit becomes 1.0 of my OpenGL units.
#define VIEW_SCALEFACTOR_4        4.0           // 1.0 ARToolkit
unit becomes 4.0 of my OpenGL units.

```

```

#define VIEW_DISTANCE_MIN          4.0           // Objects
closer to the camera than this will not be displayed.
#define VIEW_DISTANCE_MAX          4000.0       // Objects
further away from the camera than this will not be displayed.

//
=====
//   Global variables

// Preferences.
static int prefWindowed = TRUE;
static int prefWidth = 640;                    // Fullscreen
mode width.
static int prefHeight = 480;                  // Fullscreen mode
height.
static int prefDepth = 32;                    // Fullscreen
mode bit depth.
static int prefRefresh = 0;                   // Fullscreen
mode refresh rate. Set to 0 to use default rate.

// Image acquisition.
static ARUint8          *gARTImage = NULL;

// Marker detection.
static int              gARTThreshold = 100;
static long             gCallCountMarkerDetect = 0;

// Transformation matrix retrieval.
static int              gPatt_found = FALSE;   // At least one
marker.

// Drawing.
static ARParam          gARTCparam;
static ARGL_CONTEXT_SETTINGS_REF gArglSettings = NULL;

// Object Data.
static ObjectData_T    *gObjectData;
static int              gObjectDataCount;

static int detectedMarker;

//===== AUDIO =====
#define MAX_AUDIO          3
// These index the buffers and sources.
#define TEST_SOUND0        0
#define TEST_SOUND1        1
#define TEST_SOUND2        2

// Buffers hold sound data.
ALuint Buffers[MAX_AUDIO];

// Sources are points of emitting sound.
ALuint Sources[MAX_AUDIO];

```

```

// Position of the source sounds.
ALfloat SourcesPos[MAX_AUDIO][3];

// Velocity of the source sounds.
ALfloat SourcesVel[MAX_AUDIO][3];

// Position of the Listener.
ALfloat ListenerPos[] = { 0.0, 0.0, 0.0 };

// Velocity of the Listener.
ALfloat ListenerVel[] = { 0.0, 0.0, 0.0 };

// Orientation of the Listener. (first 3 elements are "at", second 3
are "up")

// Also note that these should be units of '1'.
ALfloat ListenerOri[] = { 0.0, 0.0, -1.0, 0.0, 1.0, 0.0 };

/* ALboolean LoadALData()

*          This function will load our sample data from the disk using
the Alut

*          utility and send the data into OpenAL as a buffer. A source
is then also created to play that buffer.

*/
ALboolean LoadALData()
{
    ALenum format;
    ALsizei size;
    ALvoid* data;
    ALsizei freq;
    ALboolean loop;
    int i;

    // Load wav data into buffers.
    alGenBuffers(gObjectDataCount, Buffers);

    if (alGetError() != AL_NO_ERROR)
        return AL_FALSE;

    for (i=0; i<gObjectDataCount ; i++)
    {
        fprintf(stdout, "\nTHIS%s\n", gObjectData[i].Audio_name);
        alutLoadWAVFile(gObjectData[i].Audio_name, &format, &data,
&size, &freq, &loop);
        alBufferData(Buffers[i], format, data, size, freq);
        alutUnloadWAV(format, data, size, freq);

        // Bind buffers into audio sources.
        alGenSources(gObjectDataCount, Sources);

```

```

        if (alGetError() != AL_NO_ERROR)
            return AL_FALSE;

        alSourceci (Sources[i], AL_BUFFER,    Buffers[i] );
        alSourcecf (Sources[i], AL_PITCH,    1.0           );
        alSourcecf (Sources[i], AL_GAIN,    1.0           );
        alSourcefv (Sources[i], AL_POSITION, SourcesPos[i]);
        alSourcefv (Sources[i], AL_VELOCITY, SourcesVel[i]);
        alSourceci (Sources[i], AL_LOOPING,  AL_FALSE     );
    }

    // Do another error check and return.

    if( alGetError() != AL_NO_ERROR)
        return AL_FALSE;

    return AL_TRUE;
}

void SetListenerValues()
{
    alListenerfv(AL_POSITION,    ListenerPos);
    alListenerfv(AL_VELOCITY,    ListenerVel);
    alListenerfv(AL_ORIENTATION, ListenerOri);
}

void KillALData()
{
    alDeleteBuffers(MAX_AUDIO, &Buffers[0]);
    alDeleteSources(MAX_AUDIO, &Sources[0]);
    alutExit();
}

//
//=====
//    Functions
//=====

static int setupCamera(const char *cparam_name, char *vconf, ARParam
*cparam)
{
    ARParam          wparam;
    int              xsize, ysize;

    // Open the video path.
    if (arVideoOpen(vconf) < 0) {
        fprintf(stderr, "setupCamera(): Unable to open connection to
camera.\n");
        return (FALSE);
    }
}

```

```

    }

    // Find the size of the window.
    if (arVideoInqSize(&xsize, &ysize) < 0) return (FALSE);
    fprintf(stdout, "Camera image size (x,y) = (%d,%d)\n", xsize,
ysize);

    // Load the camera parameters, resize for the window and init.
    if (arParamLoad(cparam_name, 1, &wparam) < 0) {
        fprintf(stderr, "setupCamera(): Error loading parameter
file %s for camera.\n", cparam_name);
        return (FALSE);
    }
    arParamChangeSize(&wparam, xsize, ysize, cparam);
    fprintf(stdout, "*** Camera Parameter ***\n");
    arParamDisp(cparam);

    arInitCparam(cparam);

    if (arVideoCapStart() != 0) {
        fprintf(stderr, "setupCamera(): Unable to begin camera data
capture.\n");
        return (FALSE);
    }

    return (TRUE);
}

static int setupMarkersObjects(char *objectDataFilename[])
{
    // Load in the object data - trained markers and associated
bitmap files.
    if ((gObjectData = read_VRMLdata(objectDataFilename[Dest_no-1],
&gObjectDataCount)) == NULL) {
        fprintf(stderr, "setupMarkersObjects(): read_VRMLdata returned
error !!\n");
        return (FALSE);
    }

    printf("Object count = %d\n", gObjectDataCount);

    return (TRUE);
}

// Report state of ARToolKit global variables arFittingMode,
// arImageProcMode, arglDrawMode, arTemplateMatchingMode,
arMatchingPCAMode.
static void debugReportMode(void)
{
    if(arFittingMode == AR_FITTING_TO_INPUT ) {
        fprintf(stderr, "FittingMode (Z): INPUT IMAGE\n");
    } else {
        fprintf(stderr, "FittingMode (Z): COMPENSATED IMAGE\n");
    }
}

```



```

    if( arImageProcMode == AR_IMAGE_PROC_IN_FULL ) {
        fprintf(stderr, "ProcMode (X) : FULL IMAGE\n");
    } else {
        fprintf(stderr, "ProcMode (X) : HALF IMAGE\n");
    }

    if (arglDrawModeGet(gArglSettings) == AR_DRAW_BY_GL_DRAW_PIXELS)
    {
        fprintf(stderr, "DrawMode (C) : GL_DRAW_PIXELS\n");
    } else if (arglTexmapModeGet(gArglSettings) ==
AR_DRAW_TEXTURE_FULL_IMAGE) {
        fprintf(stderr, "DrawMode (C) : TEXTURE MAPPING (FULL
RESOLUTION)\n");
    } else {
        fprintf(stderr, "DrawMode (C) : TEXTURE MAPPING (HALF
RESOLUTION)\n");
    }

    if( arTemplateMatchingMode == AR_TEMPLATE_MATCHING_COLOR ) {
        fprintf(stderr, "TemplateMatchingMode (M) : Color
Template\n");
    } else {
        fprintf(stderr, "TemplateMatchingMode (M) : BW
Template\n");
    }

    if( arMatchingPCAMode == AR_MATCHING_WITHOUT_PCA ) {
        fprintf(stderr, "MatchingPCAMode (P) : Without PCA\n");
    } else {
        fprintf(stderr, "MatchingPCAMode (P) : With PCA\n");
    }
}

static void Quit(void)
{
    arglCleanup(gArglSettings);
    arVideoCapStop();
    arVideoClose();
#ifdef _WIN32
    CoUninitialize();
#endif
    exit(0);
}

static void Keyboard(unsigned char key, int x, int y)
{
    int mode;
    switch (key) {
        case 0x1B: // Quit.
        case 'Q':
        case 'q':
            Quit();
            break;
    }
}

```

```

        case 'C':
        case 'c':
            mode = arglDrawModeGet(gArglSettings);
            if (mode == AR_DRAW_BY_GL_DRAW_PIXELS) {
                arglDrawModeSet(gArglSettings,
AR_DRAW_BY_TEXTURE_MAPPING);
                arglTexmapModeSet(gArglSettings,
AR_DRAW_TEXTURE_FULL_IMAGE);
            } else {
                mode = arglTexmapModeGet(gArglSettings);
                if (mode == AR_DRAW_TEXTURE_FULL_IMAGE)
                    arglTexmapModeSet(gArglSettings, AR_DRAW_TEXTURE_HALF_IMAGE);
                else arglDrawModeSet(gArglSettings,
AR_DRAW_BY_GL_DRAW_PIXELS);
            }
            fprintf(stderr, "*** Camera - %f (frame/sec)\n",
(double)gCallCountMarkerDetect/arUtilTimer());
            gCallCountMarkerDetect = 0;
            arUtilTimerReset();
            debugReportMode();
            break;
        case '?':
        case '/':
            printf("Keys:\n");
            printf(" q or [esc]      Quit demo.\n");
            printf(" c                Change arglDrawMode and
arglTexmapMode.\n");
            printf(" ? or /          Show this help.\n");
            printf("\nAdditionally, the ARVideo library supplied
the following help text:\n");
            arVideoDispOption();
            break;
        case 'a':
        case 'A':
            if (gPatt_found)
            {
                switch (gObjectData[detectedMarker].vrml_id)
                {
                    case 0: alSourcePlay(Sources[TEST_SOUND2]);
                            break;

                    case 1: alSourcePlay(Sources[TEST_SOUND1]);
                            break;

                    default:
                            break;
                }
            }
            else
                alSourcePlay(Sources[TEST_SOUND0]);

            break;

```

```

        case 's':
        case 'S':
            if (gPatt_found)
            {
                alSourceStop(Sources[TEST_SOUND0]);
                alSourceStop(Sources[TEST_SOUND1]);
                alSourceStop(Sources[TEST_SOUND2]);
            }
            break;
        default:
            break;
    }
}

static void Idle(void)
{
    static int ms_prev;
    int ms;
    float s_elapsed;
    ARUint8 *image;

    ARMarkerInfo *marker_info;
    int marker_num;
    int i, j, k;

    // Find out how long since Idle() last ran.
    ms = glutGet(GLUT_ELAPSED_TIME);
    s_elapsed = (float)(ms - ms_prev) * 0.001;
    if (s_elapsed < 0.01f) return; // Don't update more often than
100 Hz.
    ms_prev = ms;

    // Update drawing.
    arVrmlTimerUpdate();

    // Grab a video frame.
    if ((image = arVideoGetImage()) != NULL) {
        gARTImage = image; // Save the fetched image.
        gPatt_found = FALSE; // Invalidate any previous
detected markers.

        gCallCountMarkerDetect++; // Increment ARToolkit FPS
counter.

        // Detect the markers in the video frame.
        if (arDetectMarker(gARTImage, gARTThreshhold,
&marker_info, &marker_num) < 0) {
            exit(-1);
        }

        // Check for object visibility.

        for (i = 0; i < gObjectDataCount; i++) {

```

```

confidence          // Check through the marker_info array for highest
                    // visible marker matching our object's pattern.
                    k = -1;
                    for (j = 0; j < marker_num; j++) {
                        if (marker_info[j].id == gObjectData[i].id) {
                            if( k == -1 ) k = j; // First marker
detected.
                                else if (marker_info[k].cf <
marker_info[j].cf) k = j; // Higher confidence marker detected.
                            }
                        }
                    }

                    if (k != -1) {
                        // Get the transformation between the marker
and the real camera.
                        //fprintf(stderr, "Saw object %d.\n", i);
                        if (gObjectData[i].visible == 0) {
                            arGetTransMat(&marker_info[k],
gObjectData[i].marker_center, gObjectData[i].marker_width,
gObjectData[i].trans);
                        } else {
                            arGetTransMatCont(&marker_info[k],
gObjectData[i].trans,
gObjectData[i].marker_center, gObjectData[i].marker_width,
gObjectData[i].trans);
                        }
                        gObjectData[i].visible = 1;
                        g Patt_found = TRUE;
                    } else {
                        gObjectData[i].visible = 0;
                    }
                }

                // Tell GLUT to update the display.
                glutPostRedisplay();
            }
        }

//
// This function is called on events when the visibility of the
// GLUT window changes (including when it first becomes visible).
static void Visibility(int visible)
{
    if (visible == GLUT_VISIBLE) {
        glutIdleFunc(Idle);
    } else {
        glutIdleFunc(NULL);
    }
}

```

```

}

void renderBitmapString(
    float x,
    float y,
    void *font,
    char *string) {
    char *c;
    glRasterPos2f(x, y);
    for (c=string; *c != '\0'; c++) {
        glutBitmapCharacter(font, *c);
    }
}
// This function is called when the
// GLUT window is resized.
static void Reshape(int w, int h)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    // Call through to anyone else who needs to know about window
    // sizing here.
}
static void Display(void)
{
    int i;
    GLdouble p[16];
    GLdouble m[16];

    // Select correct buffer for this context.
    glDrawBuffer(GL_BACK);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear the
    buffers for new frame.

    arglDispImage(gARTImage, &gARTCparam, 1.0, gArglSettings); //
    zoom = 1.0.
    arVideoCapNext();
    gARTImage = NULL; // Image data is no longer valid after calling
    arVideoCapNext().

    if (gPatt_found) {
        // Projection transformation.
        arglCameraFrustumRH(&gARTCparam, VIEW_DISTANCE_MIN,
        VIEW_DISTANCE_MAX, p);
        glMatrixMode(GL_PROJECTION);
        glLoadMatrixd(p);
        glMatrixMode(GL_MODELVIEW);
    }
}

```

```

        // Viewing transformation.
        glLoadIdentity();

        // Lighting and geometry that moves with the camera should
go here.
        // (I.e. must be specified before viewing
transformations.)
        //none

        // All other lighting and geometry goes here.
        // Calculate the camera position for each object and draw
it.
        for (i = 0; i < gObjectDataCount; i++) {
            if ((gObjectData[i].visible != 0) &&
(gObjectData[i].vrm_l_id >= 0)) {
                //fprintf(stderr, "About to draw object %i\n",
i);
                arglCameraViewRH(gObjectData[i].trans, m,
VIEW_SCALEFACTOR_4);
                glLoadMatrixd(m);
                arVrmlDraw(gObjectData[i].vrm_l_id);

                detectedMarker = gObjectData[i].vrm_l_id;

                // set the map to be load
                CurrentMap = map[1+gObjectData[i].vrm_l_id];

            }
        }
    } // gPatt_found
    //else CurrentMap = map0;

    // Any 2D overlays go here.
    // Draw Text
    //// switch to projection mode
    glMatrixMode(GL_PROJECTION);

    //// save previous matrix which contains the
    ////settings for the perspective projection
    glPushMatrix();

    // reset matrix
    glLoadIdentity();

    gluOrtho2D(-300, 300, -200, 200 );

    // switch back to modelview mode
    glMatrixMode(GL_MODELVIEW);
    glPushMatrix(); // to save the previous setting if gPatt is
found
    glLoadIdentity();

```

```

        renderBitmapString(200,180,GLUT_BITMAP_HELVETICA_18,"Destination
:");
        if (Dest_no == 1)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Entrance");
        else if (Dest_no == 2)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Circulation
");
        else if (Dest_no == 3)
            renderBitmapString(150,160,GLUT_BITMAP_HELVETICA_18,"Magazine
Display");
        else if (Dest_no == 4)
            renderBitmapString(130,160,GLUT_BITMAP_HELVETICA_18,"Self-check
Machines");
        else if (Dest_no == 5)
            renderBitmapString(150,160,GLUT_BITMAP_HELVETICA_18,"Reading
Areas");
        else if (Dest_no == 6)
            renderBitmapString(130,160,GLUT_BITMAP_HELVETICA_18,"Reference
Shelves");
        else if (Dest_no == 7)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"IT Zone");
        else if (Dest_no == 8)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Basement");
        else if (Dest_no == 9)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Lift");
        else if (Dest_no == 10)
            renderBitmapString(110,160,GLUT_BITMAP_HELVETICA_18,"Newspaper
Reading Corner");
        else if (Dest_no == 11)
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Washrooms")
;
        else if (Dest_no == 12)
            renderBitmapString(130,160,GLUT_BITMAP_HELVETICA_18,"Discussion
Rooms");
        else
            renderBitmapString(200,160,GLUT_BITMAP_HELVETICA_18,"Location
X");

        glDisable (GL_DEPTH_TEST);
        glDisable (GL_TEXTURE_2D);

        glRasterPos2d (-295, -195);//left bottom corner
        glDrawPixels (256, 128, GL_BGR_EXT, GL_UNSIGNED_BYTE,
CurrentMap);

        glMatrixMode (GL_PROJECTION);
        glPopMatrix();
        glMatrixMode (GL_MODELVIEW);
        glPopMatrix();

        glutSwapBuffers();
}

```

```

int Load_BMP_Img (char *szFileName, BYTE **pBitmapData)
{
    HANDLE hFileHandle;
    BITMAPINFO *pBitmapInfo = NULL;

    unsigned long lInfoSize = 0, lBitSize = 0;
    int nTextureWidth, nTextureHeight;

    static short VERBOSE = TRUE, first = TRUE;

    hFileHandle = CreateFile (szFileName, GENERIC_READ,
FILE_SHARE_READ, NULL,
                                OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN,
NULL);

    if (hFileHandle == INVALID_HANDLE_VALUE) return FALSE;

    BITMAPFILEHEADER bitmapHeader;
    DWORD dwBytes;

    ReadFile (hFileHandle, &bitmapHeader, sizeof(BITMAPFILEHEADER),
                                &dwBytes, NULL);

    if (dwBytes != sizeof(BITMAPFILEHEADER)) return FALSE;

    if (bitmapHeader.bfType != 'MB') return FALSE;

    lInfoSize = bitmapHeader.bfOffBits - sizeof(BITMAPFILEHEADER);
    pBitmapInfo = (BITMAPINFO *) new BYTE[lInfoSize];

    ReadFile (hFileHandle, pBitmapInfo, lInfoSize, &dwBytes, NULL);

    if (dwBytes != lInfoSize) return FALSE;

    nTextureWidth = pBitmapInfo->bmiHeader.biWidth;
    nTextureHeight = pBitmapInfo->bmiHeader.biHeight;
    lBitSize = pBitmapInfo->bmiHeader.biSizeImage;

    if (VERBOSE) {
        if (first) {
            first = FALSE;
            printf ("\n");
        }
        printf ("    Image '%s' is:  %d by %d pixels\n", szFileName,
                                nTextureWidth, nTextureHeight);
    }

    if (lBitSize == 0) lBitSize = (nTextureWidth *
                                pBitmapInfo->bmiHeader.biBitCount + 7) / 8 *
                                abs(nTextureHeight);

    *pBitmapData = new BYTE[lBitSize];
}

```



```

ReadFile (hFileHandle, *pBitmapData, lBitSize, &dwBytes, NULL);

if (lBitSize != dwBytes)
{
    if (*pBitmapData) delete [] (BYTE *) *pBitmapData;
    *pBitmapData = NULL;
    return FALSE;
}

CloseHandle (hFileHandle);

return TRUE;
}

int main(int argc, char** argv)
{
    int i, j;
    char glutGamemode[32];
    const char *cparam_name =
        "Data\\camera_para.dat";
    //const char *mapFile = "Data\\IMGirc4.png";
#ifdef _WIN32
    char          *vconf = "Data\\WDM_camera_flipV.xml";
#else
    char          *vconf = "";
#endif
    //char objectDataFilename[] = "Data\\object_data_vrml";
    char* objectDataFilename[MAX_LOCATION]=
    {
        ("Data\\object_data_vrml_0"),
        ("Data\\object_data_vrml_1"),
        ("Data\\object_data_vrml_2"),
        ("Data\\object_data_vrml_3"),
        ("Data\\object_data_vrml_4"),
        ("Data\\object_data_vrml_5"),
        ("Data\\object_data_vrml_6"),
        ("Data\\object_data_vrml_7"),
        ("Data\\object_data_vrml_8"),
        ("Data\\object_data_vrml_9"),
        ("Data\\object_data_vrml_10"),
        ("Data\\object_data_vrml_11")
    };
    // Library inits.
    //

    glutInit(&argc, argv);

    // Initialize OpenAL and clear the error bit.
    alutInit(NULL, 0);
    alGetError();

    // Hardware setup.
    //

```

```

        if (!setupCamera(cparam_name, vconf, &gARTCparam)) {
            fprintf(stderr, "main(): Unable to set up AR (cparam_name)
camera.\n");
            exit(-1);
        }

#ifdef _WIN32
        CoInitialize(NULL);
#endif

        // Library setup.
        // Set up GL context(s) for OpenGL to draw into.
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
        if (!prefWindowed) {
            if (prefRefresh) sprintf(glutGamemode, "%ix%i:%i@%i",
prefWidth, prefHeight, prefDepth, prefRefresh);
            else sprintf(glutGamemode, "%ix%i:%i", prefWidth,
prefHeight, prefDepth);
            glutGameModeString(glutGamemode);
            glutEnterGameMode();
        } else {
            glutInitWindowSize(gARTCparam.xsize, gARTCparam.ysize);
            glutCreateWindow(argv[0]);
        }

        // Setup argl library for current context.
        if ((gArglSettings = arglSetupForCurrentContext()) == NULL) {
            fprintf(stderr, "main(): arglSetupForCurrentContext()
returned error.\n");
            exit(-1);
        }
        debugReportMode();
        arUtilTimerReset();

        fprintf(stdout,
"*****\n");
        fprintf(stdout, " Please Select Your Destination\n");
        fprintf(stdout, " 1. Entrance\n");
        fprintf(stdout, " 2. Circulation\n");
        fprintf(stdout, " 3. Magazine Display\n");
        fprintf(stdout, " 4. Self-check Machines\n");
        fprintf(stdout, " 5. Reading Areas\n");
        fprintf(stdout, " 6. Reference Shelves\n");
        fprintf(stdout, " 7. IT Zone\n");
        fprintf(stdout, " 8. Basement\n");
        fprintf(stdout, " 9. Lift\n");
        fprintf(stdout, " 10. Newspaper Reading Corner\n");
        fprintf(stdout, " 11. Washrooms\n");
        fprintf(stdout, " 12. Discussion Rooms\n");
        fprintf(stdout, " Please Press the number of your desired
destination\n");
        fscanf (stdin, "%d",&Dest_no);

```

```

        fprintf(stdout, "the chosen destination No. is %d\n", Dest_no);
        fprintf(stdout,
"*****\n");
        if (!setupMarkersObjects(objectDataFilename))
        {
            fprintf(stderr, "main(): Unable to
set up AR objects and markers.\n");
            Quit();
        }

        // Load the wav data.
        if (LoadALData() == AL_FALSE)
            fprintf(stdout, "main(): Unable to load the wave data\n");
        else {
            fprintf(stdout, "\nmain(): Sucessfully loaded the wave
data\n");
            SetListenerValues();
            // Setup an exit procedure.
            atexit(KillALData);
        }

        //initTexture(mapFile);

        Load_BMP_Img ("Data\\maps\\OriMap.bmp", &map[0]);
        CurrentMap = map[0]; // initialize map
        for (j=0; j<gObjectDataCount ; j++)
        {
            Load_BMP_Img (gObjectData[j].Map_name, &Image);
            map[j+1] = Image;
        }

        fprintf(stdout, "Pre-rendering the VRML objects...");
        fflush(stdout);
        glEnable(GL_TEXTURE_2D);
        for (i = 0; i < gObjectDataCount; i++) {
            arVrmlDraw(gObjectData[i].vrml_id);
        }
        glDisable(GL_TEXTURE_2D);
        fprintf(stdout, " done\n");

        // Register GLUT event-handling callbacks.
        // NB: Idle() is registered by Visibility.
        glutDisplayFunc (Display);
        glutReshapeFunc (Reshape);
        glutVisibilityFunc (Visibility);
        glutKeyboardFunc (Keyboard);

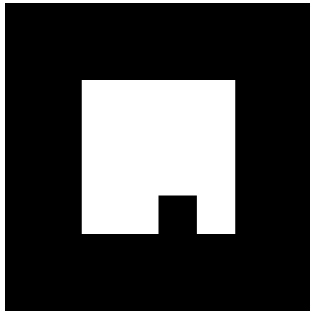
        glutMainLoop();
        return (0);
}

```

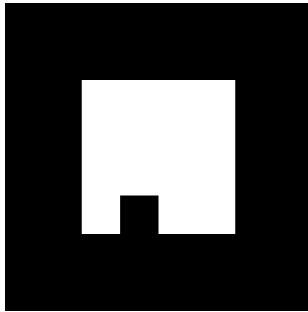
APPENDIX C

PATTERNS USED TO EVALUATE MARKER MATCHING ALGORITHM

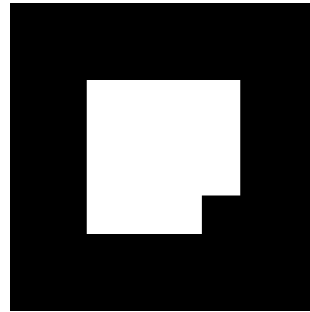
Pattern 1



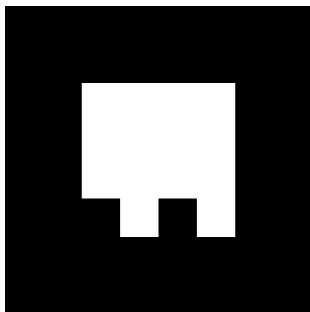
Pattern 2



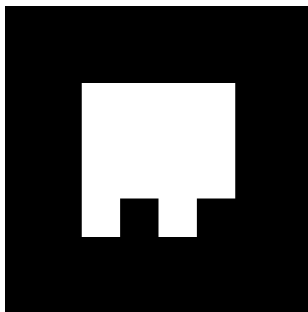
Pattern 3



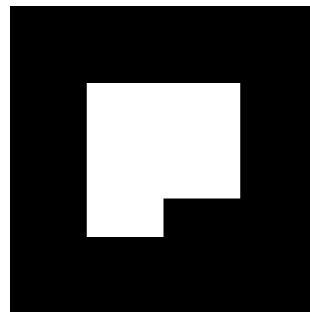
Pattern 4



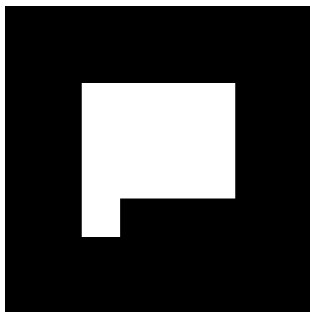
Pattern 5



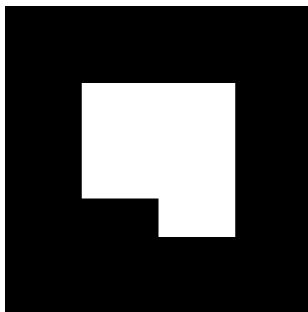
Pattern 6



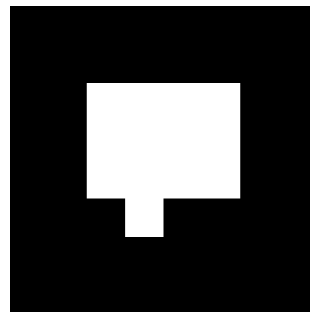
Pattern 7



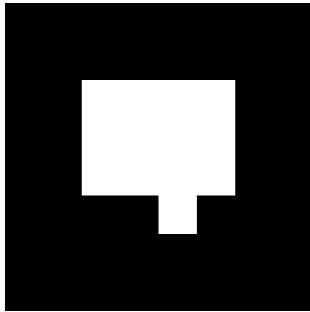
Pattern 8



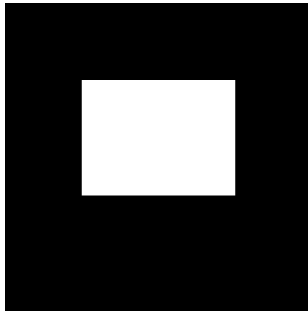
Pattern 9



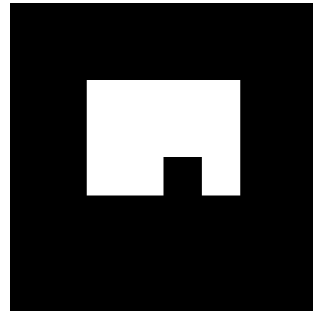
Pattern 10



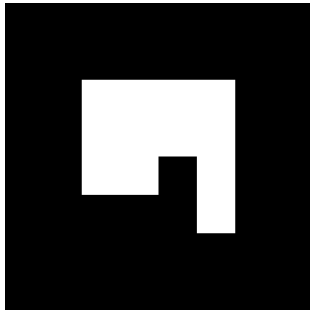
Pattern 11



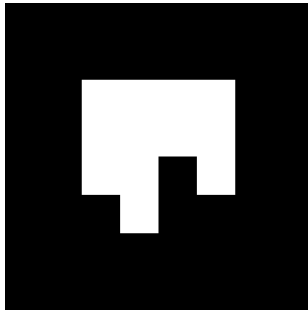
Pattern 12



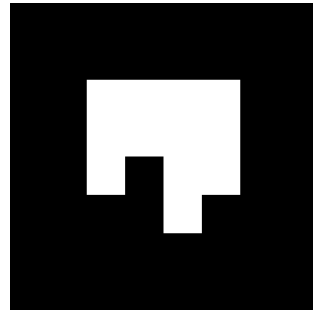
Pattern 13



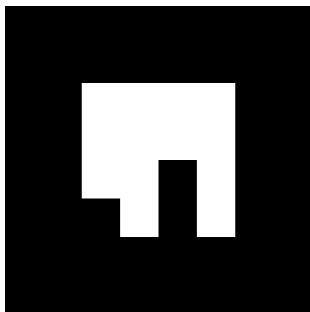
Pattern 14



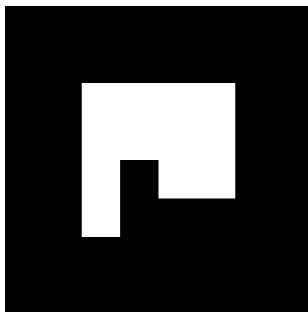
Pattern 15



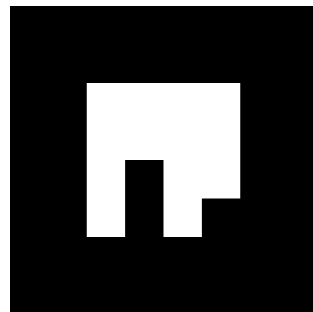
Pattern 16



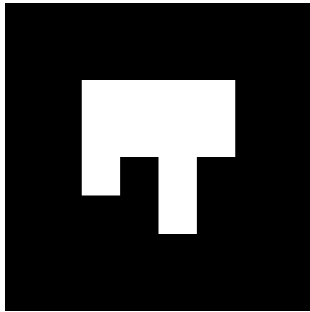
Pattern 17



Pattern 18



Pattern 19



Pattern 20

