Indoor Localization and Guidance using Augmented Reality Toolbox

by

Muhammad Adib Idzam Bin Mohd Zamri

ID: 18456

Dissertation submitted in partial fulfilment of the requirements for the

Bachelor of Engineering (Hons)

(Electrical & Electronic Engineering)

JANUARY 2017

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Indoor Localization and Guidance using Augmented Reality Toolbox

By

Muhammad Adib Idzam Bin Mohd Zamri

18456

A project dissertation submitted to the

Electrical & Electronic Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONIC)

Approved by,

____

(**Patrick Sebastian**)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2006

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.

[illegible]

MUHAMMAD ADIB IDZAM BIN MOHD ZAMRI

ABSTRACT

In this era, new technology were used to ease user. Outdoor guidance uses Global Positioning System (GPS) to pin point one's location and also and navigate him through the map which downloaded from the Internet. Even so, outdoor guidance is not effectively to apply to indoor guidance. The prime objective of this project is to develop localized indoor guidance system which run in Augmented Reality (AR) mode. AR will provide necessary information by overlaying them on the actual environment captured by camera mounted on a smart glasses and able user to guide through a specific place. Microcontroller Raspberry Pi is used to process the whole system. Instead just showing user the directions, this project also focuses on embedded information on the markers that generate AR images. This additional information can be obtained by user while going through the navigation. This project also uses object detection to create indicators that can lead the user to the markers. Using the object detection also, air-gesture system will be created to replace the existing mouse and keyboard. This will increase the mobility. Project had been tested at Universiti Teknologi PETRONAS's Chancellor Complex and showed the high mobility and functionality work as localized indoor guidance.

ACKNOWLEDGEMENT

I would like to thank you to all persons that helped and guided me throughout completing this project. First of all, I would like to thank to my supervisor, Mr. Patrick Sebastian for his support and guidance in making this project successful. He has contributed a lot in this project in term of giving idea and suggestions which to make the project to be outstanding.

I also would like to thanks to my friends including post graduates students for guiding me on how to setup OpenCV and the guide on how to use Raspberry Pi board.

Not forgotten, thank you also to my family members and other people who supported me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

LIST OF ABBREVIATIONS

AR	: Augmented Reality
IRC	: Information Resource Centre
GPS	: Global Positioning System
WLAN	: Wireless Local Area Network
RFID	: Radio Frequency Identification
UTP	: Universiti Teknologi PETRONAS
QR	: Quick Response
CH	: Chancellor Complex
RANSAC	: Random Sample Consensus

1 CHAPTER 1: INTRODUCTION

1.1 Background

The title of project is **Indoor Localization and Guidance using Augmented Reality Toolbox.** Indoor localization is locating or real time tracking of physical body inside a closed environment. Guidance here refers to navigation and methods. The project aims to create an Augmented Reality (AR) based guiding system and able to guide user indoor by tracking special markers. In addition, this project uses image processing to identify marker indicator and shows their location. The marker will consist of Quick Response (QR) code which contain embedded information. The markers however will prompt AR image with additional information to guide user to find his destination. Besides that, this project also implement on air selection menu which replaces with conventional use of mouse.

1.2 Problem Statement

Current navigation tools such as Global Positioning System (GPS) unable to pinpoint indoor destination accurately due to satellite signals being interrupted by walls and structures. For indoor, wireless technology such as Global System for Mobile (GSM), Bluetooth, infrared, Wireless Local Area Network (WLAN) and Radio Frequency Identification (RFID) can be used. Even so, the comparison by [1] indicates that these technologies are not suitable to be used in indoor guidance due to either their accuracy, signal error rate or range.

Wireless technology has limitation on signal and area of coverage, therefore indoor guidance still relies on non-interactive and conventional guidance such as map and signboard.

1.3 Objectives

The objective of this project are:

1. To develop interactive localized indoor guidance system using raspberry pi
2. To develop AR in localized indoor guidance system
3. To develop long range marker indicator finder.
4. To develop on air menu selection.
5. To develop embedded information in markers using QR code.

1.4 Scope of study

Project Location

This project conducted in indoor condition. The target location is at Universiti Teknologi PETRONAS (UTP) Chancellor Complex. The area is divided into 3 sections such as in FIGURE 14 and each sections are corresponds to actual point of interests which are the UTP Information Resource Centre (UTP IRC), UTP Chancellor Complex and UTP Chancellor Hall. These sections are named “Library”, “Home” and “CH” respectively.

Image Capture

This project uses a web camera mounted on a smart glasses. The web camera is 5 megapixel. For earlier stage, the image captured was sent to a laptop for image processing using Visual Studio 2013 software running with OpenCV version 2.4.13. Later on, the image will be processed by microcontroller Raspberry Pi instead of using laptop.

Software Development Kit (SDK)

This project uses open source SDK. Visual Studio 2013 is used as integrated development environment (IDE). OpenCV SDK is used for image capturing and processing. AR rendering uses ARToolkit SDK and ZBAR SDK is used for QR code reader.

2 CHAPTER 2: LITERATURE REVIEW

2.1 Augmented Reality (AR)

Based on [2], AR is the user's real time environment which integrates with digital information. Compared to Virtual Reality (VR), VR creates an artificial environment whereas AR overlays information on top of existing surrounding. This project will use ARToolKIt to create AR. ARToolKIT is applicable on stereos and optical see-through support. It is integrated with hardware such as smart glasses and new devices.

Basic principal of this software is that it works by identifying specific squares shape as tracking markers. When starting, image snapshot is captured by camera will be processed by Visual Studio. The software will then scan and locate the stacking marker squares. When the square is found, the distance between the square and the camera is calculated. Comparing pattern with in memory templates, a 3D virtual object is positioned aligned with the marker [2]. The image is rendered and streamed to the user screen together with additional information.

2.2 Marker Detection

AR applications will detect marker and compare it with stored marker data. After comparison, it will generate AR image based on the marker. Referring guide from [3], [4] , there are 6 steps to detect marker using C++ algorithm. This method uses the convolution of derivative of Gaussian to create an estimation component of the gradient intensity on each scan line. Therefore, the black/white edges in the image is detected. Edges can be found when local maxima which located along the scan lines is higher value than a certain threshold.

Gaussian derivative kernel used [3]:

Equation 1 Gaussian derivative kernel:

[-3 -5 0 5 3] * A

Next, Sobel operation is used to identify the orientation of the edge. The Sobel operator [4] :

Equation 2 Sobel operation:

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

Furthermore, the edge found earlier are combined together into line by using Random Sample Consensus (RANSAC)-grouper [3], [4]. Orientation of the edges able to give the line segments orientations. Next, the lines is extended along the edges to detect the corner of the marker. Remove the lines without corner and chain the filtered lines. Chain them together which consist of four lines and with four corners. The chain will have black region inside and probably contain marker. From this, AR tool is used to detect marker and create AR images.

2.3 Object Detection on Marker Indicators.

Finding marker's location will be very easy with some indicators. In this project, an object detection method will be used to identify the indicators. This indicator will indicate that marker is very closely, which help user to spot the marker easier without wondering around. For industrial use, detecting object location in digital image has become very important use to save time and ease user [5].

This object detection uses colour processing. Image are captured by camera and sent to the OpenCV and perform object recognition processing which undergoes Red Green Blue (RGB) adjustment, elimination of unwanted colour, gray scaling and circular Hough Transform [5]. Image shows the result of [5] object detection.

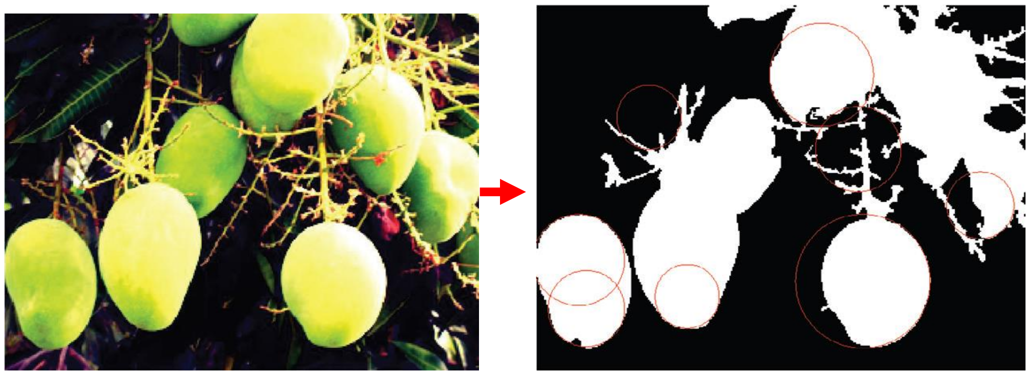


FIGURE 1 Object detection

Rosebrock [6] shown that using cv2.findContours in OpenCV to find the outline of the object in binary mask. Referring to his guide in OpenCV program, object detection and its movement can be traced for user to see.

2.4 QR Coded Markers

Compared to conventional marker, this project uses QR coded markers. QR code was developed by Denso Wave and approved as an ISO international standard (ISO/IEC18004) [7]. Nowadays, QR code is widely used in advertising by displaying the code and using QR code reader software, user can scan the code and convert it to useful form such as Uniform Resource Locator (URL) for a website [8].

FIGURE 2 QR code sample

QR code consist of 3 timing patterns (big black squares) in the corners except 1 side at bottom right. By this, the QR code in proper orientation is as shown in FIGURE 2. The version of the code is determined by total sum of modules across the QR code is subtract by 17 and divided by 4 [9].

Equation 3 QR code version:

$$\text{Version} = (\text{Sum of modules} - 17)/4$$

After that, the code's format is determined by 15 bit long: 5 bit for format information and 10 bit for error correction. The first 5 bits of the format hold the error correction level of 2 bits and data mask of 3 bits. The 5 bit XOR with 10101 and the last 3 bits is the mask number. To retrieve data, the computer unmasks the original data bytes.

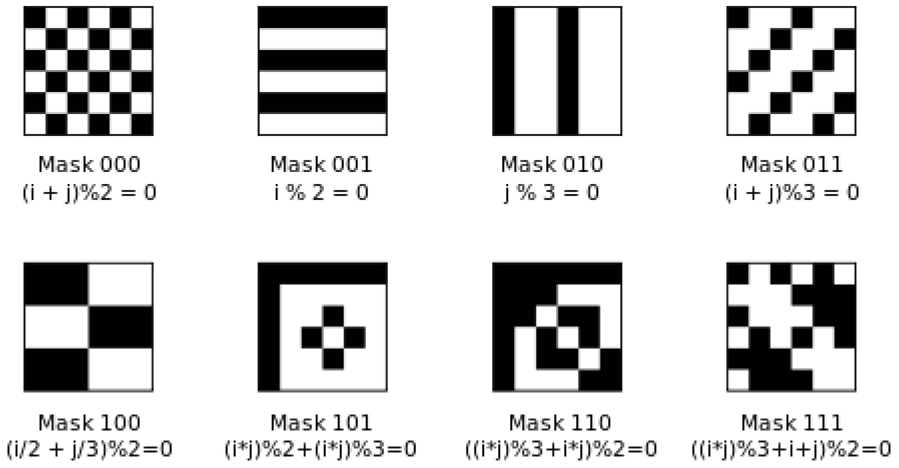


FIGURE 3 QR Code Masks

The header contain encoding type and length of data byte is store. The encoding type is stored bottom right of 4 bits, starting from bottom right and continues in zig-zag motion from left to right [9].

For each encoding type, the number of bits as follows:

TABLE 1 Encoding type

Encoding Type	Number of bits
Numeric	10
Alphanumeric	9
8-bit Byte	8

Then the bit length of the message is decoded then followed by characters in zig-zag motion as shown in FIGURE 4.

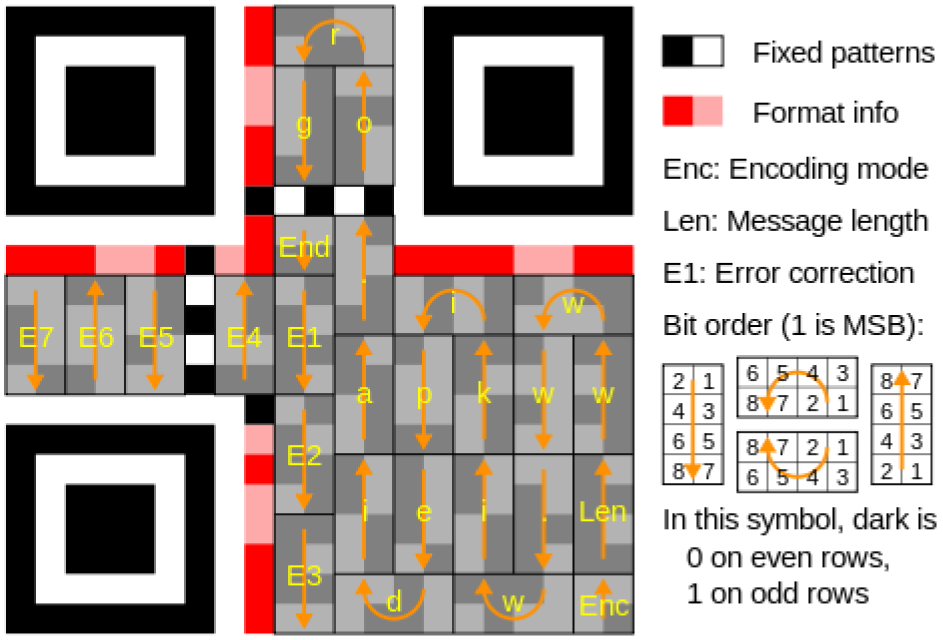


FIGURE 4 QR code decoding

2.5 Critical Analysis

Fadzly [1] uses AR to navigate user indoor. He uses ARToolKit as it is compatible with Raspberry Pi operating system, Linux. This project almost similar to the project, however this project focuses more on creating indicators for markers location, QR coded marker, and on air menu selection.

3 CHAPTER 3: METHODOLOGY

3.1 Project Flowchart

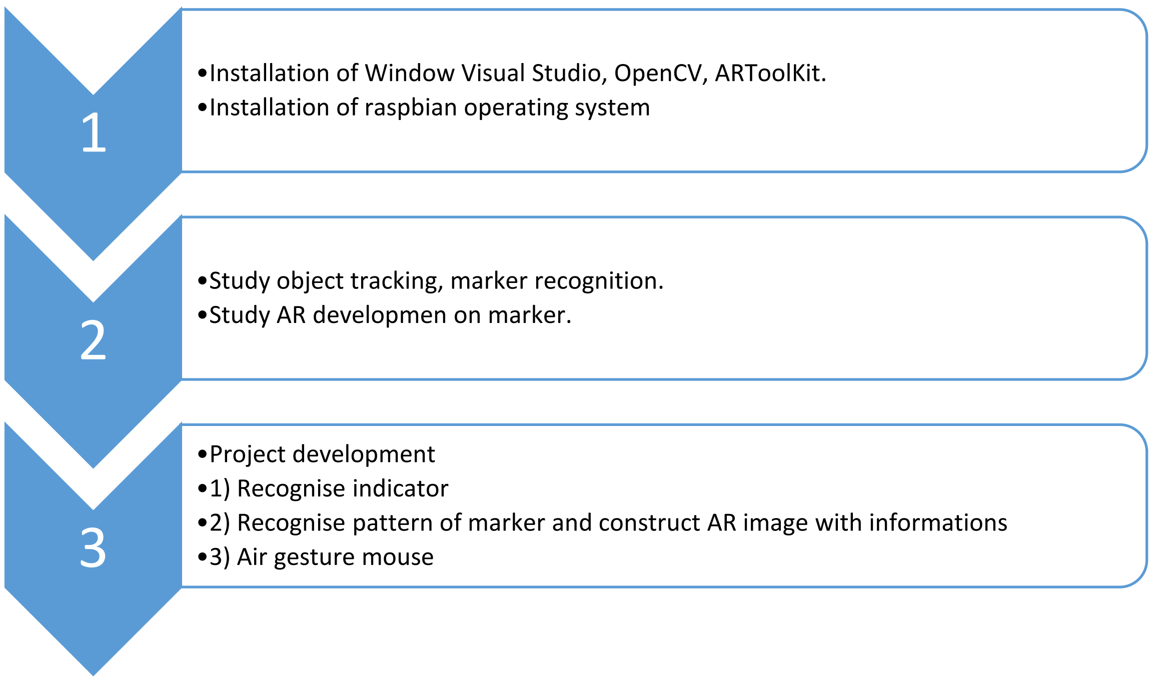

1
•Installation of Window Visual Studio, OpenCV, ARToolKit.
•Installation of raspbian operating system
2
•Study object tracking, marker recognition.
•Study AR developmen on marker.
3
•Project development
•1) Recognise indicator
•2) Recognise pattern of marker and construct AR image with informations
•3) Air gesture mouse

FIGURE 5 Methodology - Project Flowchart

3.2 Experimental Setup

- To test object detection for indicator finder, OpenCV is used. Image captured by camera is then processed. RGB is filtered. The image is adjusted the HSV to filter out the target colour. The filtered image then undergo morphological operation. The image either dilate or erode. After that, find the contor.
- To test AR, ARToolKit SDK is used along with OpenCV. From detected marker, the QR code pattern is read and compared with database pattern. Then, AR image is displayed on the marker corresponding to the marker.
- ZBAR SDK is used for QR code reader. From Scanned QR code, it then decoded into string of characters.
- Visual Studio 2013 is used instead latest version due to capability issues.

3.3 Activities Planned

TABLE 2 Activity Planned

NO	Activities	WEEK
1	Project Topic Selection, Cross Referencing	1
2	Identifying and obtaining the suitable components and software	2-3
3	Perform Image filter for Marker Detectopm	4-9
4	QR code scanner	10-16
5	Coding for AR Indoor Localization and Guidance	16-21
6	Develop On Air Selection Menu using image processing.	21-22
7	Test run on Raspberry Pi.	22-23
8	Full test run + finalized documentation.	24

3.4 Gantt Chart

#	Activity	Week														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Progress report								🟩							
2	Pre-SEDEX poster presentation										🟩					
3	Draft report													🟩		
4	Final Report														🟩	
5	Technical Paper														🟩	
6	FYP2 VIVA															🟩

FIGURE 6 Grantt Chart

3.5 Project Key Milestones

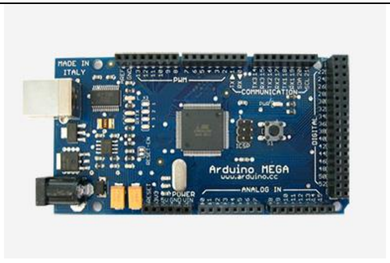
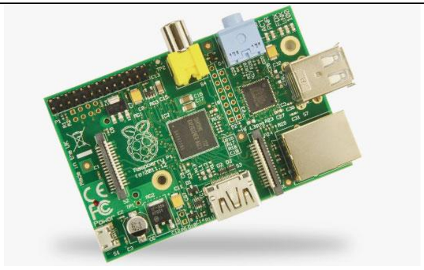
No.	Activity	Complete in
1	Progress report submit	Week 8
2	Pre-SEDEX poster presentation	Week 10
3	Draft report submit	Week 13
4	Final Report submit	Week 14
5	Technical Paper submit	Week 14
6	FYP2 VIVA	Week 15

FIGURE 7 Project Key Milestones

3.6 Identifying and obtaining the suitable components and software

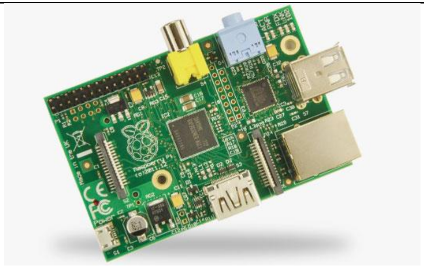
3.6.1 Component

TABLE 3 Comparison between Arduino and Raspberry Pi microcontroller

	Features	
54 Digital pin	Input/output (i/o) Pin	26 General Purpose Input Output(GPIO)
x	HDMI port	✓
x	RCA video output	✓
x	USB port	✓

Based on comparison TABLE 3, Arduino microcontroller lacks of features compared to Raspberry Pi for this project. Raspberry Pi is more flexible to use since this project will require to use USB webcam and require video output ports for video output.

TABLE 4 Comparison of Raspberry Pi Model

		
Raspberry Pi 1 Model A+	Model	Raspberry Pi 1 Model B
700MHz Broadcom BCM2835	Processor	700MHz Broadcom BCM2835
512MB	RAM	512MB
1	USB 2.0 port	2
40	GPIO	26
✓	HDMI	✓
Non-split	RCA video output	Split audio and video

Based on the comparison TABLE 4, both model are usable for this project. Model B has less input/output pin compared to model A. For this project, Model B is easier to use because of the split of RCA for audio and video which can separately connected to headphone and screen.

3.6.2 Software

This project uses Visual Studio 2013 and OpenCv version 2.4.13 library. In Microsoft environment, Visual Studio (VS) is the software to be used for image processing. In VS, the programming environments, building and debugging is handled well. In addition, ARToolkit and ZBAR SDK is used for AR rendering and QR coder reader respectively.

4 CHAPTER 4: OVERALL SYSTEM

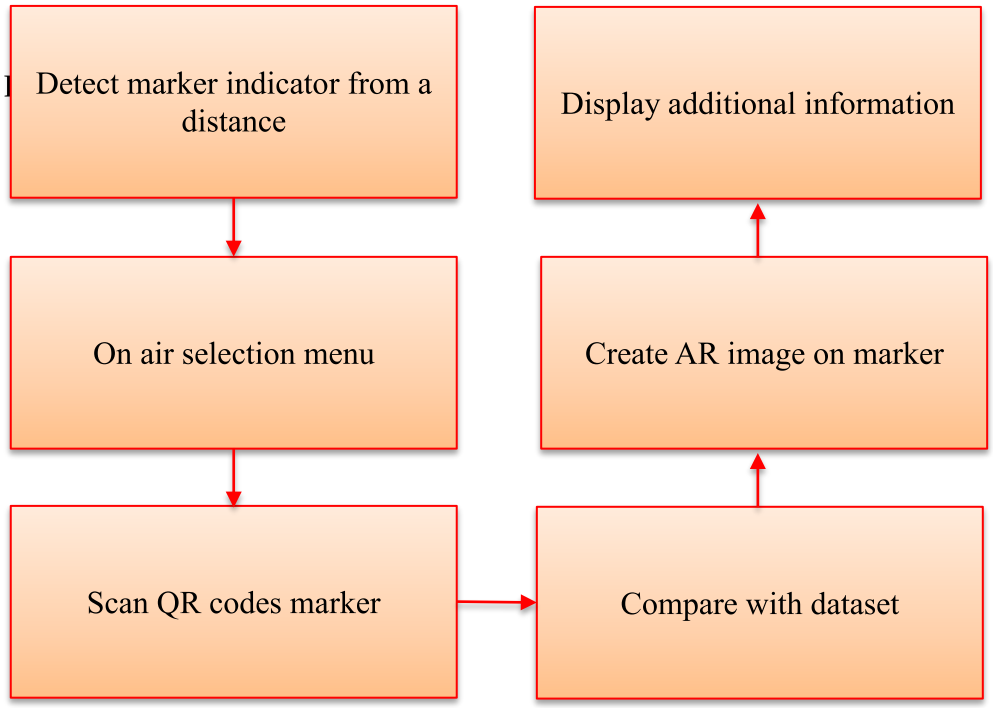

Detect marker indicator from a distance
On air selection menu
Scan QR codes marker
Compare with dataset
Create AR image on marker
Display additional information

FIGURE 8 Overall System Flowchart

IGURE 8 shows the overall flow of system. Firstly user need to locate marker, which can be done more easily with the help of marker indicator. Then, using on air selection menu, user choose his destination. After that, user need to scan the QR coded markers. The system will filter the captured image and read the marker definitions and compared it with database. Then, an image showing direction is rendered on the marker. In addition, some information can be prompted on the screen such as history of the location.

4.1 Overall Completed System

This section discuss about this project final progress. The project covers following objectives:

1. To develop AR in localized indoor guidance system
2. To develop long range marker indicator finder.
3. To develop on air menu selection.
4. To develop embedded information in markers using QR code.
5. To develop interactive localized indoor guidance system using raspberry pi

The project completed all above 4 objectives, however, for objective number 5, it is not complete. The localized indoor guidance system developed is not yet user friendly and it is not integrate with raspberry pi. The current progress of the interactive localized indoor guidance system only developed in Visual Studio using actual markers and colour indicators. Some adjustment needed in the coding so the system more user friendly. In Chapter 5 we will discuss further regarding result of the project developed.

4.2 Issues faced

In the period of project development, some issues were found and solved. Following are the issues that are encountered:

Issues	Action
Marker indicator detects multiple colours	Reduce the range of HSV value so filter other colour completely
Markers, and colour indicators should not be reflective.	Printed markers and colour indicators should not laminated to avoid reflective properties.
Error in program window's name	Add" _ITERATOR_DEBUG_LEVEL=0" in Visual Studio Pre-processor under C/C++ property to solve problem in creating task bar window and solve the missing window name.
Too sensitive on air menu selection	Uses 2 colour indicator (red and blue) as menu selector to avoid reading surrounding colour.
Un-accurate marker reading	Recalibrate the camera and generate new camera_para.dat file new camera calibration values.

5 CHAPTER 5: RESULT AND DISCUSSION.

5.1 Perform Image filter for Marker Indicator

A marker indicator is introduced because the camera is not high resolution thus the captured image was pixilate and hard to detect marker from long distance. Having a powerful camera can remove the dependency of this subsystem.

The marker indicator will indicate a marker nearby to it. This will ease user to find the marker.

This subsystem, object detection of colour and shape is used to detect indicators which indicate the position of the markers. The indicator is made of specific shape and colour. This is to distinguish its property from surrounding and make identification easier.

Based on study [10], the least popular colour is brown. Therefore for this project we use brownish colour with HSV value in TABLE 5 to distinguish itself from surrounding. In addition to make it more distinguish, the shape of the indicator also determined which is circle. The indicator for this prototype is circle and HSV value of TABLE 4. In future for improvement, more unique detail such as colour in pattern, number of edges can be used so the indicator more accurate without affected by certain condition.

5.1.1 Setup Process:

The setup process as follows:

1. Setup the window environment,
2. Create a new project in VS
3. Include the file dependencies for this project.
4. Add this line " _ITERATOR_DEBUG_LEVEL=0" in Pre-processor under C/C++ property to solve problem in creating task bar window and solve the missing window name.

5. Create the program, build and test run. Please refer the code in the appendix section.

5.1.2 Flow-chart of Image filter for Marker Indicator

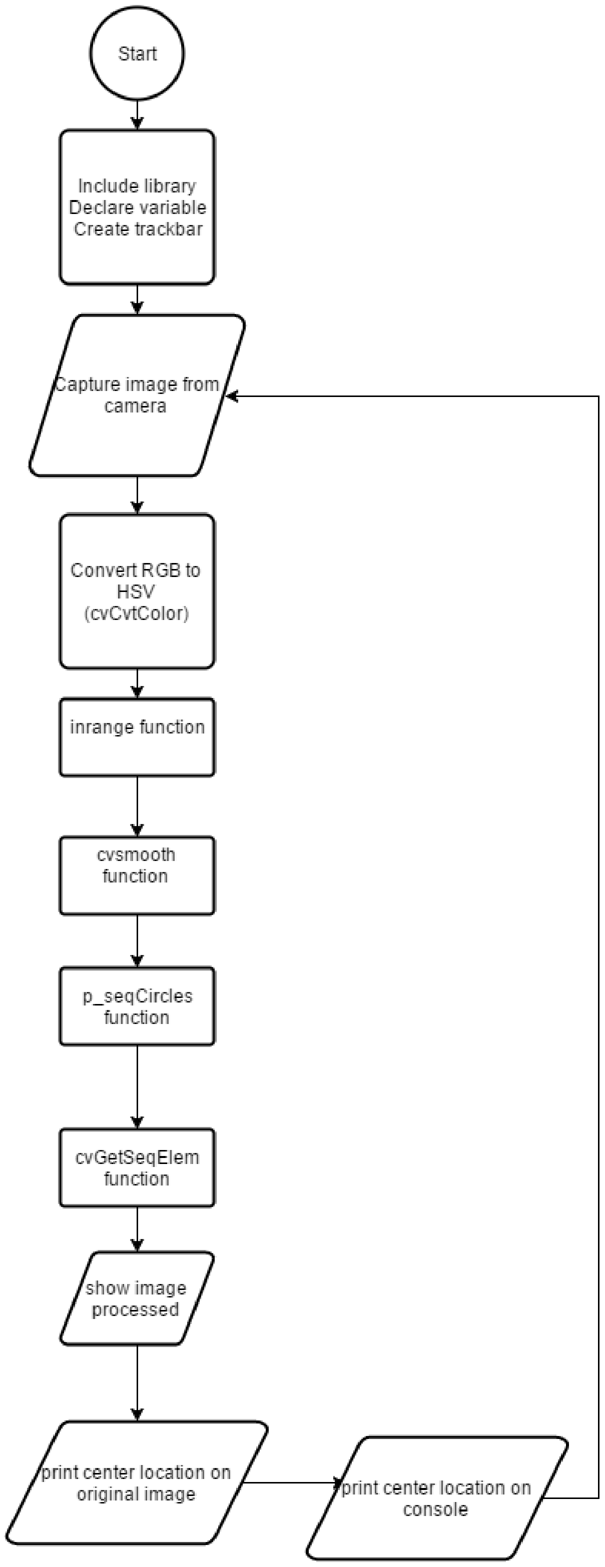


FIGURE 9 Flowchart of Marker Indicator

Program flow:

1. Include library, declare variable and create trackbar.
2. Image captured from webcam
3. Image feed changed from RGB to HSV using cvCvtColor
4. Use inrange function to identify if array elements lie between the elements of two other arrays. This function filter all other HSV value and left with desired value.
5. Use cvsmooth function to make the processed image smooth and easier for next process.
6. Use p_seqCircles function to detect circle structure
7. cvGetSeqElem function read positions of the structure
8. Show the images and the position of the structure.

5.1.3 Result of Image filter for Marker Indicator

The image feed is converted from Red Green Blue (RGB) to Hue Saturation Value (HSV). Track bar is adjusted to desired value. The inrange function will use this values to filter out other value and left with the desired value.

From the value left, which here is brown in colour, the circle structure is identified from the filtered colour and a centre is drawn. The centre location and its radius is print. The effective range for this radius of 4.24cm object surface is 4m far using 5MP web cam under of range of 200-2016 lux of light luminance.

TABLE 5 HSV of marker indicator

Variable	Value	
	Min	Max
Hue	11	19
Saturation	139	181
Value	128	153

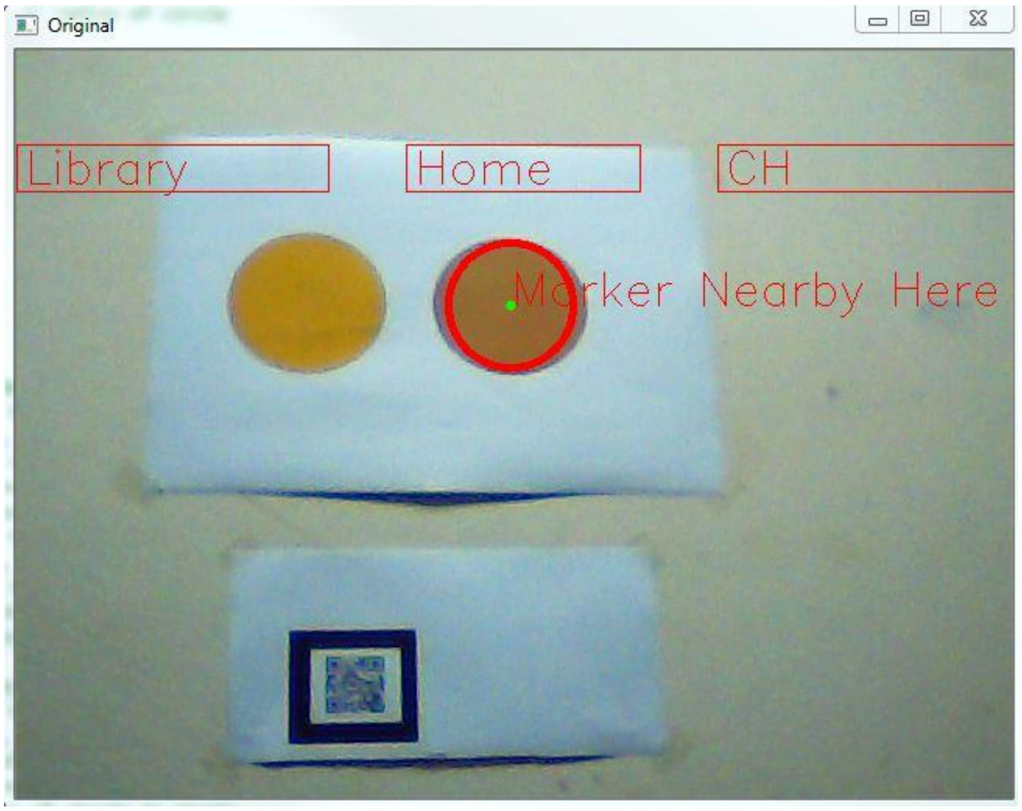


FIGURE 10 Marker Indicator

FIGURE 10 shows the marker indicator that informs user the marker is nearby. By having tight HSV range of values, the marker is distinct from surrounding.

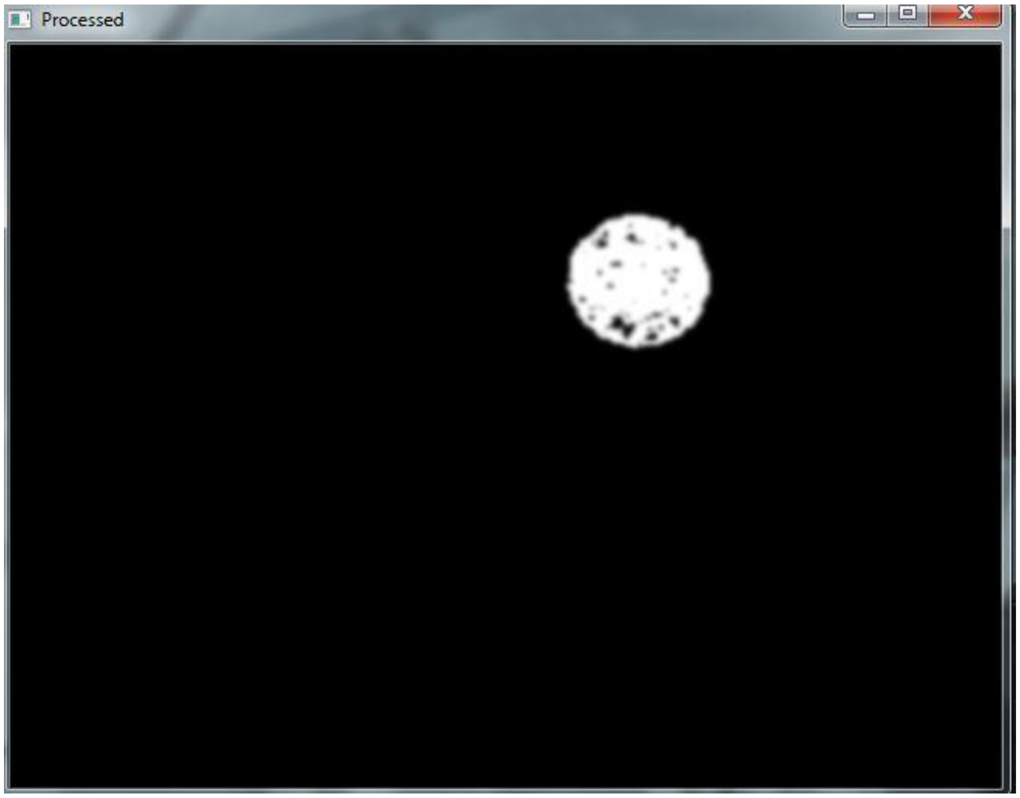
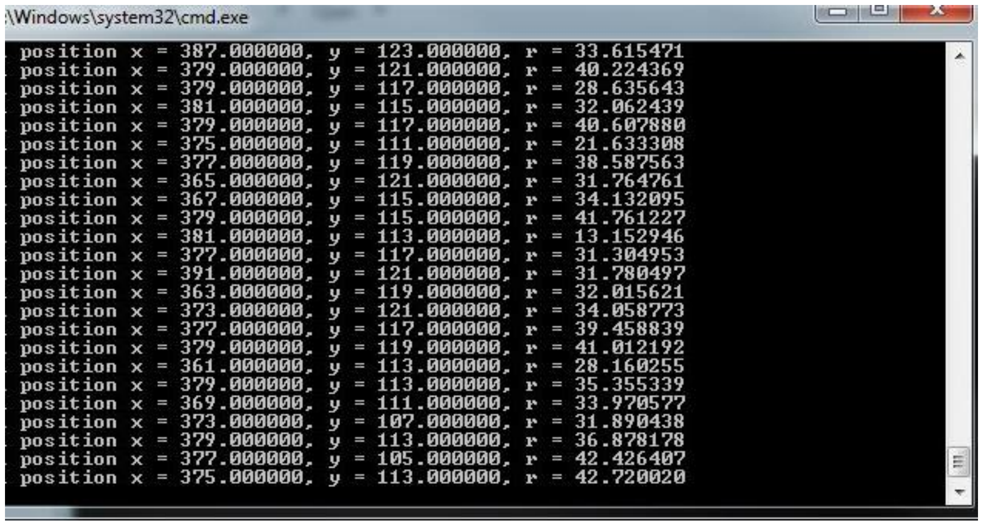


FIGURE 11 Processed image of marker indicator



```
position x = 387.000000, y = 123.000000, r = 33.615471
position x = 379.000000, y = 121.000000, r = 40.224369
position x = 379.000000, y = 117.000000, r = 28.635643
position x = 381.000000, y = 115.000000, r = 32.062439
position x = 379.000000, y = 117.000000, r = 40.607880
position x = 375.000000, y = 111.000000, r = 21.633308
position x = 377.000000, y = 119.000000, r = 38.587563
position x = 365.000000, y = 121.000000, r = 31.764761
position x = 367.000000, y = 115.000000, r = 34.132095
position x = 379.000000, y = 115.000000, r = 41.761227
position x = 381.000000, y = 113.000000, r = 13.152946
position x = 377.000000, y = 117.000000, r = 31.304953
position x = 391.000000, y = 121.000000, r = 31.780497
position x = 363.000000, y = 119.000000, r = 32.015621
position x = 373.000000, y = 121.000000, r = 34.058773
position x = 377.000000, y = 117.000000, r = 39.458839
position x = 379.000000, y = 119.000000, r = 41.012192
position x = 361.000000, y = 113.000000, r = 28.160255
position x = 379.000000, y = 113.000000, r = 35.355339
position x = 369.000000, y = 111.000000, r = 33.970577
position x = 373.000000, y = 107.000000, r = 31.890438
position x = 379.000000, y = 113.000000, r = 36.878178
position x = 377.000000, y = 105.000000, r = 42.426407
position x = 375.000000, y = 113.000000, r = 42.720020
```

FIGURE 12 Position of the marker indicator

FIGURE 12 shows the position of the marker indicator in x and y axis and the radius. Value x and y determines the center coordinate of the circle. This value is used to draw the red circle around it corresponds to the value of radius. A string “Marker Nearby Here” will be prompted centering to the x and y.

5.2 On air selection menu

Instead using mouse and keyboard to do menu selection, this project implements on air selection menu. Using same method in Marker Indicator, the coordinate of the centre of the red and blue circle as in FIGURE 13 is used as condition for the menu selection.

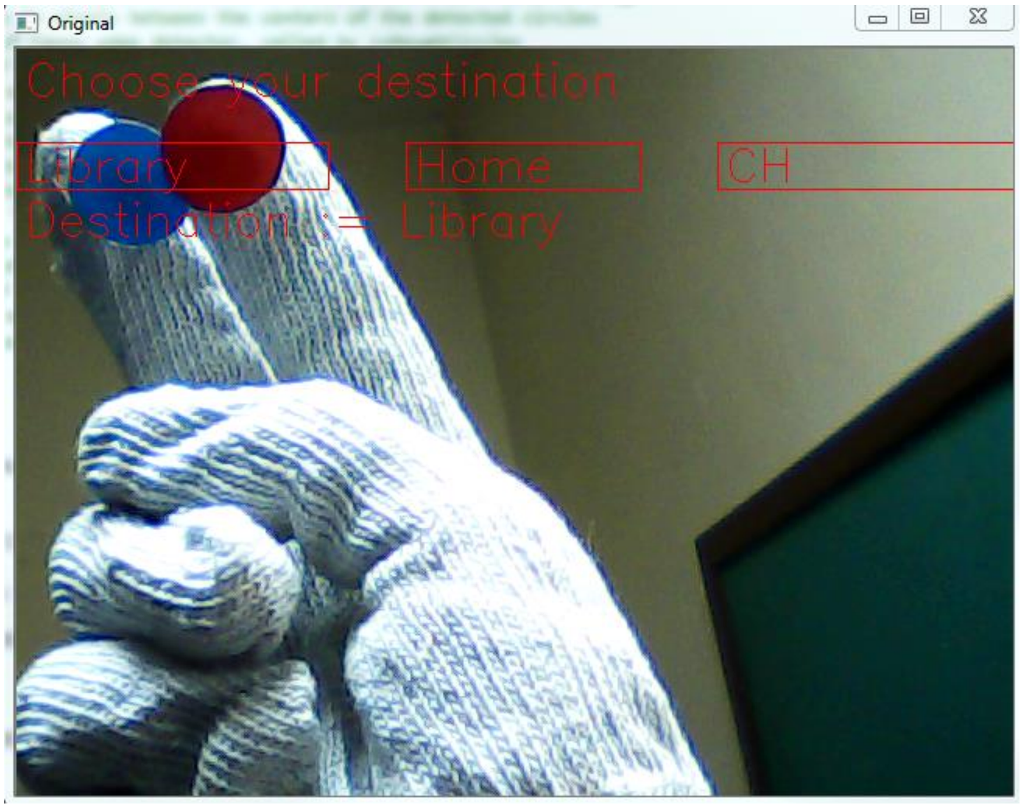


FIGURE 13 On air menu selection

5.2.1 Result of on air selection menu

TABLE 6 Range x,y for menu selection

Destination	x Range min	y Range min	x Range max	y Range max
Library	0	60	200	90
Home	250	60	400	90
CH	450	60	800	90

For example, destination "Library" will be set if the coordinate for both red and blue circle fall between 0<x<60 and 200<y<90.

5.3 QR code scanner.

For this project, QR codes as markers for AR display are used. This project require ZBAR SDK to identify QR codes and decode into strings of characters. After that we will compare the string of characters with our database and print information. For this project, we will use 3 QR code marker as prototype. The QR codes are generated using online QR code generator from http://www.qr-code-generator.com/.

5.3.1 Setup Process

1. Install ZBAR from : http://sourceforge.net/projects/zbar/files/zbar/0.10/zbar-0.10-setup.exe/download
2. In visual studio project, open system properties. Import headers (.h files) and libraries (.lib) at "Additional Dependencies" under VC++ directories tab.
3. Copy libzbar-0.dll into the project directory folder.
4. Run test program.
5. Modify coding to compare string read from QR code and database to print information from database.

5.3.2 Flow-chart of QR code scanner.

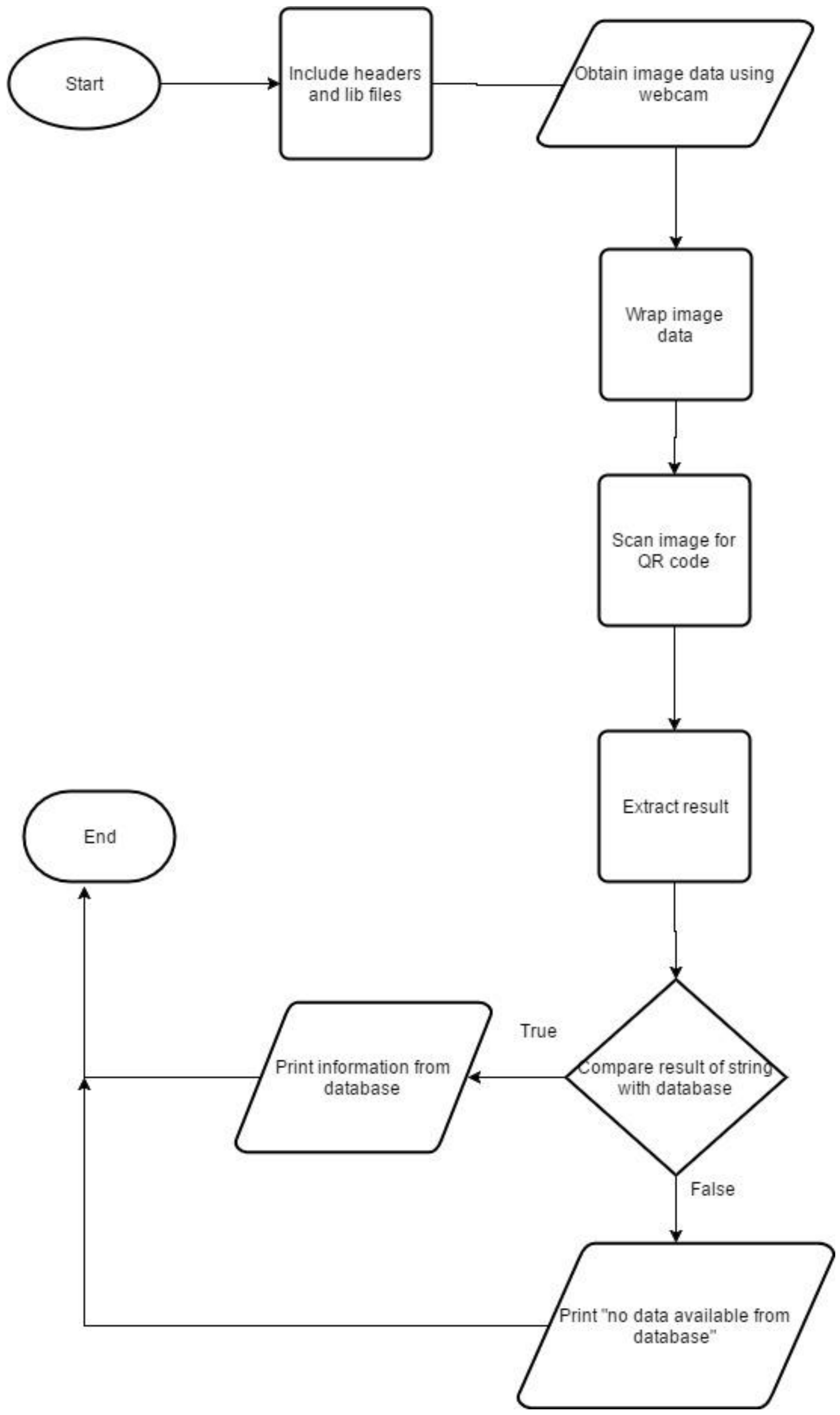

Start
Include headers and lib files
Obtain image data using webcam
Wrap image data
Scan image for QR code
Extract result
Compare result of string with database
True
Print information from database
False
Print "no data available from database"
End

FIGURE 14 Flow chart of QR code scanner

5.3.3 Result of QR code scanner.

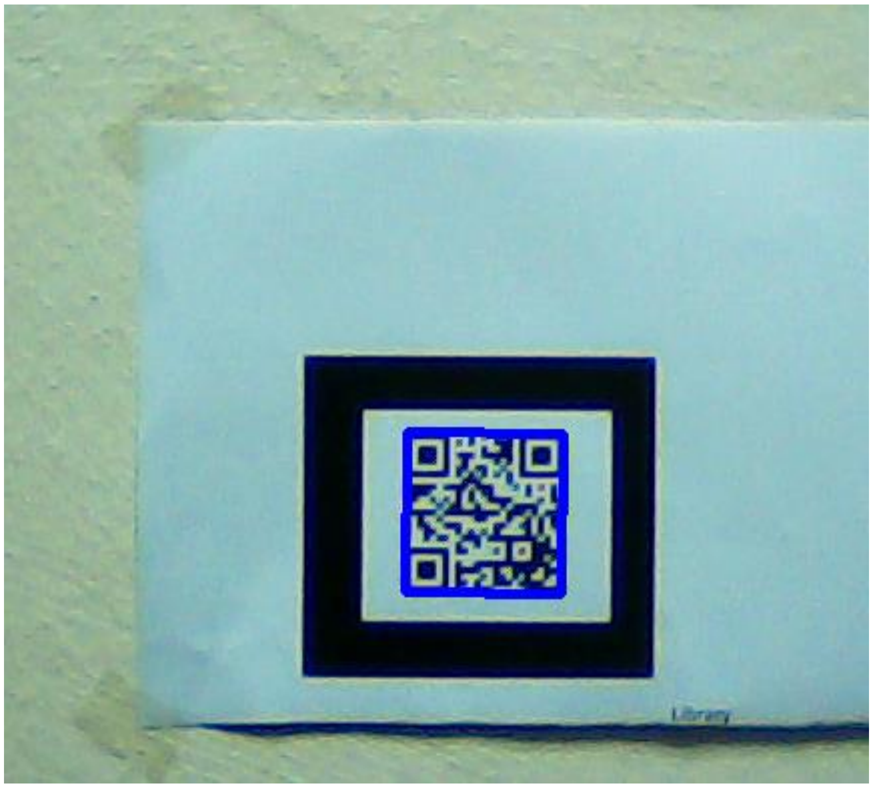
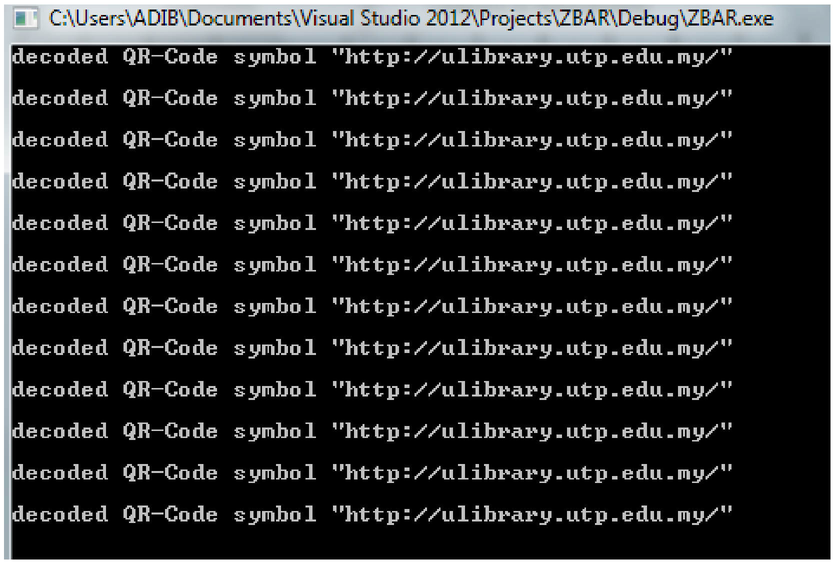


FIGURE 15 QR code scanning

FIGURE 15 show the captured frame. In this frame, the QR code is detected. The area inside the blue square is processed using Zbar library to decode the QR code.



```
C:\Users\ADIB\Documents\Visual Studio 2012\Projects\ZBAR\Debug\ZBAR.exe
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
decoded QR-Code symbol "http://ulibrary.utp.edu.my/"
```

FIGURE 16 Console output for QR scanner program

FIGURE 16 shows the result of decoded QR code. In this, a string is printed as the output. The string will be compared with available data. If matched, a sequence of information will be printed such as the name of marker, current location, points of interest at that location and etc.

QR code markers generated as follows:

TABLE 7 QR codes database

Link	QR Code	Marker Name
https://www.utp.edu.my/SitePages/Home.aspx		qr_code_utp_home
http://ulibrary.utp.edu.my/		qr_code_library
https://www.utp.edu.my/Students/SitePages/Home.aspx		qr_code_student(ch)

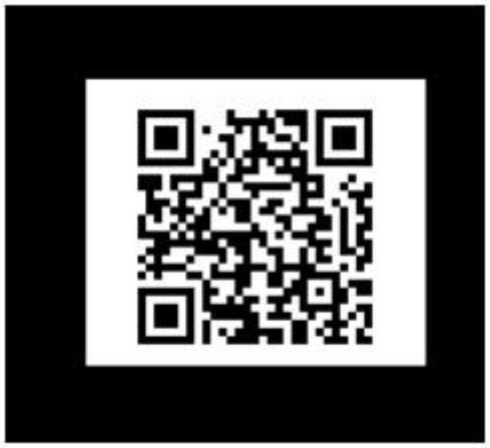


FIGURE 17 QR marker “qr_code utp home”

FIGURE 17 is the photoshoped image of earlier QR code for “qr_code utp home”. The condition for the marker must be bordered by 2-3 cm black border for easier recognition of the marker. The size of the marker is 6cmx6cm. The marker then processed into .pat file using ARToolkit marker generator [11].

5.4 Coding for AR Indoor Localization and Guidance

Using ARToolkit, AR library is used to generate images over QR code markers. This section discuss the steps for AR image rendering on markers. The AR image rendered on markers show the direction for user to follow to reach destination.

5.4.1 Setup Process

1) Download from https://artoolkit.org/ and Install ARToolkit.
2) Setup environment variables
3) Include AR .lib files and .bin files to property sheet.
4) Develop code
5) Build and test run.

5.4.2 Flow-chart of AR Indoor Localization and Guidance.

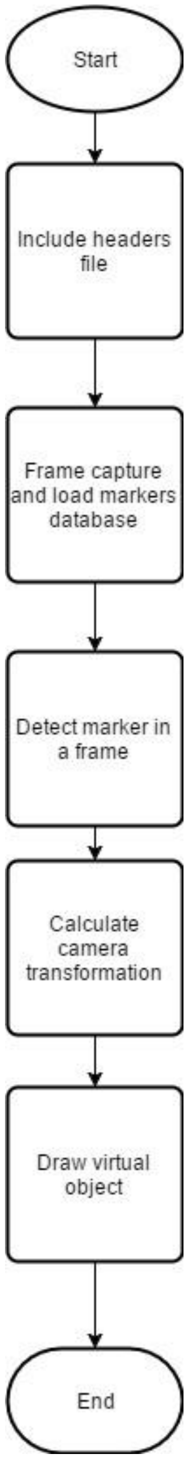


FIGURE 18 Flow chart of AR Indoor Localization and Guidance

5.4.3 Result of AR Indoor Localization and Guidance.

For this project, 3 markers were each used for 3 locations respectively. The locations are named Library, Home and CH. TABLE 8 and FIGURE 19 show arrangements.

TABLE 8 Location and marker name

Location	Marker placed
Library	qr_code_library
Home	qr_code utp_home
CH	qr_code_student(ch)

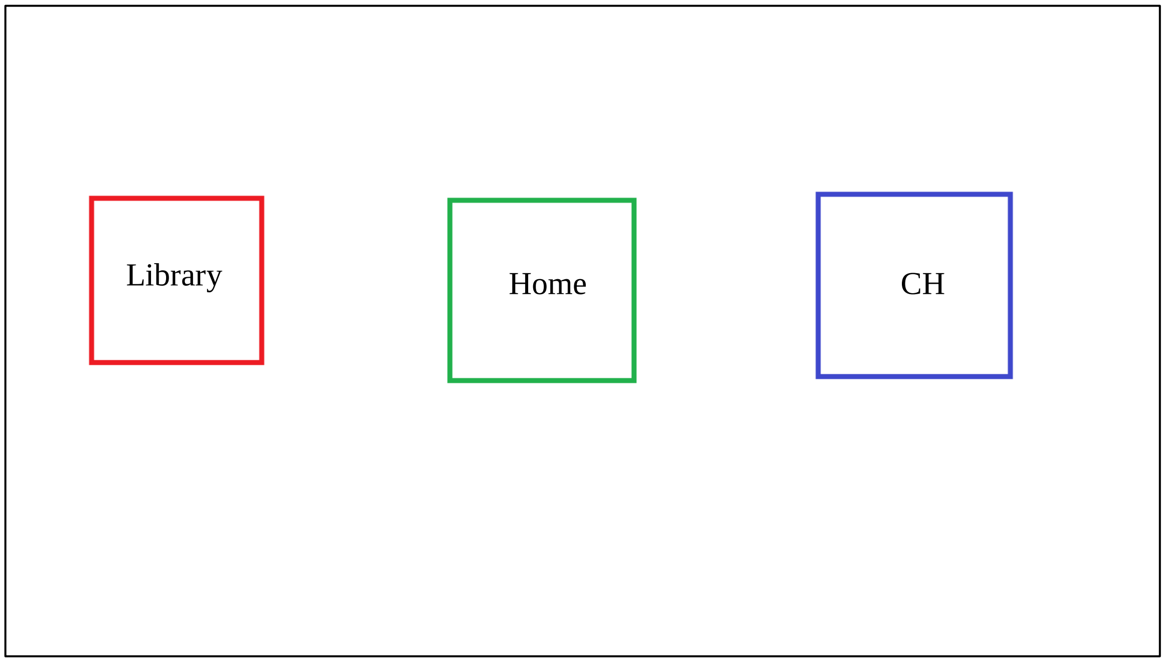


FIGURE 19 Location of markers

TABLE 9 Route

	Current Position		
Destination	Library	Home	CH
Library	Arrive	Left	Left
Home	Right	Arrive	Left
CH	Right	Right	Arrive

TABLE 9 shows that direction is shown differently for each selection of destination. For example, if user wants to go to CH, he will prompted by image show that he need to go to "Right" at Library and Home position. FIGURE 16 is the image prompted for current situation at location Library.

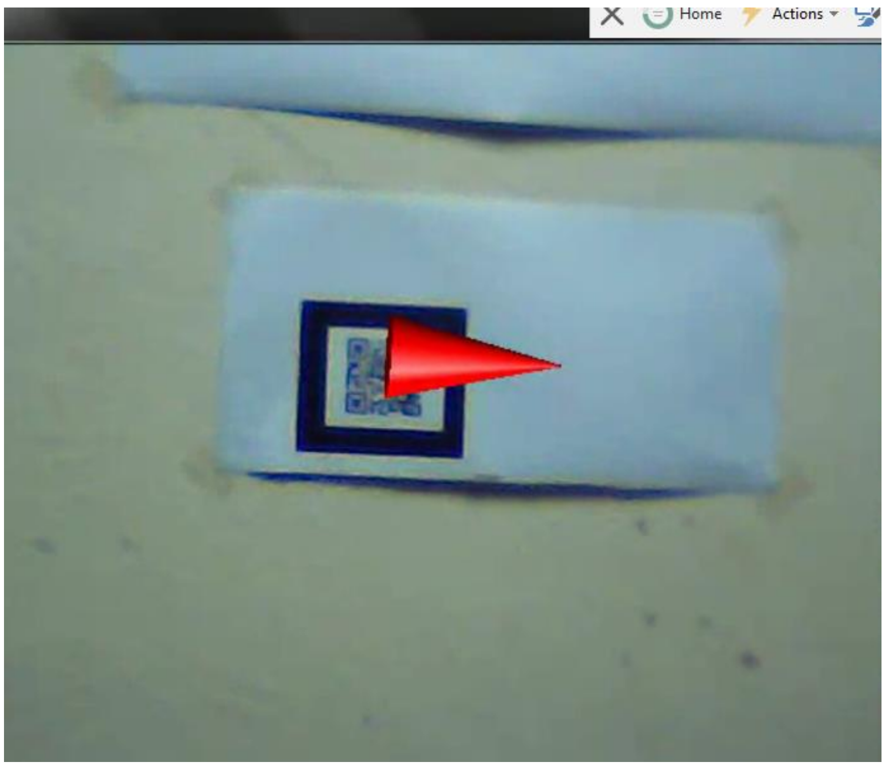


FIGURE 20 Marker "qr_code_library" showing direction to the right

6 CONCLUSION AND RECOMMENDATION

To sum up, Indoor Localization and Guidance using Augmented Reality Toolbox will create a new concept of indoor navigation to replace the existing conventional method. A marker indicator is used to ease user to find marker's location. Air selection menu removes the need for mouse and keyboard and QR coded markers now hold more information rather than conventional markers. In future, this technology can improved by adding more functional subsystem such as implementing communication service inside the system and audio navigation.

7 REFERENCES

1] F. Malek, "AUGMENTED REALITY BASED INDOOR POSITIONING NAVIGATION TOOL," Tronoh, 2016.

2] M. Rouse, "augmented reality (AR)," February 2016. [Online]. Available: http://whatis.techtarget.com/definition/augmented-reality-AR. [Accessed 14 OCTOBER 2016].

3] Corné, "Marker Detection for AR Applications," infi, 2 April 2010. [Online]. Available: https://infi.nl/nieuws/marker-detection-for-augmented-reality-applications/. [Accessed 20 OCTOBER 2016].

4] M. Hirzer, "Marker Detection for Augmented," computer graphics & vision, Austria, 2008.

5] R. Hussin, M. R. Juhari, N. W. Kang, R. C. Ismail and A. Kamarudin, "Digital Image Processing Techniques for Object Detection From Complex Background Image," *Procedia Engineering,* no. 41, pp. 340-344, 2012.

6] A. Rosebrock, "OpenCV Track Object Movement," pyimagesearch, 21 September 2015. [Online]. Available: http://www.pyimagesearch.com/2015/09/21/opencv-track-object-movement/. [Accessed 20 OCTOBER 2016].

7] "Information Technology — Automatic identification and data capture techniques — Bar code symbology — QR Code, ISO/IEC 18004," in *Int'l Organization for Standardization,* 2000.

8] E. Kang, "A rectification method for quick response code image," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014),* 2014.

A. Fuller, "Decoding small QR codes by hand," Solder and Flux, [Online]. Available: http://blog.qartis.com/decoding-small-qr-codes-by-hand/. [Accessed 1

[9] MARCH 2017].

[10] S. Work, "True Colors – Breakdown of Color Preferences by Gender," KISSmetrics, [Online]. Available: https://blog.kissmetrics.com/gender-and-color/. [Accessed 22 FEBUARY 2017].

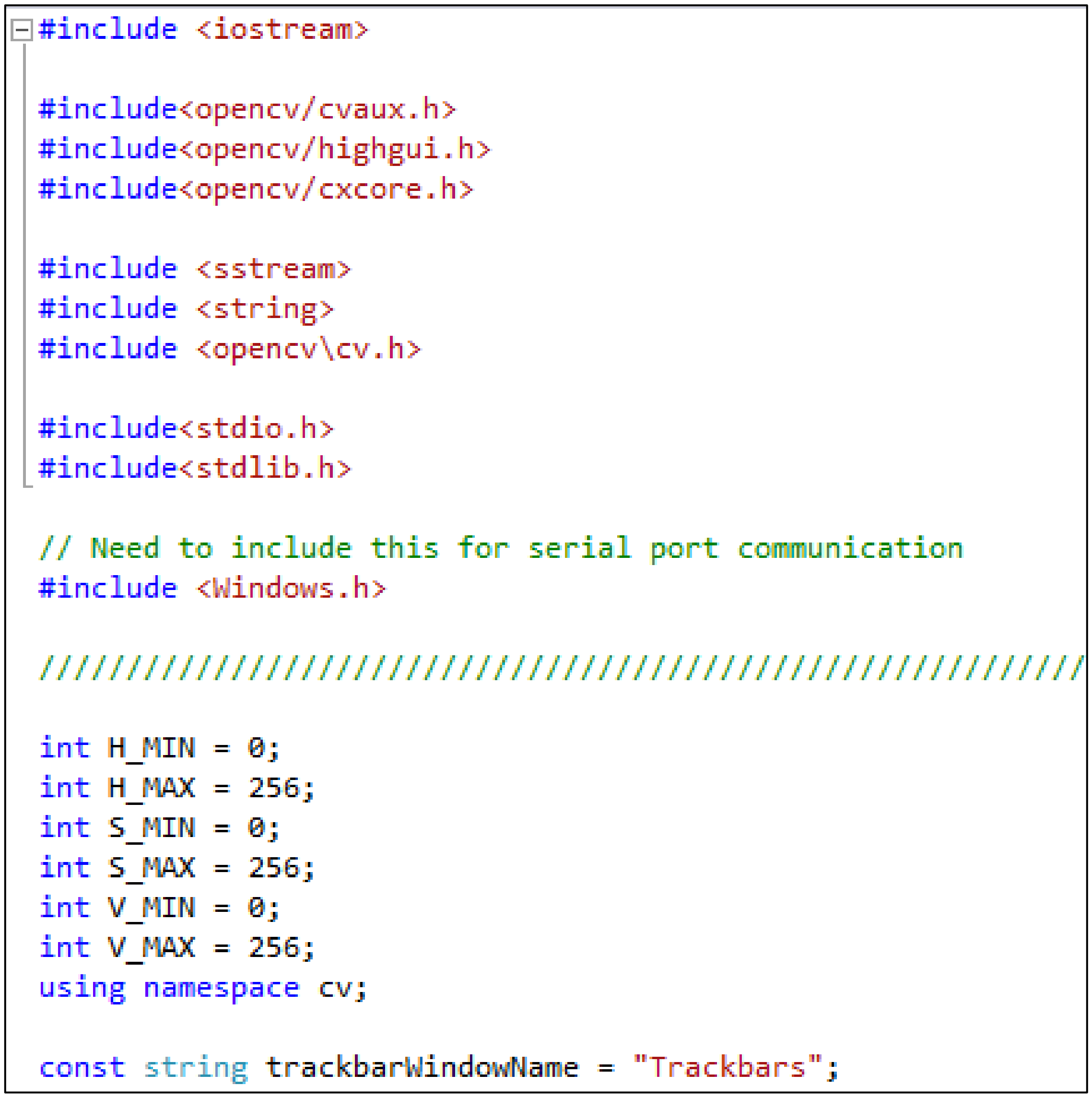
[11] "Marker"s" Generator Online Released!," [Online]. Available: http://flash.tarotaro.org/blog/2009/07/12/mgo2/. [Accessed 25 DECEMBER 2016].

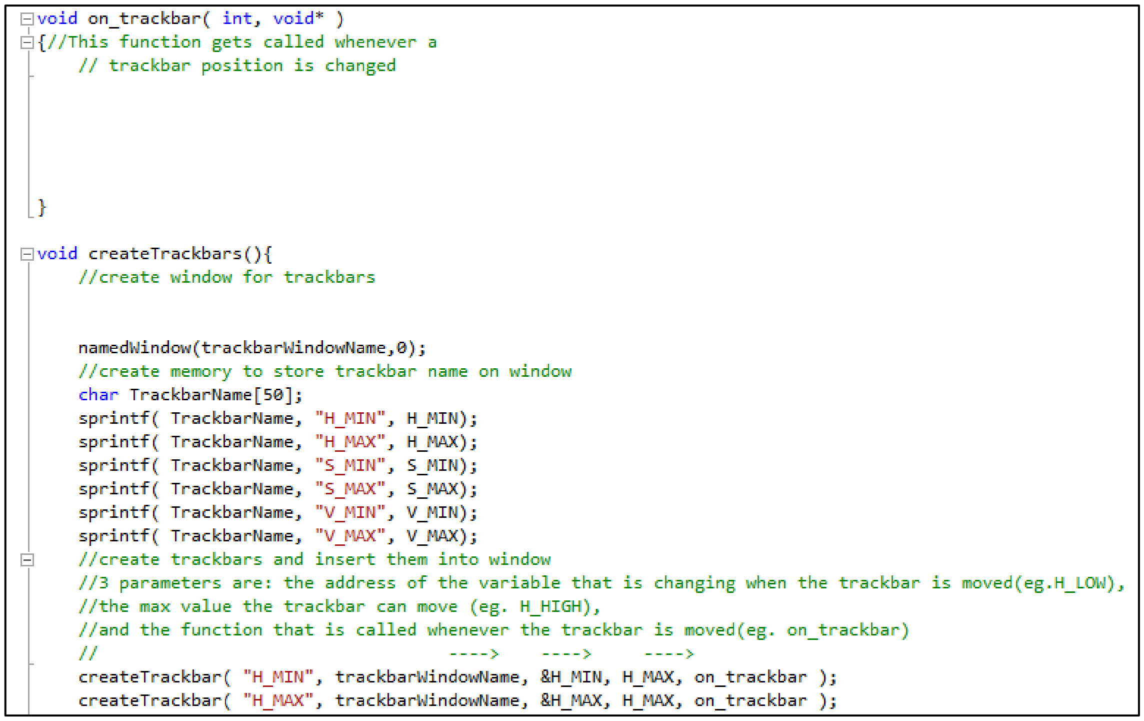
8 APPENDIX

QR Code

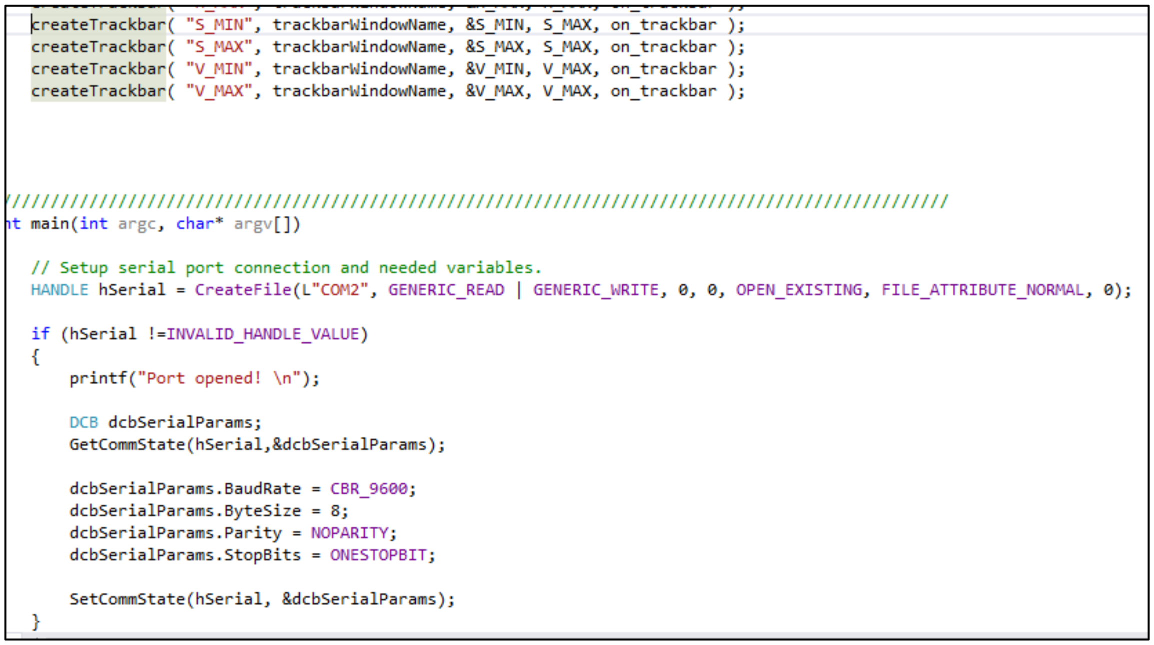
Example of AR

Object Detection Code:

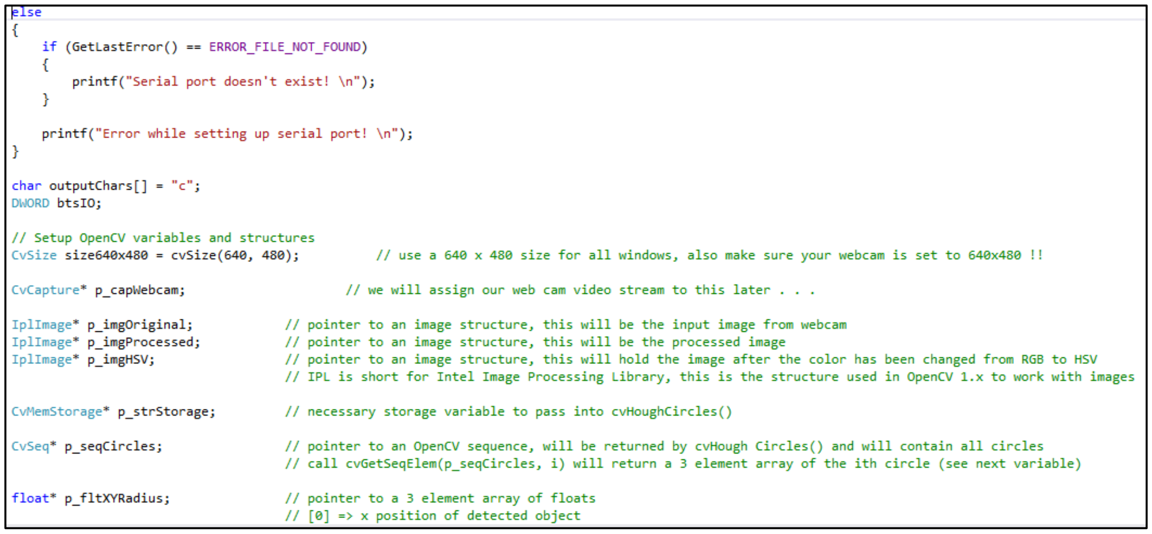

```
#include <iostream>

#include<opencv/cvaux.h>
#include<opencv/highgui.h>
#include<opencv/cxcore.h>

#include <sstream>
#include <string>
#include <opencv\cv.h>

#include<stdio.h>
#include<stdlib.h>

// Need to include this for serial port communication
#include <Windows.h>

///////////////////////////////////////////////////////////////

int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
using namespace cv;

const string trackbarWindowName = "Trackbars";
```



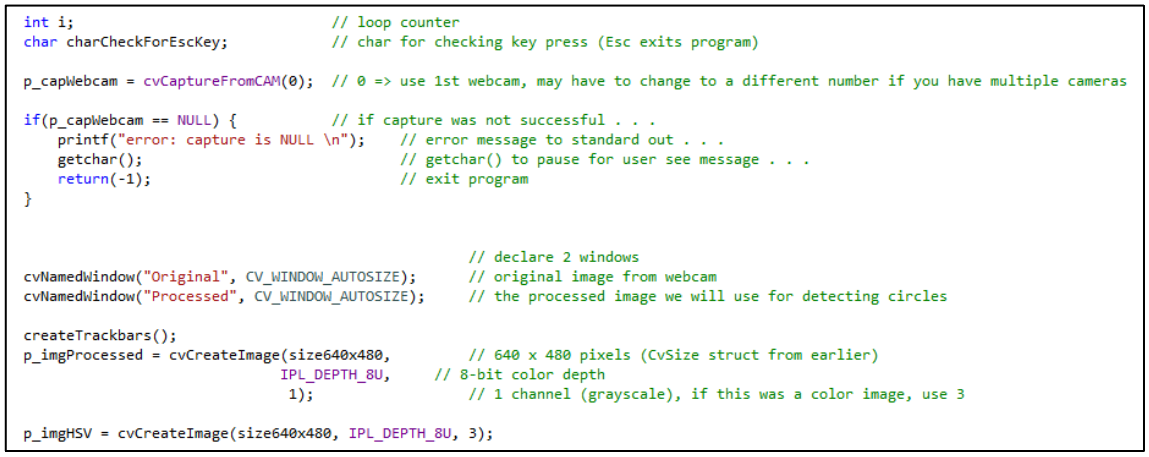
```
void on_trackbar( int, void* )
{//This function gets called whenever a
    // trackbar position is changed






}

void createTrackbars(){
    //create window for trackbars


    namedWindow(trackbarWindowName,0);
    //create memory to store trackbar name on window
    char TrackbarName[50];
    sprintf( TrackbarName, "H_MIN", H_MIN);
    sprintf( TrackbarName, "H_MAX", H_MAX);
    sprintf( TrackbarName, "S_MIN", S_MIN);
    sprintf( TrackbarName, "S_MAX", S_MAX);
    sprintf( TrackbarName, "V_MIN", V_MIN);
    sprintf( TrackbarName, "V_MAX", V_MAX);
    //create trackbars and insert them into window
    //3 parameters are: the address of the variable that is changing when the trackbar is moved(eg.H_LOW),
    //the max value the trackbar can move (eg. H_HIGH),
    //and the function that is called whenever the trackbar is moved(eg. on_trackbar)
    //                                  ---->    ---->     ---->
    createTrackbar( "H_MIN", trackbarWindowName, &H_MIN, H_MAX, on_trackbar );
    createTrackbar( "H_MAX", trackbarWindowName, &H_MAX, H_MAX, on_trackbar );
```



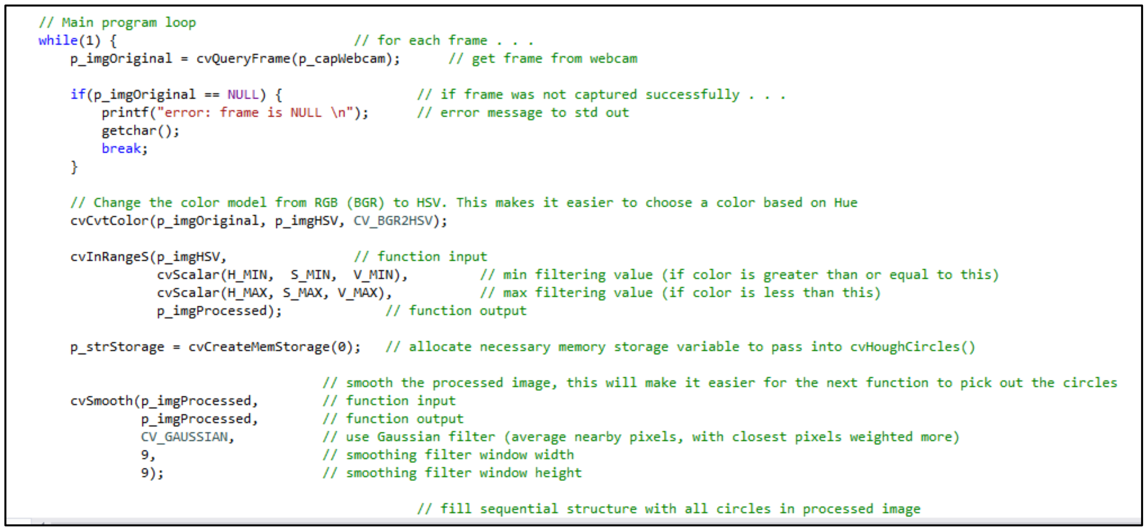
```
    createTrackbar( "S_MIN", trackbarWindowName, &S_MIN, S_MAX, on_trackbar );
    createTrackbar( "S_MAX", trackbarWindowName, &S_MAX, S_MAX, on_trackbar );
    createTrackbar( "V_MIN", trackbarWindowName, &V_MIN, V_MAX, on_trackbar );
    createTrackbar( "V_MAX", trackbarWindowName, &V_MAX, V_MAX, on_trackbar );




/////////////////////////////////////////////////////////////////////////////////////////
t main(int argc, char* argv[])

    // Setup serial port connection and needed variables.
    HANDLE hSerial = CreateFile(L"COM2", GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

    if (hSerial !=INVALID_HANDLE_VALUE)
    {
        printf("Port opened! \n");

        DCB dcbSerialParams;
        GetCommState(hSerial,&dcbSerialParams);

        dcbSerialParams.BaudRate = CBR_9600;
        dcbSerialParams.ByteSize = 8;
        dcbSerialParams.Parity = NOPARITY;
        dcbSerialParams.StopBits = ONESTOPBIT;

        SetCommState(hSerial, &dcbSerialParams);
    }
```



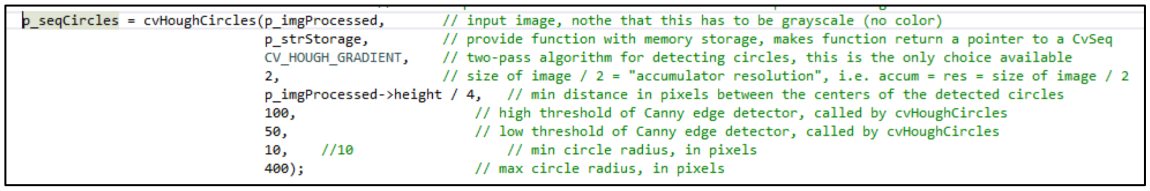
```
else
{
    if (GetLastError() == ERROR_FILE_NOT_FOUND)
    {
        printf("Serial port doesn't exist! \n");
    }

    printf("Error while setting up serial port! \n");
}

char outputChars[] = "c";
DWORD btsIO;

// Setup OpenCV variables and structures
CvSize size640x480 = cvSize(640, 480);          // use a 640 x 480 size for all windows, also make sure your webcam is set to 640x480 !!

CvCapture* p_capWebcam;                         // we will assign our web cam video stream to this later . . .

IplImage* p_imgOriginal;            // pointer to an image structure, this will be the input image from webcam
IplImage* p_imgProcessed;           // pointer to an image structure, this will be the processed image
IplImage* p_imgHSV;                 // pointer to an image structure, this will hold the image after the color has been changed from RGB to HSV
                                    // IPL is short for Intel Image Processing Library, this is the structure used in OpenCV 1.x to work with images

CvMemStorage* p_strStorage;         // necessary storage variable to pass into cvHoughCircles()

CvSeq* p_seqCircles;                // pointer to an OpenCV sequence, will be returned by cvHough Circles() and will contain all circles
                                    // call cvGetSeqElem(p_seqCircles, i) will return a 3 element array of the ith circle (see next variable)

float* p_fltXYRadius;               // pointer to a 3 element array of floats
                                    // [0] => x position of detected object
```



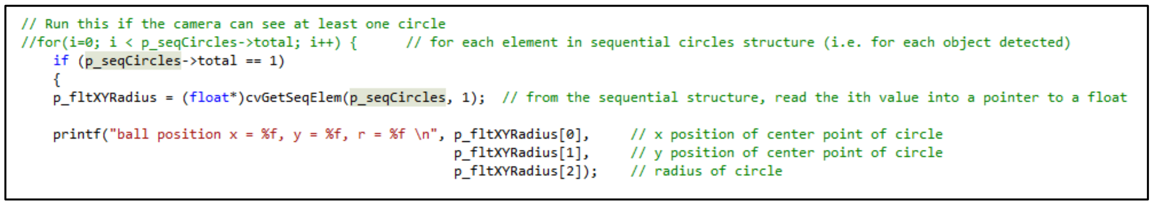
```
int i;                              // loop counter
char charCheckForEscKey;            // char for checking key press (Esc exits program)

p_capWebcam = cvCaptureFromCAM(0);  // 0 => use 1st webcam, may have to change to a different number if you have multiple cameras

if(p_capWebcam == NULL) {           // if capture was not successful . . .
    printf("error: capture is NULL \n");    // error message to standard out . . .
    getchar();                              // getchar() to pause for user see message . . .
    return(-1);                             // exit program
}


                                            // declare 2 windows
cvNamedWindow("Original", CV_WINDOW_AUTOSIZE);      // original image from webcam
cvNamedWindow("Processed", CV_WINDOW_AUTOSIZE);     // the processed image we will use for detecting circles

createTrackbars();
p_imgProcessed = cvCreateImage(size640x480,         // 640 x 480 pixels (CvSize struct from earlier)
                               IPL_DEPTH_8U,    // 8-bit color depth
                                1);                 // 1 channel (grayscale), if this was a color image, use 3

p_imgHSV = cvCreateImage(size640x480, IPL_DEPTH_8U, 3);
```



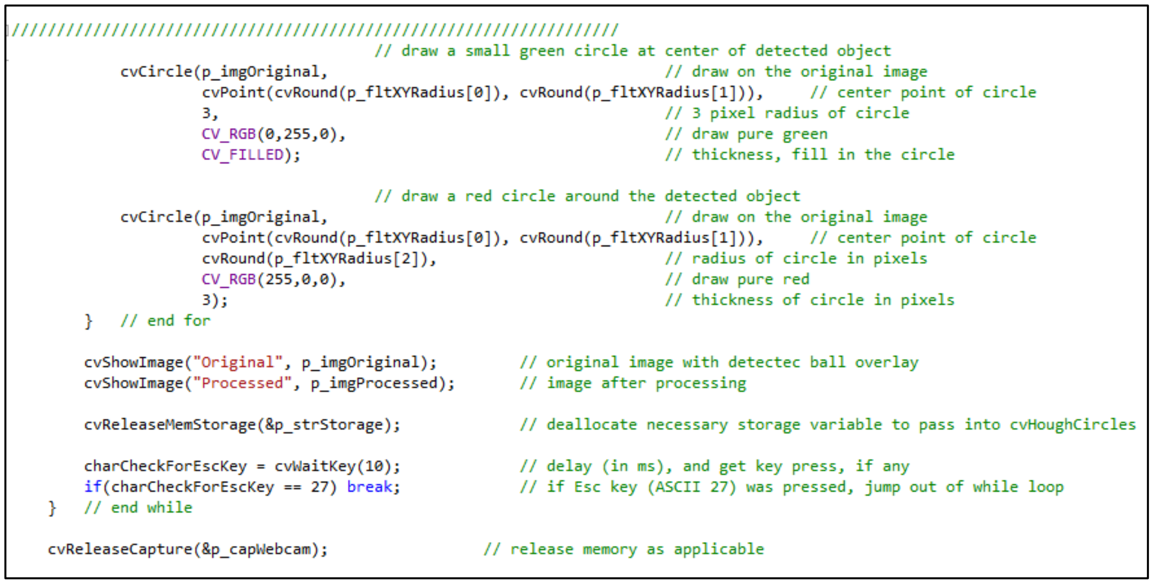
```
// Main program loop
while(1) {                              // for each frame . . .
    p_imgOriginal = cvQueryFrame(p_capWebcam);      // get frame from webcam

    if(p_imgOriginal == NULL) {                 // if frame was not captured successfully . . .
        printf("error: frame is NULL \n");      // error message to std out
        getchar();
        break;
    }

    // Change the color model from RGB (BGR) to HSV. This makes it easier to choose a color based on Hue
    cvCvtColor(p_imgOriginal, p_imgHSV, CV_BGR2HSV);

    cvInRangeS(p_imgHSV,                // function input
               cvScalar(H_MIN,  S_MIN,  V_MIN),         // min filtering value (if color is greater than or equal to this)
               cvScalar(H_MAX, S_MAX, V_MAX),           // max filtering value (if color is less than this)
               p_imgProcessed);             // function output

    p_strStorage = cvCreateMemStorage(0);   // allocate necessary memory storage variable to pass into cvHoughCircles()

                                    // smooth the processed image, this will make it easier for the next function to pick out the circles
    cvSmooth(p_imgProcessed,        // function input
             p_imgProcessed,        // function output
             CV_GAUSSIAN,           // use Gaussian filter (average nearby pixels, with closest pixels weighted more)
             9,                     // smoothing filter window width
             9);                    // smoothing filter window height

                                            // fill sequential structure with all circles in processed image
```



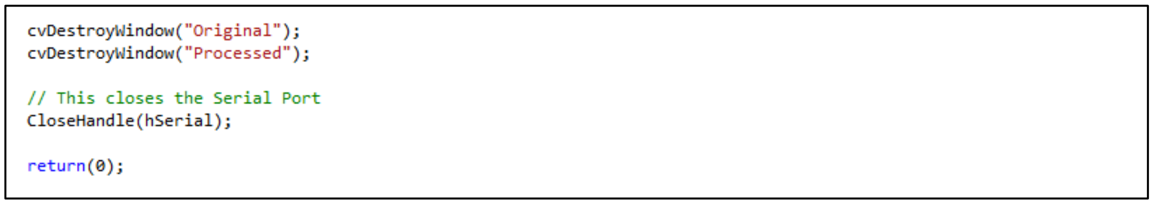
```
p_seqCircles = cvHoughCircles(p_imgProcessed,       // input image, nothe that this has to be grayscale (no color)
                              p_strStorage,         // provide function with memory storage, makes function return a pointer to a CvSeq
                              CV_HOUGH_GRADIENT,    // two-pass algorithm for detecting circles, this is the only choice available
                              2,                    // size of image / 2 = "accumulator resolution", i.e. accum = res = size of image / 2
                              p_imgProcessed->height / 4,    // min distance in pixels between the centers of the detected circles
                              100,                      // high threshold of Canny edge detector, called by cvHoughCircles
                              50,                       // low threshold of Canny edge detector, called by cvHoughCircles
                              10,    //10                   // min circle radius, in pixels
                              400);                     // max circle radius, in pixels
```



```
// Run this if the camera can see at least one circle
//for(i=0; i < p_seqCircles->total; i++) {      // for each element in sequential circles structure (i.e. for each object detected)
    if (p_seqCircles->total == 1)
    {
    p_fltXYRadius = (float*)cvGetSeqElem(p_seqCircles, 1);  // from the sequential structure, read the ith value into a pointer to a float

    printf("ball position x = %f, y = %f, r = %f \n", p_fltXYRadius[0],     // x position of center point of circle
                                                      p_fltXYRadius[1],     // y position of center point of circle
                                                      p_fltXYRadius[2]);    // radius of circle
```

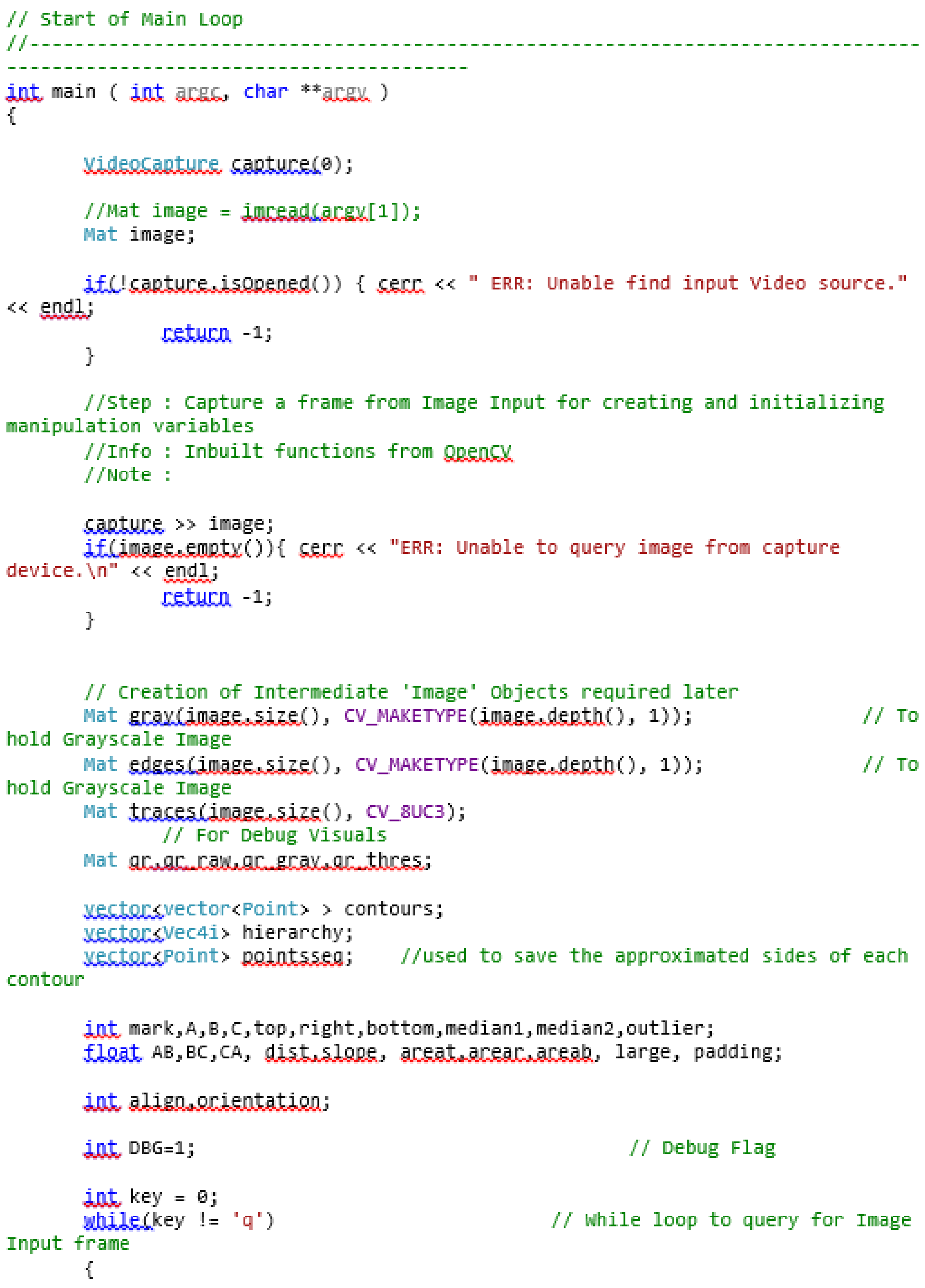


```
////////////////////////////////////////////////////////////////////
                                        // draw a small green circle at center of detected object
            cvCircle(p_imgOriginal,                                      // draw on the original image
                     cvPoint(cvRound(p_fltXYRadius[0]), cvRound(p_fltXYRadius[1])),      // center point of circle
                     3,                                                  // 3 pixel radius of circle
                     CV_RGB(0,255,0),                                    // draw pure green
                     CV_FILLED);                                         // thickness, fill in the circle

                                        // draw a red circle around the detected object
            cvCircle(p_imgOriginal,                                      // draw on the original image
                     cvPoint(cvRound(p_fltXYRadius[0]), cvRound(p_fltXYRadius[1])),      // center point of circle
                     cvRound(p_fltXYRadius[2]),                          // radius of circle in pixels
                     CV_RGB(255,0,0),                                    // draw pure red
                     3);                                                 // thickness of circle in pixels
        }   // end for

        cvShowImage("Original", p_imgOriginal);         // original image with detectec ball overlay
        cvShowImage("Processed", p_imgProcessed);       // image after processing

        cvReleaseMemStorage(&p_strStorage);             // deallocate necessary storage variable to pass into cvHoughCircles

        charCheckForEscKey = cvWaitKey(10);             // delay (in ms), and get key press, if any
        if(charCheckForEscKey == 27) break;             // if Esc key (ASCII 27) was pressed, jump out of while loop
    }   // end while

    cvReleaseCapture(&p_capWebcam);                 // release memory as applicable
```

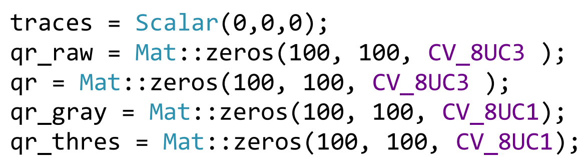


```
cvDestroyWindow("Original");
cvDestroyWindow("Processed");

// This closes the Serial Port
CloseHandle(hSerial);

return(0);
```

QR scanner program code snip:



```
// Start of Main Loop
//------------------------------------------------------------------------------
-----------------------------------------
int main ( int argc, char **argv )
{

       VideoCapture capture(0);

       //Mat image = imread(argv[1]);
       Mat image;

       if(!capture.isOpened()) { cerr << " ERR: Unable find input Video source."
<< endl;
              return -1;
       }

       //Step : Capture a frame from Image Input for creating and initializing
manipulation variables
       //Info : Inbuilt functions from OpenCV
       //Note :

       capture >> image;
       if(image.empty()){ cerr << "ERR: Unable to query image from capture
device.\n" << endl;
              return -1;
       }


       // Creation of Intermediate 'Image' Objects required later
       Mat gray(image.size(), CV_MAKETYPE(image.depth(), 1));               // To
hold Grayscale Image
       Mat edges(image.size(), CV_MAKETYPE(image.depth(), 1));              // To
hold Grayscale Image
       Mat traces(image.size(), CV_8UC3);
              // For Debug Visuals
       Mat qr,qr_raw,qr_gray,qr_thres;

       vector<vector<Point> > contours;
       vector<Vec4i> hierarchy;
       vector<Point> pointsseq;    //used to save the approximated sides of each
contour

       int mark,A,B,C,top,right,bottom,median1,median2,outlier;
       float AB,BC,CA, dist,slope, areat,arear,areab, large, padding;

       int align,orientation;

       int DBG=1;                                          // Debug Flag

       int key = 0;
       while(key != 'q')                            // While loop to query for Image
Input frame
       {
```



```
traces = Scalar(0,0,0);
qr_raw = Mat::zeros(100, 100, CV_8UC3 );
qr = Mat::zeros(100, 100, CV_8UC3 );
qr_gray = Mat::zeros(100, 100, CV_8UC1);
qr_thres = Mat::zeros(100, 100, CV_8UC1);
```

```
                capture >> image; // Capture Image from Image Input

                cvtColor(image,gray,CV_RGB2GRAY);        // Convert Image captured
from Image Input to GrayScale
                Canny(gray, edges, 100 , 200, 3);        // Apply Canny edge
detection on the gray image


                findContours( edges, contours, hierarchy, RETR_TREE,
CHAIN_APPROX_SIMPLE); // Find contours with hierarchy

                mark = 0;                                                  //
Reset all detected marker count for this frame

                // Get Moments for all Contours and the mass centers
                vector<Moments> mu(contours.size());
                vector<Point2f> mc(contours.size());

                for( int i = 0; i < contours.size(); i++ )
                {       mu[i] = moments( contours[i], false );
                        mc[i] = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 );
                }


                // Start processing the contour data

                // Find Three repeatedly enclosed contours A,B,C
                // NOTE: 1. Contour enclosing other contours is assumed to be the
three Alignment markings of the QR code.
                // 2. Alternately, the Ratio of areas of the "concentric" squares
can also be used for identifying base Alignment markers.
                // The below demonstrates the first method

                for( int i = 0; i < contours.size(); i++ )
                {
                        //Find the approximated polygon of the contour we are
examining
                        approxPolyDP(contours[i], pointsseq, arcLength(contours[i],
true)*0.02, true);
                        if (pointsseq.size() == 4)      // only quadrilaterals
contours are examined
                        {
                                int k=i;
                                int c=0;

                                while(hierarchy[k][2] != -1)
                                {
                                        k = hierarchy[k][2] ;
                                        c = c+1;
                                }
                                if(hierarchy[k][2] != -1)
                                c = c+1;

                                if (c >= 5)
                                {
                                        if (mark == 0)          A = i;

                                         else if  (mark == 1) B = i;        // i.e., A
is already found, assign current contour to B
```

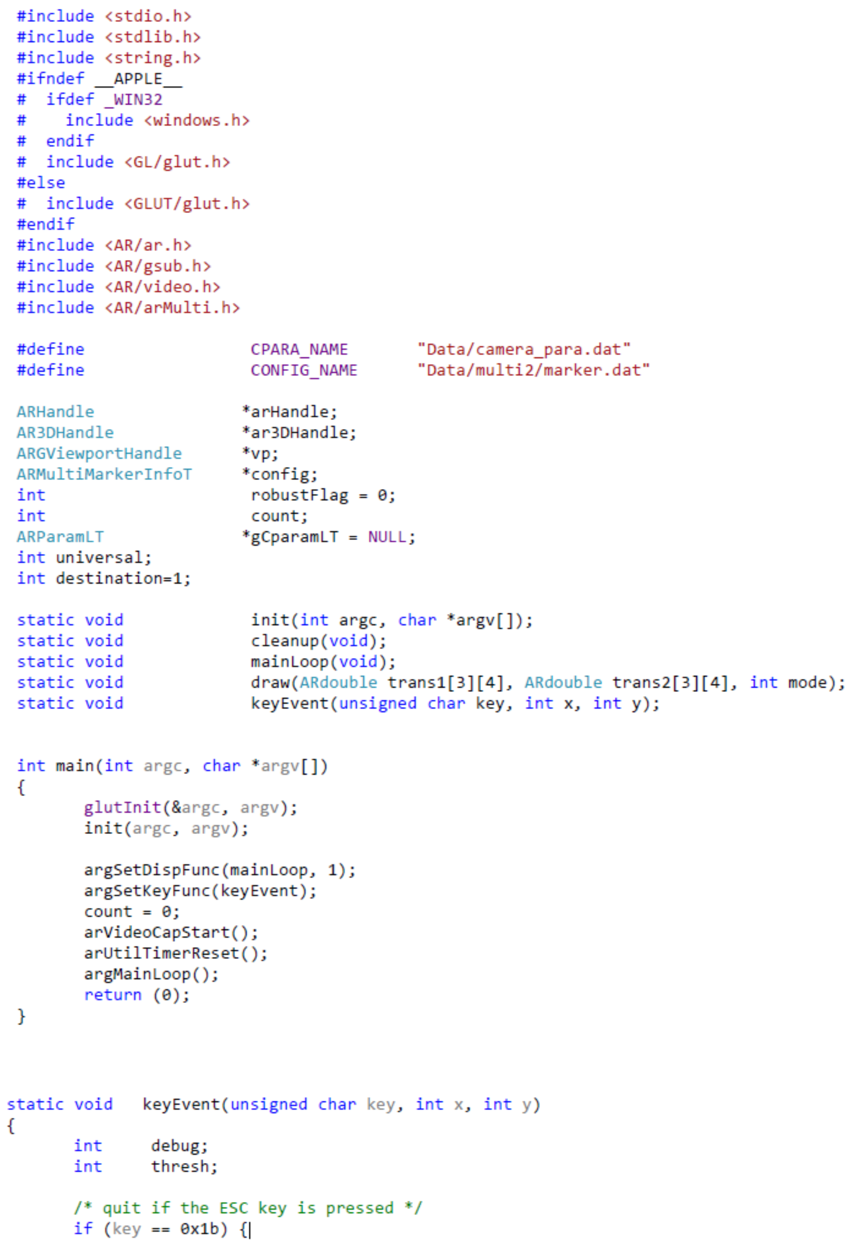
```
                                    else if  (mark == 2) C = i;        // i.e., A
and B are already found, assign current contour to C
                                    mark = mark + 1 ;
                             }
                       }
              }


              if (mark >= 3)              // Ensure we have (atleast 3; namely
A,B,C) 'Alignment Markers' discovered
              {
                     // We have found the 3 markers for the QR code; Now we need
to determine which of them are 'top', 'right' and 'bottom' markers

                     // Determining the 'top' marker
                     // Vertex of the triangle NOT involved in the longest side is
the 'outlier'

                     AB = cv_distance(mc[A],mc[B]);
                     BC = cv_distance(mc[B],mc[C]);
                     CA = cv_distance(mc[C],mc[A]);

                     if ( AB > BC && AB > CA )
                     {
                            outlier = C; median1=A; median2=B;
                     }
                     else if ( CA > AB && CA > BC )
                     {
                            outlier = B; median1=A; median2=C;
                     }
                     else if ( BC > AB && BC > CA )
                     {
                            outlier = A;  median1=B; median2=C;
                     }

                     top = outlier;
       // The obvious choice

                     dist = cv_lineEquation(mc[median1], mc[median2],
mc[outlier]); // Get the Perpendicular distance of the outlier from the longest
side
                     slope = cv_lineSlope(mc[median1], mc[median2],align);
       // Also calculate the slope of the longest side

                     // Now that we have the orientation of the line formed
median1 & median2 and we also have the position of the outlier w.r.t. the line
                     // Determine the 'right' and 'bottom' markers

                     if (align == 0)
                     {
                            bottom = median1;
                            right = median2;
                     }
                     else if (slope < 0 && dist < 0 )         // Orientation -
North
                     {
                            bottom = median1;
                            right = median2;
```

```
                    orientation = CV_QR_NORTH;
                }
                else if (slope > 0 && dist < 0 )         // Orientation -
East
                {
                        right = median1;
                        bottom = median2;
                        orientation = CV_QR_EAST;
                }
                else if (slope < 0 && dist > 0 )         // Orientation -
South
                {
                        right = median1;
                        bottom = median2;
                        orientation = CV_QR_SOUTH;
                }

                else if (slope > 0 && dist > 0 )         // Orientation -
West
                {
                        bottom = median1;
                        right = median2;
                        orientation = CV_QR_WEST;
                }


                // To ensure any unintended values do not sneak up when QR
code is not present
                float area_top,area_right, area_bottom;

                if( top < contours.size() && right < contours.size() &&
bottom < contours.size() && contourArea(contours[top]) > 10 &&
contourArea(contours[right]) > 10 && contourArea(contours[bottom]) > 10 )
                {

                        vector<Point2f> L,M,O, tempL,tempM,tempO;
                        Point2f N;

                        vector<Point2f> src,dst;         // src - Source
Points basically the 4 end co-ordinates of the overlay image

        // dst - Destination Points to transform overlay image

                        Mat warp_matrix;

                        cv_getVertices(contours,top,slope,tempL);
                        cv_getVertices(contours,right,slope,tempM);
                        cv_getVertices(contours,bottom,slope,tempO);

                        cv_updateCornerOr(orientation, tempL, L);
            // Re-arrange marker corners w.r.t orientation of the QR code
                        cv_updateCornerOr(orientation, tempM, M);
            // Re-arrange marker corners w.r.t orientation of the QR code
                        cv_updateCornerOr(orientation, tempO, O);
            // Re-arrange marker corners w.r.t orientation of the QR code

                        int iflag =
getIntersectionPoint(M[1],M[2],O[3],O[2],N);

                          src.push_back(L[0]);
                          src.push_back(M[1]);
                          src.push_back(N);
                          src.push_back(O[3]);
```

```
                            dst.push_back(Point2f(0,0));
                            dst.push_back(Point2f(qr.cols,0));
                            dst.push_back(Point2f(qr.cols, qr.rows));
                            dst.push_back(Point2f(0, qr.rows));

                            if (src.size() == 4 && dst.size() == 4 )
       // Failsafe for WarpMatrix Calculation to have only 4 Points with src and
dst
                            {
                                   warp_matrix = getPerspectiveTransform(src,
dst);
                                   warpPerspective(image, qr_raw, warp_matrix,
Size(qr.cols, qr.rows));
                                   copyMakeBorder( qr_raw, qr, 10, 10, 10,
10,BORDER_CONSTANT, Scalar(255,255,255) );

                                   cvtColor(qr,qr_gray,CV_RGB2GRAY);
                                   threshold(qr_gray, qr_thres, 127, 255,
CV_THRESH_BINARY);

                                   //threshold(qr_gray, qr_thres, 0, 255,
CV_THRESH_OTSU);
                                   //for( int d=0 ; d < 4 ; d++){
       src.pop_back(); dst.pop_back(); }
                            }

                            //Draw contours on the image
                            drawContours( image, contours, top ,
Scalar(255,200,0), 2, 8, hierarchy, 0 );
                            drawContours( image, contours, right ,
Scalar(0,0,255), 2, 8, hierarchy, 0 );
                            drawContours( image, contours, bottom ,
Scalar(255,0,100), 2, 8, hierarchy, 0 );

                            // Insert Debug instructions here
                            if(DBG==1)
                            {
                                   // Debug Prints
                                   // Visualizations for ease of understanding
                                   if (slope > 5)
                                          circle( traces, Point(10,20) , 5 ,
Scalar(0,0,255), -1, 8, 0 );
                                   else if (slope < -5)
                                          circle( traces, Point(10,20) , 5 ,
Scalar(255,255,255), -1, 8, 0 );

                                   // Draw contours on Trace image for analysis

                                   drawContours( traces, contours, top ,
Scalar(255,0,100), 1, 8, hierarchy, 0 );
                                   drawContours( traces, contours, right ,
Scalar(255,0,100), 1, 8, hierarchy, 0 );
                                   drawContours( traces, contours, bottom ,
Scalar(255,0,100), 1, 8, hierarchy, 0 );

// Draw points (4 corners) on Trace image for each Identification marker
              circle( traces, L[0], 2,  Scalar(255,255,0), -1, 8, 0 );

          circle( traces, L[1], 2,  Scalar(0,255,0), -1, 8, 0 );
```

```
                circle( traces, L[2], 2,  Scalar(0,0,255), -1, 8, 0 );
                circle( traces, L[3], 2,  Scalar(128,128,128), -1, 8, 0 );

                circle( traces, M[0], 2,  Scalar(255,255,0), -1, 8, 0 );
                circle( traces, M[1], 2,  Scalar(0,255,0), -1, 8, 0 );
                circle( traces, M[2], 2,  Scalar(0,0,255), -1, 8, 0 );
                circle( traces, M[3], 2,  Scalar(128,128,128), -1, 8, 0 );

                circle( traces, O[0], 2,  Scalar(255,255,0), -1, 8, 0 );
                circle( traces, O[1], 2,  Scalar(0,255,0), -1, 8, 0 );
                circle( traces, O[2], 2,  Scalar(0,0,255), -1, 8, 0 );
                circle( traces, O[3], 2,  Scalar(128,128,128), -1, 8, 0 );

                // Draw point of the estimated 4th Corner of (entire) QR Code
                circle( traces, N, 2,  Scalar(255,255,255), -1, 8, 0 );

                // Draw the lines used for estimating the 4th Corner of QR Code
                line(traces,M[1],N,Scalar(0,0,255),1,8,0);
                line(traces,O[3],N,Scalar(0,0,255),1,8,0);


                // Show the Orientation of the QR Code wrt to 2D Image Space
                int fontFace = FONT_HERSHEY_PLAIN;

                if(orientation == CV_QR_NORTH)
                 {
 putText(traces, "NORTH", Point(20,30), fontFace, 1, Scalar(0, 255, 0), 1, 8);
                 }
                 else if (orientation == CV_QR_EAST)
         {
putText(traces, "EAST", Point(20,30), fontFace, 1, Scalar(0, 255, 0), 1, 8);
         }
                else if (orientation == CV_QR_SOUTH)
         {
putText(traces, "SOUTH", Point(20,30), fontFace, 1, Scalar(0, 255, 0), 1, 8);
        }
                else if (orientation == CV_QR_WEST)
       {
putText(traces, "WEST", Point(20,30), fontFace, 1, Scalar(0, 255, 0), 1, 8);
       }

// Debug Prints
                            }

                       }
                }

                imshow ( "Image", image );
                imshow ( "Traces", traces );
                imshow ( "QR code", qr_thres );

                 if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If
'esc' key is pressed, break loop
        {
            cout << "esc key is pressed by user" << endl;
            break;
        }

}       // End of 'while' loop

return 0;
}
```

AR Rendering program code snip:



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#ifndef __APPLE__
#  ifdef _WIN32
#    include <windows.h>
#  endif
#  include <GL/glut.h>
#else
#  include <GLUT/glut.h>
#endif
#include <AR/ar.h>
#include <AR/gsub.h>
#include <AR/video.h>
#include <AR/arMulti.h>

#define                 CPARA_NAME       "Data/camera_para.dat"
#define                 CONFIG_NAME      "Data/multi2/marker.dat"

ARHandle               *arHandle;
AR3DHandle             *ar3DHandle;
ARGViewportHandle      *vp;
ARMultiMarkerInfoT     *config;
int                     robustFlag = 0;
int                     count;
ARParamLT              *gCparamLT = NULL;
int universal;
int destination=1;

static void             init(int argc, char *argv[]);
static void             cleanup(void);
static void             mainLoop(void);
static void             draw(ARdouble trans1[3][4], ARdouble trans2[3][4], int mode);
static void             keyEvent(unsigned char key, int x, int y);


int main(int argc, char *argv[])
{
       glutInit(&argc, argv);
       init(argc, argv);

       argSetDispFunc(mainLoop, 1);
       argSetKeyFunc(keyEvent);
       count = 0;
       arVideoCapStart();
       arUtilTimerReset();
       argMainLoop();
       return (0);
}


static void   keyEvent(unsigned char key, int x, int y)
{
       int     debug;
       int     thresh;

       /* quit if the ESC key is pressed */
       if (key == 0x1b) {
```

```
                ARLOG("*** %f (frame/sec)\n", (double)count / arUtilTimer());
                cleanup();
                exit(0);
        }

        if (key == 'd') {
                arGetDebugMode(arHandle, &debug);
                debug = 1 - debug;
                arSetDebugMode(arHandle, debug);
        }

        if (key == '1') {
                arGetDebugMode(arHandle, &debug);
                if (debug) {
                        arGetLabelingThresh(arHandle, &thresh);
                        thresh -= 5;
                        if (thresh < 0) thresh = 0;
                        arSetLabelingThresh(arHandle, thresh);
                        ARLOG("thresh = %d\n", thresh);
                }
        }
        if (key == '2') {
                arGetDebugMode(arHandle, &debug);
                if (debug) {
                        arGetLabelingThresh(arHandle, &thresh);
                        thresh += 5;
                        if (thresh > 255) thresh = 255;
                        arSetLabelingThresh(arHandle, thresh);
                        ARLOG("thresh = %d\n", thresh);
                }
        }

        if (key == ' ') {
                robustFlag = 1 - robustFlag;
                if (robustFlag) ARLOG("Robust estimation mode.\n");
                else            ARLOG("Normal estimation mode.\n");
        }
}

/* main loop */
static void mainLoop(void)
{
        ARUint8         *dataPtr;
        ARMarkerInfo    *marker_info;
        int             marker_num;
        int             imageProcMode;
        int             debugMode;
        double          err;
        int             i;



         /* grab a video frame */
         if ((dataPtr = (ARUint8 *)arVideoGetImage()) == NULL) {
                arUtilSleep(2);
                return;
         }

         if (count == 100) {
                ARLOG("*** %f (frame/sec)\n", (double)count / arUtilTimer());
                arUtilTimerReset();
                count = 0;
        }
        count++;

        /* detect the markers in the video frame */
        if (arDetectMarker(arHandle, dataPtr) < 0) {
                cleanup();
                exit(0);
        }
        marker_num = arGetMarkerNum(arHandle);
        marker_info = arGetMarker(arHandle);
        //printf("marker num= %i \n",marker_num);

        argDrawMode2D(vp);
        arGetDebugMode(arHandle, &debugMode);
        if (debugMode == 0) {
                argDrawImage(dataPtr);
        }
        else {
                arGetImageProcMode(arHandle, &imageProcMode);
                if (imageProcMode == AR_IMAGE_PROC_FRAME_IMAGE) {
                        argDrawImage(arHandle->labelInfo.bwImage);
                }
                else {
                        argDrawImageHalf(arHandle->labelInfo.bwImage);
                }
                glColor3f(1.0f, 0.0f, 0.0f);
                glLineWidth(2.0f);
                for (i = 0; i < marker_num; i++) {
                        argDrawSquareByIdealPos(marker_info[i].vertex);
                }
                glLineWidth(1.0f);
        }

        if (robustFlag) {
                err = arGetTransMatMultiSquareRobust(ar3DHandle, marker_info, marker_num,
config);
        }
        else {
                err = arGetTransMatMultiSquare(ar3DHandle, marker_info, marker_num,
config);
        }
        if (config->prevF == 0) {
                argSwapBuffers();
                return;
        }
        //ARLOGd("err = %f\n", err);


        argDrawMode3D(vp);
        glClearDepth(1.0);
        glClear(GL_DEPTH_BUFFER_BIT);
        for (i = 0; i < config->marker_num; i++) {
                if (config->marker[i].visible >= 0)
                {
                        draw(config->trans, config->marker[i].trans, 0);
                        printf("marker [i] = %i \n", i);
```

```
                    switch (destination)
                    {
                    case 1:
                    {
                           printf("Library \n");
                           if (i == 0) universal = 0;
                           if (i == 1) universal = 1;
                           if (i == 2) universal = 1;
                           break;
                    }
                    case 2:
                    {
                           printf("HOME \n");
                           if (i == 0) universal = 2;
                           if (i == 1) universal = 0;
                           if (i == 2) universal = 1;
                           break;
                    }
                    case 3 :
                    {
                           printf("CH \n");
                           if (i == 0) universal = 2;
                           if (i == 1) universal = 2;
                           if (i == 2) universal = 0;
                           break;
                    }


                    //if (i == 0) universal = 1;
                    }
             }
             else                                  draw(config->trans,
config->marker[i].trans, 1);
       }

       argSwapBuffers();
}

static void   init(int argc, char *argv[])
{
       ARParam          cparam;
       ARGViewport      viewport;
       ARPattHandle    *arPattHandle;
       char             vconf[512];
       char             configName[512];
       int              xsize, ysize;
       AR_PIXEL_FORMAT pixFormat;
       int              i;


       configName[0] = '\0';
       vconf[0] = '\0';
       for (i = 1; i < argc; i++) {
              if (strncmp(argv[i], "-config=", 8) == 0) {
                     strcpy(configName, &argv[i][8]);
              }
              else {
                     if (vconf[0] != '\0') strcat(vconf, " ");
                     strcat(vconf, argv[i]);
```

```
    }
}
if (configName[0] == '\0') strcpy(configName, CONFIG_NAME);

/* open the video path */
if (arVideoOpen(vconf) < 0) exit(0);
/* find the size of the window */
if (arVideoGetSize(&xsize, &ysize) < 0) exit(0);
ARLOGi("Image size (x,y) = (%d,%d)\n", xsize, ysize);
if ((pixFormat = arVideoGetPixelFormat()) < 0) exit(0);

/* set the initial camera parameters */
if (arParamLoad(CPARA_NAME, 1, &cparam) < 0) {
    ARLOGe("Camera parameter load error !!\n");
    exit(0);
}
arParamChangeSize(&cparam, xsize, ysize, &cparam);
ARLOG("*** Camera Parameter ***\n");
arParamDisp(&cparam);
if ((gCparamLT = arParamLTCreate(&cparam, AR_PARAM_LT_DEFAULT_OFFSET)) == NULL) {
    ARLOGe("Error: arParamLTCreate.\n");
    exit(-1);
}

if ((arHandle = arCreateHandle(gCparamLT)) == NULL) {
    ARLOGe("Error: arCreateHandle.\n");
    exit(0);
}
if (arSetPixelFormat(arHandle, pixFormat) < 0) {
    ARLOGe("Error: arSetPixelFormat.\n");
    exit(0);
}

if ((ar3DHandle = ar3DCreateHandle(&cparam)) == NULL) {
    ARLOGe("Error: ar3DCreateHandle.\n");
    exit(0);
}

if ((arPattHandle = arPattCreateHandle()) == NULL) {
    ARLOGe("Error: arPattCreateHandle.\n");
    exit(0);
}
arPattAttach(arHandle, arPattHandle);

if ((config = arMultiReadConfigFile(configName, arPattHandle)) == NULL) {
    ARLOGe("config data load error !!\n");
    exit(0);
}
if (config->patt_type == AR_MULTI_PATTERN_DETECTION_MODE_TEMPLATE) {
    arSetPatternDetectionMode(arHandle, AR_TEMPLATE_MATCHING_COLOR);


else if (config->patt_type == AR_MULTI_PATTERN_DETECTION_MODE_MATRIX) {
    arSetPatternDetectionMode(arHandle, AR_MATRIX_CODE_DETECTION);
}
else { // AR_MULTI_PATTERN_DETECTION_MODE_TEMPLATE_AND_MATRIX
    arSetPatternDetectionMode(arHandle, AR_TEMPLATE_MATCHING_COLOR_AND_MATRIX);
}
```

```
        /* open the graphics window */
        viewport.sx = 0;
        viewport.sy = 0;
        viewport.xsize = xsize;
        viewport.ysize = ysize;
        if ((vp = argCreateViewport(&viewport)) == NULL) exit(0);
        argViewportSetCparam(vp, &cparam);
        argViewportSetPixFormat(vp, pixFormat);
}

/* cleanup function called when program exits */
static void cleanup(void)
{
        arParamLTFree(&gCparamLT);
        arVideoCapStop();
        arVideoClose();
        argCleanup();
}

static void draw(ARdouble trans1[3][4], ARdouble trans2[3][4], int mode)
{
        ARdouble  gl_para[16];
        GLfloat   light_position[] = { 100.0f, -200.0f, 200.0f, 0.0f };
        GLfloat   light_ambi[] = { 0.1f, 0.1f, 0.1f, 0.0f };
        GLfloat   light_color[] = { 1.0f, 1.0f, 1.0f, 0.0f };
        GLfloat   mat_flash[] = { 1.0f, 1.0f, 1.0f, 0.0f };
        GLfloat   mat_flash_shiny[] = { 50.0f };
        GLfloat   mat_diffuse[] = { 0.0f, 0.0f, 1.0f, 1.0f };
        GLfloat   mat_diffuse1[] = { 1.0f, 0.0f, 0.0f, 1.0f };
        int       debugMode;

        glEnable(GL_DEPTH_TEST);
        glDepthFunc(GL_LEQUAL);

        /* load the camera transformation matrix */
        glMatrixMode(GL_MODELVIEW);
        argConvGlpara(trans1, gl_para);
#ifdef ARDOUBLE_IS_FLOAT
        glLoadMatrixf(gl_para);
#else
        glLoadMatrixd(gl_para);
#endif
        argConvGlpara(trans2, gl_para);
#ifdef ARDOUBLE_IS_FLOAT
        glMultMatrixf(gl_para);
#else
        glMultMatrixd(gl_para);
#endif


       glEnable(GL_LIGHTING);
       glEnable(GL_LIGHT0);
       glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, 1);
       glLightfv(GL_LIGHT0, GL_POSITION, light_position);
       glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambi);
       glLightfv(GL_LIGHT0, GL_DIFFUSE, light_color);
       glLightfv(GL_LIGHT0, GL_SPECULAR, light_color);
       glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
       glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
```

```
	if (mode == 0) {
		glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
		glMaterialfv(GL_FRONT, GL_AMBIENT, mat_diffuse);
	}
	else {
		glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse1);
		glMaterialfv(GL_FRONT, GL_AMBIENT, mat_diffuse1);
	}
	//////////////////////////////////////////draw
	glMatrixMode(GL_MODELVIEW);
	glPushMatrix();
	glTranslatef(0.0f, 0.0f, 20.0f);
	if (universal==1)
	glRotatef(90.0, 1.0, -90.0, 0.0); //kiri
	if (universal == 2)
		glRotatef(90.0, 1.0, 90.0, 0.0); //kanan
	else
		glRotatef(90.0, 1.0, 90.0, 0.0); //home
	//glRotatef(50, 1.0, 90.0, 0.0);  //kanan
	arGetDebugMode(arHandle, &debugMode);
	if (debugMode == 0) glutSolidCone(10.0,40.0,20,24);
	else                glutWireCube(40.0);
	glPopMatrix();

	glDisable(GL_LIGHT0);
	glDisable(GL_LIGHTING);
	glDisable(GL_DEPTH_TEST);
}
```