EMBEDDED SYSTEM OBJECT TRACKING USING WEBCAM

By

NG JIA YAN

FINAL PROJECT REPORT

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

CERTIFICATION OF APPROVAL

EMBEDDED SYSTEM OBJECT TRACKING USING WEBCAM

by

Ng Jia Yan

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 & Electronics Engineering)

Approved:

____

Mr. Patrick Sebastian
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

August 2014

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.

____

Ng Jia Yan

Abstract

The extensive availability of hardware devices and intensive expansion their computing power have been the catalyst behind the rapid development of computer vision. In this project, an implementation of object tracking in an inexpensive and small embedded system platform is presented. The tracking system comprised of two Raspberry Pis with two different cameras used: a webcam and Raspicam module. Three communication connection models of the system are discussed in this paper for establishing communication between the two Raspberry Pis. Data sharing between these two hardware platforms is the proposed solution for resolving the limited processing power each platform possesses. The SimpleCV, an open source framework that provides free computer vision libraries that is useful for object detection and tracking algorithm development.

Acknowledgment

I would like to express my greatest appreciation to my supervisor, Mr. Patrick for allowing me to carry out this project under his under supervision. Without his guidance and support I wouldn’t have completed the project. Secondly, I would like to thank my family and friends who have been very supportive throughout my final year study. Lastly I would like to thank Universiti Teknologi Petronas for providing the resources and opportunity to assist me in completing my final year project.

LIST OF CONTENTS

Abstract………………………………………………………………………. i
Acknowledgment ………………………………………………………………. ii
LIST OF CONTENTS………………………………………………………. iii
LIST OF TABLES……………………………………………………………… v
LIST OF FIGURES…………………………………………………………….. vi
LIST OF ABBREVIATIONS…………………………………………………… vii
Chapter 1
Introduction……………………………………………………………………… 1
1.1 Background…………………………………………………………………….. 2
1.2 Problem Statement …………………………………………………………….. 3
1.3 Objectives and Scope of Study………………………………………………… 3
Chapter 2
Literature Review ……………………………………………………………….. 4
2.1 Introduction ……………………………………………………………………. 4
2.2 Related Work…………………………………………………………………… 4
2.3 Raspberry Pi's System Specification And Limitations……………………… 6
2.4 Object Detection And Tracking Algorithms………………………………… 6
2.4.1 Object Detection……………………………………………………………….. 6
2.4.2 Object Tracking ……………………………………………………………….. 7
2.5 Framework for Object Tracking For Raspberry Pi ………………………… 8
Chapter 3
Project Methodology …………………………………………………………… 9
3.1 Project Flow Overview………………………………………………………… 9
3.2 Object Detection & Tracking Implementation……………………………… 10
3.3 Tracking System proposed using two Raspberry Pis……………………….. 11
3.3.1 Tracking System Model 1……………………………………………………. 12
3.3.1 Tracking System Model 2…………………………………………………… 12
3.3.3 Tracking System Model 3……………………………………………………. 13
3.4 Project Key Milestone…………………………………………………………. 14
3.5 Gantt Chart…………………………………………………………………….. 15
Chapter 4
Results & Discussion ………………………………………………………….. 16
4.1 Webcam compatibility issue ………………………………………………….. 16
4.1.1 Compatibility in Raspbian & SimpleCV…………………………………. 16
4.1.2 Models of Tested Webcam…………………………………………………….. 17
4.2 Object Detection & Object Tracking…………………………………………. 18
4.2.1 Basic Shape Tracking using Feature Detections …………………………. 18
4.2.2 White Ball Tracking with color and blobs detection……………………… 19
4.2.3 Tennis Ball Tracking using CAMShift algorithm ………………………… 20
4.2.3 Color Spectrum & Lighting effects on algorithm ………………………… 21
4.2.4 Face Detection & Tracking…………………………………………………… 22
4.2.5 Face Recognition ……………………………………………………………. 24
4.3 Communication between two raspberry pis………………………………… 25
4.3.1 Wireless communication ……………………………………………………. 25
4.3.2 Wired Communication through Ethernet………………………………….. 26

4.3.3 Wired Communication via router…………………………………………..… 27
Chapter 5
Recommendation & Conclusion ………………………………………………… 29
References………………………………………………………………………. 30

LIST OF TABLES

Table 1: Key Milestone of Project……………………………………… 14
Table 2: Project Plan with Gantt Chart ………………………………….15
Table 3: List of not working webcam…………………………………… 17
Table 4 : impact of lighting on color detection………………………….. 21

LIST OF FIGURES

Figure 1: Project Flow Chart………………………………………… 9
Figure 2: Object Detection & Tracking Flow Diagram ……………….. 10
Figure 3: General block of tracking system proposed…………………. 11
Figure 4: Model 1 for Overall System connection…………………….. 12
Figure 5: Model 2 for Overall System connection……………………… 12
Figure 6: Model 3 for Overall System connection……………………… 13
Figure 7: camera driver Incompatibility………………………………… 16
Figure 8: camera undetected in SimpleCV…………………………….. 16
Figure 9: Error message shown in SimpleCV…………………………… 17
Figure 10: Circle Detection………………………………..…………….. 18
Figure 11: Series of image for ball tracking……………………………… 19
Figure 12: Tennis Ball Tracking result using CAMshift algorithm……… 20
Figure 13: Series of Image captured during face tracking………………. 22
Figure 14: face detection sensitivity in various head angle……………… 23
Figure 15: Face recognition using key points matching technique……… 24
Figure 16: Raspberry pi to pi communication via wireless router………. 25
Figure 17: internet message shown in router connection attempt……….. 25
figure 18: communication through ethernet with HDMI display………… 26
Figure 19: Error message show when tracking done with HDMI display.. 26
Figure 20: communication via router with VNC remote desktop display….27
Figure 21: SSH communication & tracking result on second raspberry pi.. 28

LIST OF ABBREVIATIONS

CAMShift……	Continuously Adaptive Mean Shift
CPU…..………	Central Processing Unit
CCTV………..	Closed Circuit Television
DSP……….…	Digital Signal Processor
GI……………	Gastro-intestinal
GPU…………	Graphics Processing Unit
HDMI……….	High Definition Multimedia Interface
MATLAB……	Matrix Laboratory
OpenCV…..…	Open Source Computer Vision
RCA…………	Radio Corporation of America (cable)
RPI………..…	Raspberry Pi
SD………..…	Secure Digital Memory Card
SimpleCV…..	Simple Source Computer Vision
SFTP……..…	Secure File Transfer Protocol
SSH…………	Secure Shell
SoC…………	System on Chip
V4l2…………	Video4Linux2
VNC…………	Virtual Network Computing
WLAN……….	Wireless Local Area Network

Chapter 1

Introduction

Evolution of information technology, computation and telecommunication in late 20th century has marked the transformation of industrial age to information age. We can witness the advancement of the technology continues until today. With the advent of technologies available nowadays, it has not just changed the way how information is being disseminated but also our interaction with these technologies. Various forms of multimedia devices such as cell phones, laptops, MP4 players, video camera, tablets, etc. are basically ubiquitous in our daily lives. Undoubtedly, this demanding application couldn't have been satisfied without progression in implementation of signal processing on embedded systems. In recent decades, because of the advancement of telecommunication in transmitting data with faster rate, image and video processing has become a popular topic to be explored in the field of computer science and electronic engineering. There are also demanding commercial needs in the application of image and video processing such as smart surveillance and security.

Computer vision comprised of integration of diverse fields such as engineering, mathematics, biology, physics and computer science. At the early stage of development, its applications were limited to government sector or large cooperation with high capital investment due to costly hardware platform required for image and video processing. Through technology advancement in recent decades, there are demands in implementing its applications with the use of simpler and less expensive hardware platform. Its progression from being a niche tool to becoming a commonly used utility in the public sector is mainly due to the increasing availability of inexpensive hardware that can cope with the processing requirements.

The key objective of this project is to explore and utilize the capacity of Raspberry Pi in developing an object tracking platform. With webcam serves as the eye and the Raspberry Pi being the brain of the tracking system, a predefined object presents within the camera framework will be tracked over time under certain constraints such as limited number of object, simple object appearance, fixed camera motion and good lighting condition. The secondary goal of this project is to expand the functionality of the tracking platform by establishing a communication network, allowing the transfer of data processed and acquired by the two raspberry pi. This allows the targeted object being tracked over longer distance with the use of multiple cameras.

The studies comprised of the interfacing of Raspberry Pi with external peripherals such as WIFI dongle and webcam, finding suitable tracking algorithm to be implemented in the Linux operating system, communication networks in Raspberry Pi and Python programming in SimpleCV. The primary phase of the project focuses on the interfacing of Raspberry Pi with compatible webcam model and data processing in SimpleCV framework. In the meanwhile the network communication feature is first built upon the establishment of internet access within the Raspberry Pi.

1.1 Background

Object tracking is one of the crucial features in image and video processing. The relentless demand of its application has been driving the interest of researchers and engineers in this scope of study. We can find the trace of object tracking implementation in various fields such as industrial inspection system, surveillance, human-computer interaction, traffic monitoring, vehicle navigation and so on [1]. The performance improvement of personal computer has served as catalyst to drive the development of image and video processing algorithm. Image and video processing typically involves complex algorithm to process intensive amount of data. Therefore specialized processor is most often required to perform the sophisticated computations involved in the algorithm. [2]

1.2 Problem Statement

Tremendous amount of object tracking algorithms have been developed in recent decades, however they are normally implemented in conventional computer system which mostly processes high processing technology and high power consumption. This circumstance is gradually changing due to the booming of embedded system hardware platforms in recent years. This phenomenon becomes the driving force in a growing interest of the application of these algorithms within an ease to use, smaller and cheaper hardware platform. For this particular project, the target is to tackle this interest by attempting to develop a tracking platform using Raspberry pi which was released in 2012.

1.3 Objectives and Scope of Study

This project aims for exploiting the capacity of raspberry pi technology in order to develop a tracking platform with compatible tracking algorithm for the raspberry pi's system. The area and scope of work involves interfacing between hardware/circuit with raspberry pi and software development which comprised of tracking algorithm implementation. There is also an interest of exploring the data communication transmission of raspberry pi through a network for tracking purpose. This concept is to allow two cameras to "communicate" to each other and share the data acquired respectively for tracking an object in a wider framework. This project model simulates the fundamental principle of a surveillance system which can track a specific object over certain distances with multiple of cameras. All in all, the two primary goals of this project are to develop a tracking system which is able locate a specific object and perform tracking within its framework and establish a smart tracking system where two cameras can track the target within its respective frameworks and share the data acquired to each other through a network.

Chapter 2

Literature Review

2.1 Introduction

According to [2] image and video processing can be a challenging task because the system developed involves integration of optimized software approaches in a sustainable hardware platform. At the glance of hardware platforms progression for image and video processing, from the earliest known image processing device which is a picture scanner to recent processor technology of a powerful single chip processor called graphics processing unit (GPU) [2], we can see that the opportunity to implement complex processing algorithm in a lower power-consumed and smaller sized hardware platform has been opened up.

Apart from the availability of the hardware platforms, there are tremendous amount of image and video processing algorithms have been developed over years. When it comes to object tracking algorithm, the two key principles in performing object tracking are firstly detecting the objects of interest followed by tracking them in each frames. [3]. When deciding which tracking algorithm is the best fit, physical constraints such as the shape and appearance of objects, the amount of tracked objects, the lighting condition and the camera movement have to be taken into account. The discrepancy of algorithms in object tracking basically depends on the approaches used to tackle those issues. [1]

2.2 Related Work

The public request of more applications in portable devices has resulted in the growing interest in migrating works which were once done on general-purpose computers into portable devices. Numerous projects or research works had been tested using embedded system platforms to satisfy this increasing demand. Conventional surveillance systems utilize CCTV cameras to perform monitoring whereas data captured will be processed solely by computation workstation. There is an implementation of embedded smart camera to perform video analysis and video compression for traffic surveillance system as in [4]. A comparable performance of running high-level video processing algorithm has been achieved in a digital signal processor (DSP) platform.

As published in [5], face recognition systems have also been developed with DSP-based system to meet the demand of low power consumption and low cost systems in portable devices. As a matter of fact, hardware platforms' size, mobility and affordability are now the three key factors that are being taken into consideration in the system design.

There are also handful of projects make use of the advancing microcontroller technology to accommodate that need. Those applications are as straightforward as a colourful object tracking or as innovative as object tracking inside our gastro-intestinal (GI) tract. [6] [7]. As outlined in [2], hardware platform for image and video processing has been evolving over time, allowing researchers and engineers the opportunity to exploring the optimized combination of hardware platform and software approach. Similar object tracking application can also found in [3] [8] by using FPGA-based platform. Within a similar hardware platform, different algorithms or optimization approaches can be used to target different aspect of problems in image and video processing. All in all, the integration between the hardware and software is the key determinant in the success of the system.

Among all the functionalities of video analysis, tracking is the fundamental part in computer vision system. It has been a hot topic for researches over past years. [1] [3] [7] [8]. Its significant can even be traced in the most recent research topic of augmented reality- a markerless tracking solution. [9]. Aside from high-end tracking solution in sophisticated technology, people are also more interested to seeking for object tracking implementation in a cheaper, portable and lower power consumption hardware platform. After all, as technology advances, it is no longer a privilege for specific government agency or owned by a secret taskforce. From the project works in [10] [11], numerous of object tracking projects are completed with the use of raspberry pi, a credit card-size microcomputer which released to the public in 2012. [12]

2.3 Raspberry Pi's System Specification And Limitations

The Raspberry Pi(RPi) has a Broadcom BCM2835 system on chip (SoC), combing with a low power ARM1176JZF-S 700 MHz processor and a Broadcom VideoCore IV GPU. It comes in two versions: Model A and Model B. This project uses Model B since it has higher specifications with 512MB of RAM, an onboard 100mb Ethernet port and two USB ports. There is an SD card slot for data storage, 3.5mm audio output jack, RCA and HDMI video output. [13] There is always a trade-off between power consumption and performance. RPi's performance is limited with its small size and low power consumption. Comparisons that were done by the Raspberry Foundations showed that its overall performance is on a par with a 300MHz clocked Intel Pentium 2 processor and its graphics capabilities are more or less the same as the original Microsoft Xbox games console. [14]

2.4 Object Detection And Tracking Algorithms

2.4.1 Object Detection

Object detection mechanism is prerequisite to any tracking algorithms either during the object's first appearance in the frame or in every frame of the video feed. The objects are identified and classified through its characteristic such as size, position and its rotation. In order to accomplish object detection, image in each frame need to be divided into segments, eliminate unrelated information or extract important features from the images. [15] Therefore most of the object detection make use of information extracted in single frame. Nonetheless false detections are likely to happen with this method. In order to tackle this issue, some algorithms calculate the temporal information from the frames sequence.

Object detection methods are classified into four main categories: point detection, segmentation, background modeling and supervised classifiers. [1] Point detection utilizes interest points in images which are distinguish with respective to its vicinity. In segmentation algorithms, images are subdivided into similar regions. Good partition criteria and ways to achieve efficient partitioning are the most important issues need to be addressed in this technique. A background model which is a representation of the scene is built and compared to each incoming frame to find the deviations. The significant change found in the modeling indicates an object motion. For supervised learning, a complete set of learning templates are required to be stored as the inputs would be mapped to desired outputs with the examples given. [1]

2.4.2 Object Tracking

As according to [15], the three main components in a typical object tracking system are object representation, dynamic model and search engine. Object representation typically uses both appearance and shape models or either one of them. Each model has its processing limitation on the deformation or the type of motion. In [1], tracking algorithms are categories into pointing tracking, kernel tracking and silhouette tracking. Point tracking as its name suggested, point representation is used in every consecutive frames. The point association is basically based on the previous object position and motion. In kernel tracking, object is represented by its shape and appearance. The computation of kernel motion in consecutive frames is then used to track the object. On the other hand, tracking with silhouette approach is carried out through the estimation of object region in each frame.

To minimize the complexity in computation of tracking algorithm, typical tracking system uses a dynamic model which uses default or learned training data to estimate the target states. However it normally requires faster processor since it is challenging to implement this model in object moving in high velocity. The optimization of the tracking algorithm is done through the search mechanism where the strategy relies on the objective function. Exhaustive search engine accomplish high tracking performance with the expense of high computational load. On contrary, decent tracking performance can only achieved with sampling-based search approaches provided there is no significant change in the object states. The trade-off between these two approaches called stochastic search mechanism is then used.

2.5 Framework for Object Tracking For Raspberry Pi

There are quite number of software framework for building computer vision applications. Among those software framework, OpenCV offers wide range of algorithm implementations such as image acquisition, segmentation, object detection, tracking, decoding and encoding. [16] There are tremendous work on tracking applications using OpenCV library. [17][18][19]. Likewise, MATLAB is also a very popular computing tool used for computer vision and image processing. It offers its users an highly interactive environment with great help section documented with various examples and explanations, allowing them to quickly develop algorithms. It also comes with a set of debugging features where users are allowed to fix the bug in execution mode. [20]. Nonetheless, when it comes to processing time, OpenCV is faster than MATLAB in executing the image processing algorithms. such as smoothing, image pyramid, sober derivate and canny edge detection. On the other hand, more source code need to be written to perform same function in OpenCV. [21] With SimpleCV, few lines of code are required to accomplish the same functionality compared to other open source libraries. It is a wrapper for OpenCV which bundles opens source computer vision libraries together in Python as programming language. By streamlining most of the common tasks, it is easier to develop computer vision systems with SimpleCV. Even though SimpleCV has fewer features than MATLAB and OpenCV have but it offers substantial simple computer vision applications. It allows loading images directly from the internet which MATLAB and OpenCV don’t have. [22] Judging from speed perspective, SimpleCV lies in the middle between MATLAB and OpenCV but since it wrappers OpenCV, the applications can be ported to OpenCV to speed up the execution time. Similar to MATLAB, it offers good debugging features and memory management. Leveraging on the hardware platform used in this project and the simplicity of algorithm implementation in SimpleCV, development of processing algorithms in this framework for this project is justifiable.

Chapter 3

Project Methodology

3.1 Project Flow Overview

Academic documentation regarding raspberry pi is scarcely available since the technology is considerably new to the market. Very few research papers related to object tracking using raspberry pi have been published. The main reference for developing tracking algorithm using SimpleCV in raspberry pi mainly came from internet sources such as forums and official information released by raspberry pi foundation. Whereas study of various tracking algorithms proposed in the reviewed research papers for embedded system implementation is done to find its suitability to be implemented in raspberry pi system as according to its hardware limitation. This project is carried out according to phases as follow:

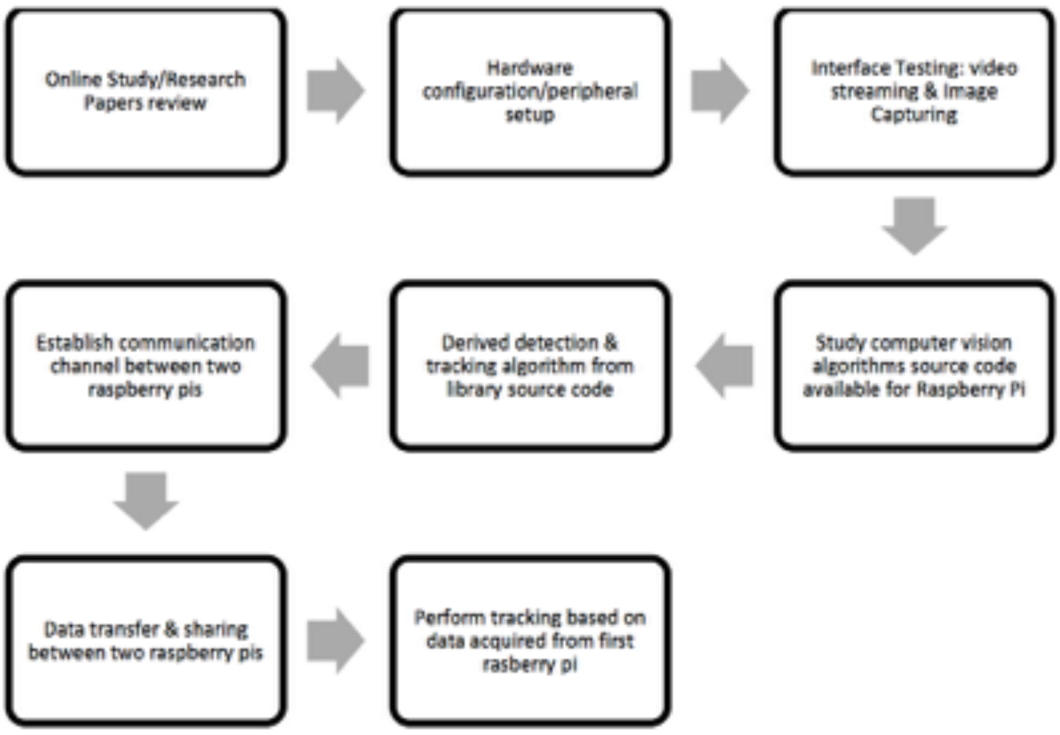

Online Study/Research Papers review
Hardware configuration/peripheral setup
Interface Testing: video streaming & Image Capturing
Study computer vision algorithms source code available for Raspberry Pi
Derived detection & tracking algorithm from library source code
Establish communication channel between two raspberry pis
Data transfer & sharing between two raspberry pis
Perform tracking based on data acquired from first rasberry pi

FIGURE 1: PROJECT FLOW CHART

3.2 Object Detection & Tracking Implementation

Object to be tracked need to be identified before tracking algorithm is initiated. The most basic tracking system can only involve in tracking any object moving in the frame without identifying what is the object. In this project, object detection is first achieved before initializing tracking algorithm. Features detection algorithms are explored in depth to study the capability of raspberry pi in implement object tracking system. The implementation of the system comprised of steps illustrated as diagram below.

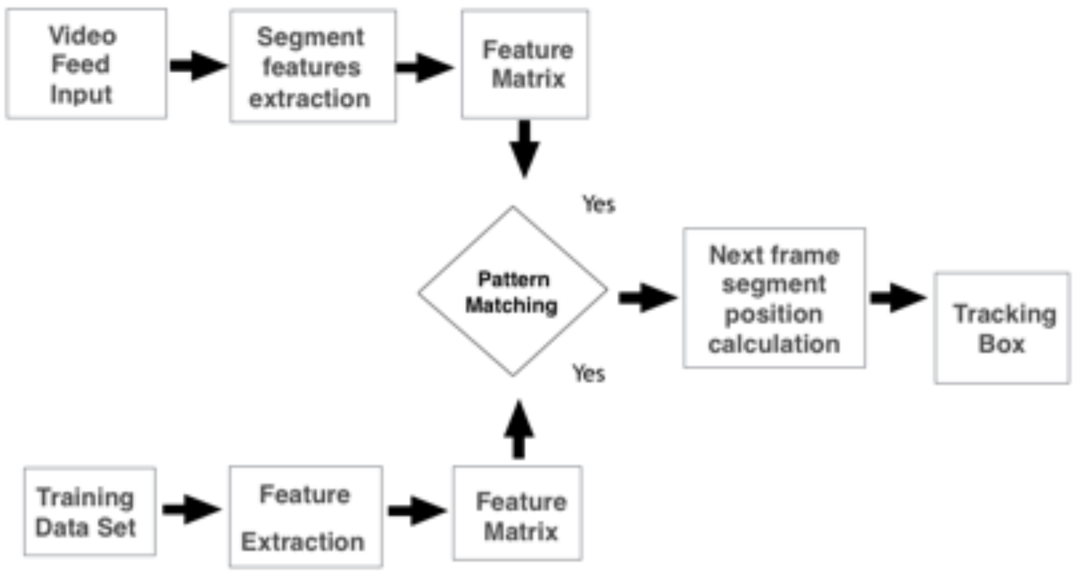


FIGURE 2: OBJECT DETECTION & TRACKING FLOW DIAGRAM

Segment features of moving object will firstly be extracted and description of the moving object will then be compared against to the object description for the target. Once the system is able to recognize its targeted object, the current segment position is captured and movement projection in next frame will be estimated and hence tracking of the particular object begins.

3.3 Tracking System proposed using two Raspberry Pis

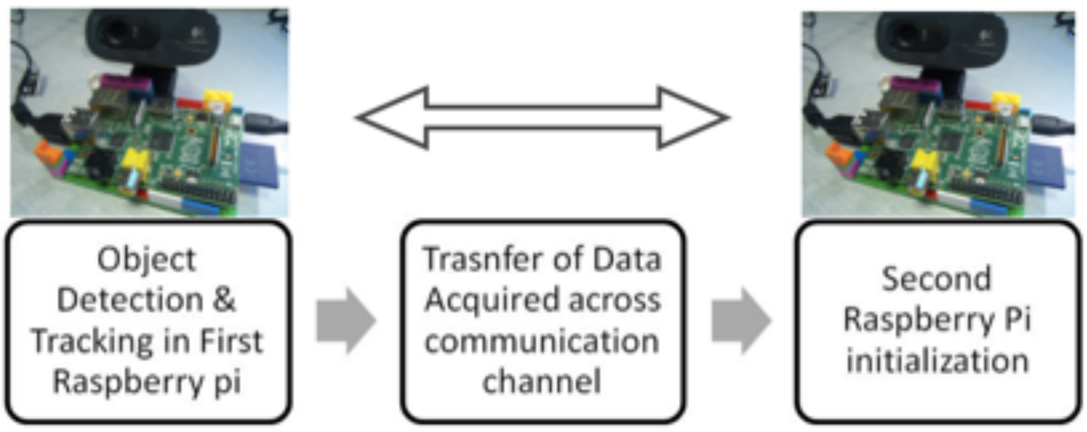

Object Detection & Tracking in First Raspberry pi
Trasnfer of Data Acquired across communication channel
Second Raspberry Pi initialization

FIGURE 3: GENERAL BLOCK OF TRACKING SYSTEM PROPOSED

Processing capability of one raspberry pi very limited to achieve complex computational tasks like object detection, object tracking and object recognition in high performance. The proposed solution in this project is to establish a communication channel between two raspberry pis in order for them to share data needed to accomplish their respective tasks. Through this manner, each raspberry pi can compensate each other processing power and perform tasks with lesser data loaded.

In this project, the tracking system of the second raspberry pi will only be initiated when it receives data acquired by the first raspberry pi. The first raspberry pi will pass the data when the target leaves its camera view. The second pi will then process the data received from the first raspberry pi to perform object detection and tracking. In this case the data required for object detection does not have to be preloaded to the second raspberry pi and hence the objective of load sharing between two pis is accomplished.

3.3.1 Tracking System Model 1

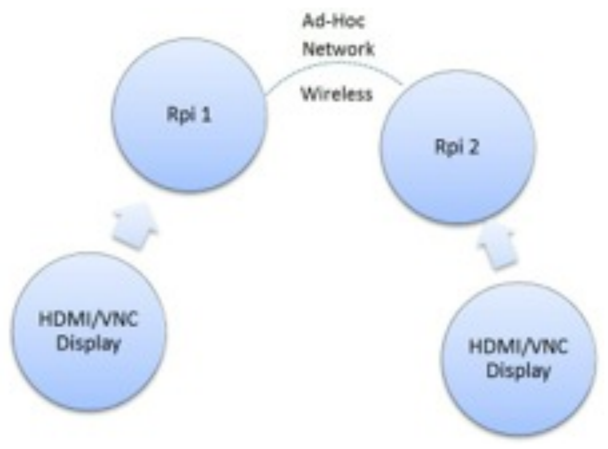


FIGURE 4: MODEL 1 FOR OVERALL SYSTEM CONNECTION

In this model, the two raspberry pis are connected wireless to a same network and ad-hoc network is establish for communication between them. The respective raspberry pi has its own desktop display either with HDMI display or VNC remote desktop. The method used in this model is ideal solution for raspberry pi to be used under wider range of locations since data can be exchanged through wireless network.

3.3.1 Tracking System Model 2

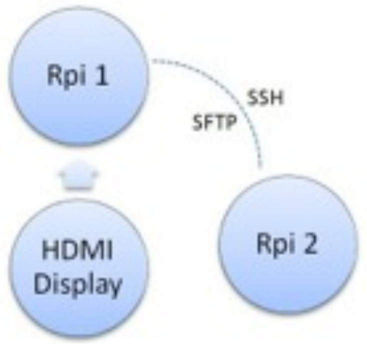


FIGURE 5: MODEL 2 FOR OVERALL SYSTEM CONNECTION

Model 2 has the first raspberry pi connected to keyboard and mouse with HDMI display. In the meanwhile the ethernet port of second raspberry pi is connected to first raspberry pi's ethernet port. Wired communication between two raspberry pis can be accomplished through directly connecting Ethernet to their respective ports. Data transfer is done through secure file transfer protocol (SFTP)

3.3.3 Tracking System Model 3

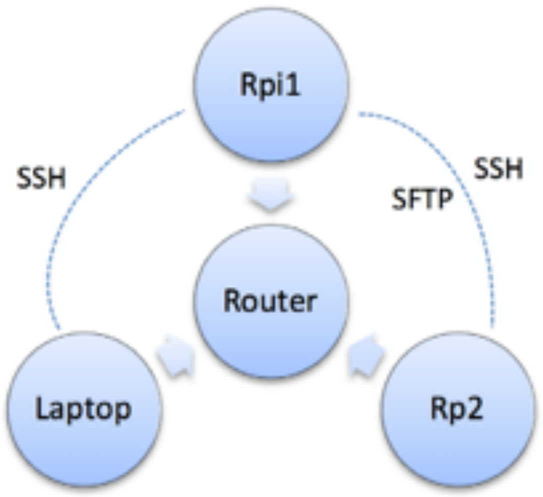


FIGURE 6: MODEL 3 FOR OVERALL SYSTEM CONNECTION

Secure shell (SSH) network protocol enables remote command line login to the raspberry pi from the laptop. Therefore the graphical desktop of the first raspberry pi can be projected on the laptop screen and the raspberry pi can be controlled remotely with laptop's keyboard and mousepad in this model. However the remote desktop access is done via a router so that the second raspberry pi can also share the same network when it is connected to the router.

3.4 Project Key Milestone

Description of Milestone	Sub-step (if any)
Setup raspberry pi and display the system on screen	Sub step #1: Booting raspberry pi & operating system installed Sub step #2: Enable display of the system on monitor/ laptop screen through Ethernet cable connection.
Enable communication between raspberry pi and external modules/ devices	Sub step #1: Enable communication between webcam, wireless network/WIFI dongle and raspberry pi.
Video recording and streaming	Sub step #1: Capture video and recording it with webcam Sub step #2: Streaming video and display on screen
Data storage on local/ external drive	Sub step #1: Store video recorded in local drive/ transmit it remotely for storage through internet
Object detection with programming algorithm	Sub step #1: Basic programming in raspberry pi Sub step #2: Program object detection algorithm
Object tracking under specific constraints	Sub step #1: Tracking specific object under certain environmental limitation
Data transmission between two raspberry pi	Sub step #1: Establish data transmission between two raspberry pi through model proposed Sub step #2: Transfer necessary data required for initializing the tracking on second raspberry pi
Tracking initialization in second raspberry pi when required data obtained from first pi	Sub step #1: Embedded data transfer protocol in tracking algorithm to transfer data while tracking the object in first raspberry pi Sub step #2: Tracking script run on second pi from startup and algorithm will only be activated when data required is received from first raspberry pi

Table 1: Key Milestone of Project

3.5 Gantt Chart

	Objectives/Tasks	May					June				July					August			
A	Literature Review & Research	1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4
1.1	V4L2 driver installation in Raspbian	■																	
1.2	Computer Vision in Python Language	■	■	■	■	■	■	■											
1.3	Object Detection & Tracking Algorithm available in SimpleCV library	■	■	■	■	■	■	■	■	■	■	■							
1.4	Communication between Raspberry Pis								■	■	■	■	■	■	■	■	■	■	■
B	Implementation																		
2.1	Programming detection & tracking algorithm in Python			■	■	■	■	■	■	■									
2.2	Experiment on Tracking Algorithm							■	■	■									
2.3	Wireless router setup for Raspberry Pi communication									■	■	■	■	■	■				
2.4	Pi communication via Ethernet using Secure Shell (SSH)														■	■	■	■	■

Table 2: Project Plan with Gantt Chart

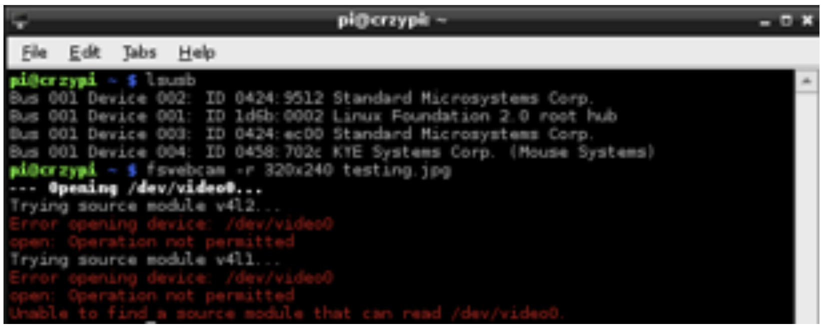
Chapter 4

Results & Discussion

4.1 Webcam compatibility issue

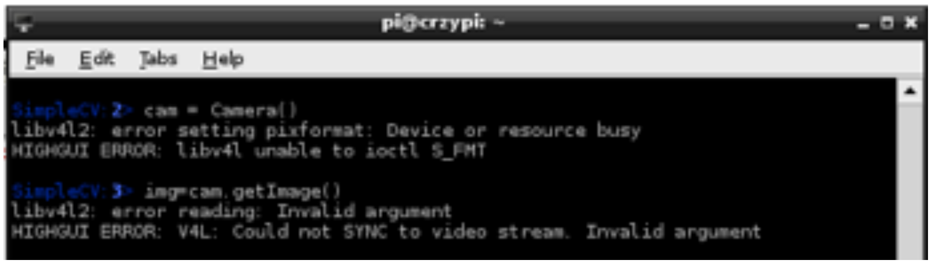
4.1.1 Compatibility in Raspbian & SimpleCV

Challenges are faced in finding compatible webcam and software configuration that works both in raspberry pi and SimpleCV framework since there are limited webcam models are workable in raspberry pi in the first place. Several webcam models are tested for its compatibility in raspberry pi and SimpleCV. Many of them are found not compatible because although their USB connection listed in the pi itself but the v4l2 (Video4Linux2) does not seem to be supporting some old webcams and hence the camera module from SimpleCV library is not able to feed the images for further image processing.



```
pi@crzypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 0458:702c KYE Systems Corp. (Mouse Systems)
pi@crzypi ~ $ fswebcam -r 320x240 testing.jpg
--- Opening /dev/video0...
Trying source module v4l2...
Error opening device: /dev/video0
open: Operation not permitted
Trying source module v4l1...
Error opening device: /dev/video0
open: Operation not permitted
Unable to find a source module that can read /dev/video0.
```

FIGURE 7: CAMERA DRIVER INCOMPATIBILITY



```
SimpleCV:2> cam = Camera()
libv4l2: error setting pixformat: Device or resource busy
HIGHGUI ERROR: libv4l unable to ioctl S_FMT

SimpleCV:3> img=cam.getImage()
libv4l2: error reading: Invalid argument
HIGHGUI ERROR: V4L: Could not SYNC to video stream. Invalid argument
```

FIGURE 8: CAMERA UNDETECTED IN SIMPLECV

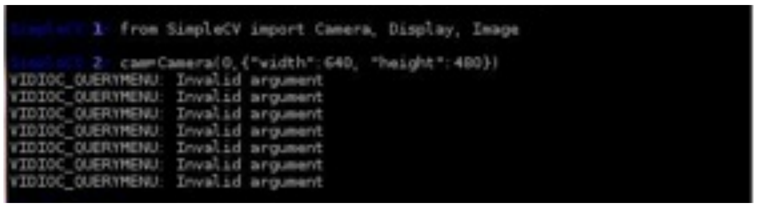
4.1.2 Models of Tested Webcam

The brand of the webcams is not the factor that governs the compatibility issue and most of the proven workable webcams are expensive and scarcely available in Malaysia market. There is also a likelihood that a webcam is compatible in Raspbian but not compatible in the SimpleCV. Nonetheless, there are two unbranded webcams, against all odds, able to feed images in SimpleCV successfully but the image resolutions are too low to be useful for capturing information needed for object tracking. Therefore they cannot be used for the purpose of this project. The several unworkable webcam models in raspbian are listed as following:

No.	Model
1	Creative M6 Webcam NX Pro
2	Genius Videocam TREK 320 R
3	V-Gear Talkcam Pro
4	Comat CPC401
5	other 2 unbranded webcams

TABLE 3: LIST OF NOT WORKING WEBCAM

Logitech c270h is the only one working in Raspbian among the tested webcams. It works for several trials in SimpleCV for capturing images but it shows error message and failed to recognize the driver in later attempts. Despite the intensive research on solving the problem, no concrete solution is found . This issue has been discussion in several linux and raspberry pi forums but only few suggestions were proposed to resolve it. The suggested approaches to resolve the issue are as follow: shell script written for uvcvideo, initiate raspberry pi updates and firmware update. The compatibility issue is not fully resolved by the solution discussed. Error message below is shown when the camera is initiated in SimpleCV but the driver can still be recognized. Therefore images can still be captured despite the error occurred.



```
SimpleCV:1: from SimpleCV import Camera, Display, Image

SimpleCV:2: cam=Camera(0,{"width":640, "height":480})
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
```

FIGURE 9: ERROR MESSAGE SHOWN IN SIMPLECV

4.2 Object Detection & Object Tracking

4.2.1 Basic Shape Tracking using Feature Detections

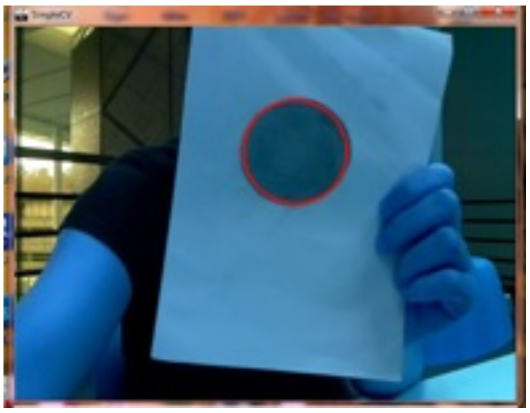


FIGURE 10: CIRCLE DETECTION

Detection and tracking for simple shape like triangle, rectangular and circle can easily be done with SimpleCV. Algorithms for feature detections like blobs, colors, lines, circles and corners are embedded in SimpleCV library. Basic shape like a circle can be detected by utilizing the basic feature detections mentioned earlier. Basic tracking algorithm as well as advanced tracking technique like CAMShift and Lucas-Kanade tracking algorithm are also available in SimpleCV library.

In this circle tracking, blobs and color feature detections are used to segregate the circle from its image background. Color Detection algorithm in SimpleCV can only effectively detect limited color from the spectrum. Colors that lies in between the "ambiguity" zone of color spectrum such as crimson, indigo and violet are hardly to be detected by this technique. Aside from this limitation, when the object color is not distinctive from its background color or any object with similar shape that has color that falls in adjacent color spectrum, the object is hardly to be detected as well.

On top of that, the ambient lighting has direct impact to the effectiveness of this tracking technique. Values in color spectrum are affected by the amount of light the surface reflected. For instance a white ball appears to be less white in a poor lighting condition. However, color detection utilize the contrast of the object's color with its surrounding background color, sometimes in poor lighting condition can actually enhance the object detection especially when the object has bright color.

4.2.2 White Ball Tracking with color and blobs detection

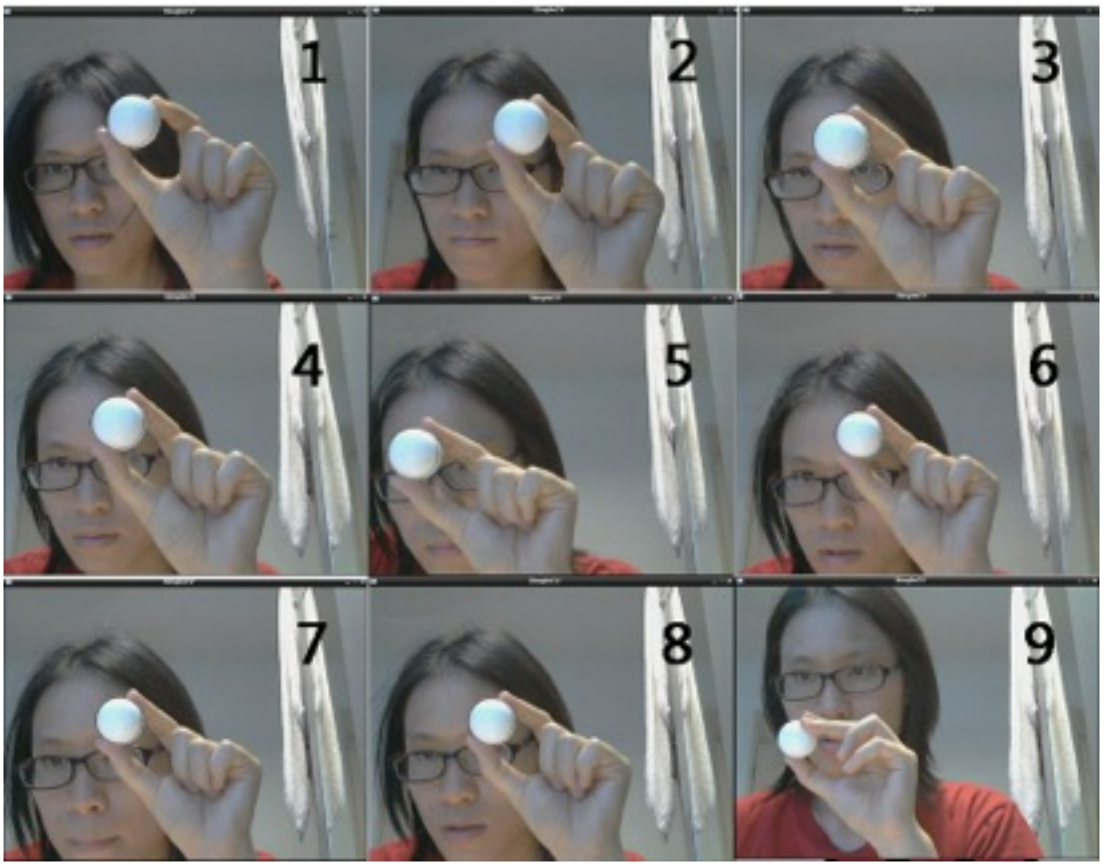
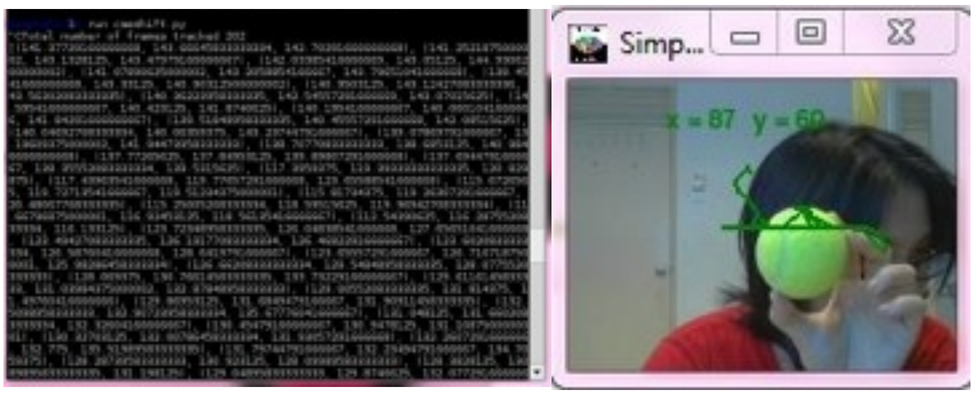


FIGURE 11: SERIES OF IMAGE FOR BALL TRACKING

Figure about shows series of image captured during tracking for a white ball. From the sequence of the images, the ball in image no. 2 and image no. 9 are not detected. The algorithm developed can only cope up with slight changes occurred within the frame. When abrupt changes occurred during the tracking, the object is unlikely to be detected and hence been tracked at the following frame. Average time taken to process each images is 5 to 8 second, thus the tracking system doesn't react in real time due to the limitation of processing power the raspberry pi possesses.

The color detection technique used in this algorithm is not applicable to track a tennis ball because of the algorithm used couldn't distinguish the tennis ball's color from its background easily. In conclusion, color detection is best used for distinctive color like black, white, bright red and blue.

4.2.3 Tennis Ball Tracking using CAMShift algorithm



```
$ run camshift.py
^Ctotal number of frames tracked 202
[illegible]
```

FIGURE 12: TENNIS BALL TRACKING RESULT USING CAMSHIFT ALGORITHM

CAMshift algorithm in SimpleCV uses histogram back projection to segment the image of the target. With this technique, histogram of the image is created and reapplied to the modified histogram on the test image to search for the object of interest. The adaptation of the algorithm in SimpleCV allow tennis ball tracking to be done easily because the color of the object has no impact to the tracking algorithm. In this case, the tennis ball is tracked over 202 frames and the movement of the tennis ball is drawn on screen to show the past position of the ball.

4.2.3 Color Spectrum & Lighting effects on algorithm

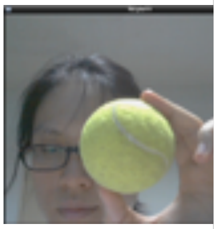
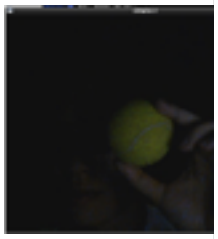
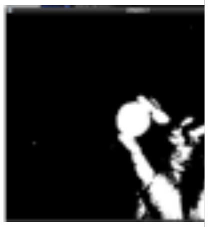
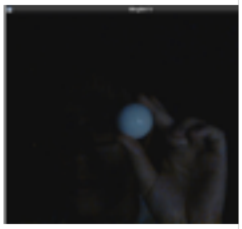
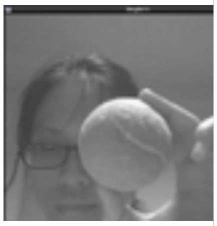
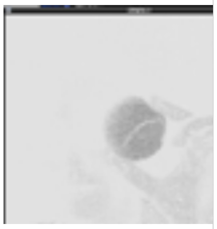
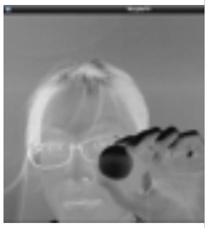
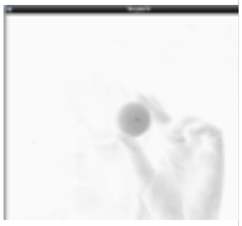
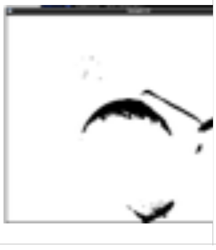
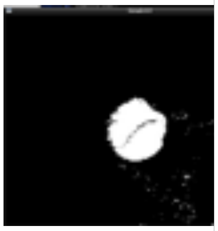
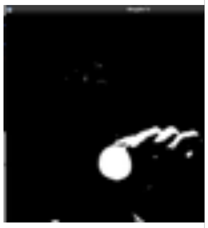
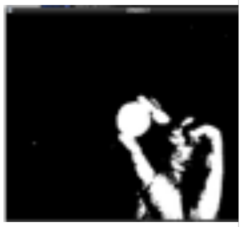
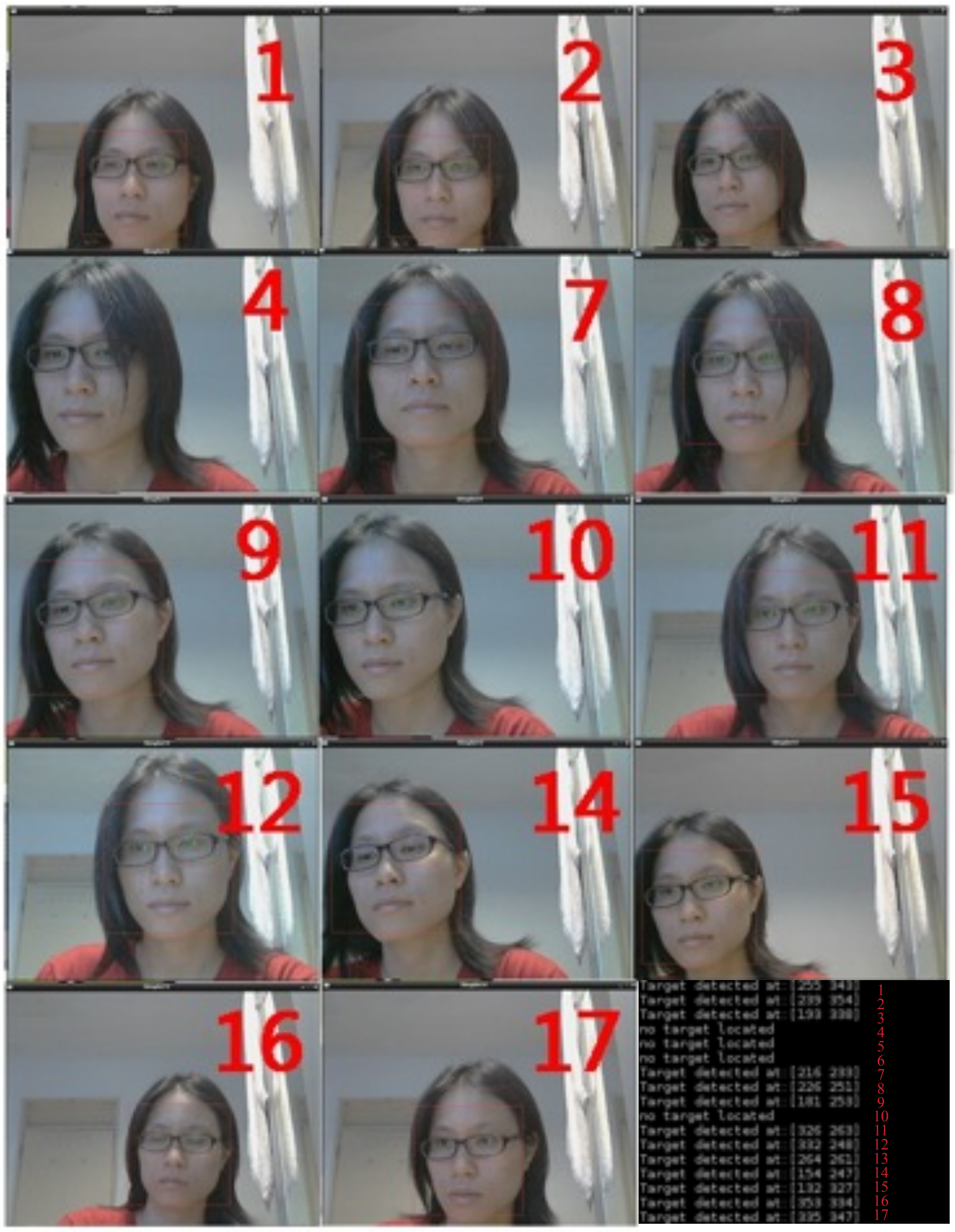
	Tennis Ball Detection		White Ball Detection	
	Light On	Light Off	Light On	Light Off
Original Image Captured				
Distinguish object color from its background				
Converting image into pure black and white				

TABLE 4 : IMPACT OF LIGHTING ON COLOR DETECTION

Color detection technique works better for tennis ball detection under poor lighting condition because its green color becomes distinguish when its background becomes dark. On the other hand, white ball detection has similar performance under both lighting condition.

All in all, color detection technique is only suitable for simple object detection. When an object has complex features with no distinguish color, this technique is no longer effective in detecting the object.

4.2.4 Face Detection & Tracking



```
Target detected at:[255 343]   1
Target detected at:[239 354]   2
Target detected at:[193 338]   3
no target located              4
no target located              5
no target located              6
Target detected at:[216 233]   7
Target detected at:[226 251]   8
Target detected at:[181 259]   9
no target located              10
Target detected at:[326 263]   11
Target detected at:[332 248]   12
Target detected at:[264 261]   13
Target detected at:[154 247]   14
Target detected at:[132 327]   15
Target detected at:[353 334]   16
Target detected at:[335 347]   17
```

FIGURE 13: SERIES OF IMAGE CAPTURED DURING FACE TRACKING

Results printed on terminal are showed at the bottom right picture. The sequence of frames associated with each print on terminal are labelled according. The processing time taken by the raspberry pi is longer than tracking a ball due to the significant amount of details and complexity needed to be processed in the raspberry pi. Average time for each frame to be processed and display the outcome on screen is 8-12 seconds. From the results shown above, target is lost tracked in frame no. 4, 5, 6 and 10. From frame no. 3 to frame no. 4, the scale of the object is changed (from far to near). Abrupt change in consecutive frames reduced the likelihood of the object to be tracked due to the limited tracking ability the algorithm developed in this project.

Apart from the change in scale of object, the targeted face can only be detected when the person is facing the camera without tilting his or her head. In this project, face detection algorithm is derived from the Haar-like features algorithm that is developed by Intel. The classifier trained in the algorithm detect facial features such as eyes, nose and mouth in the frontal faces. Therefore a forward looking face is easier to be detected compared to a face looking to the side.

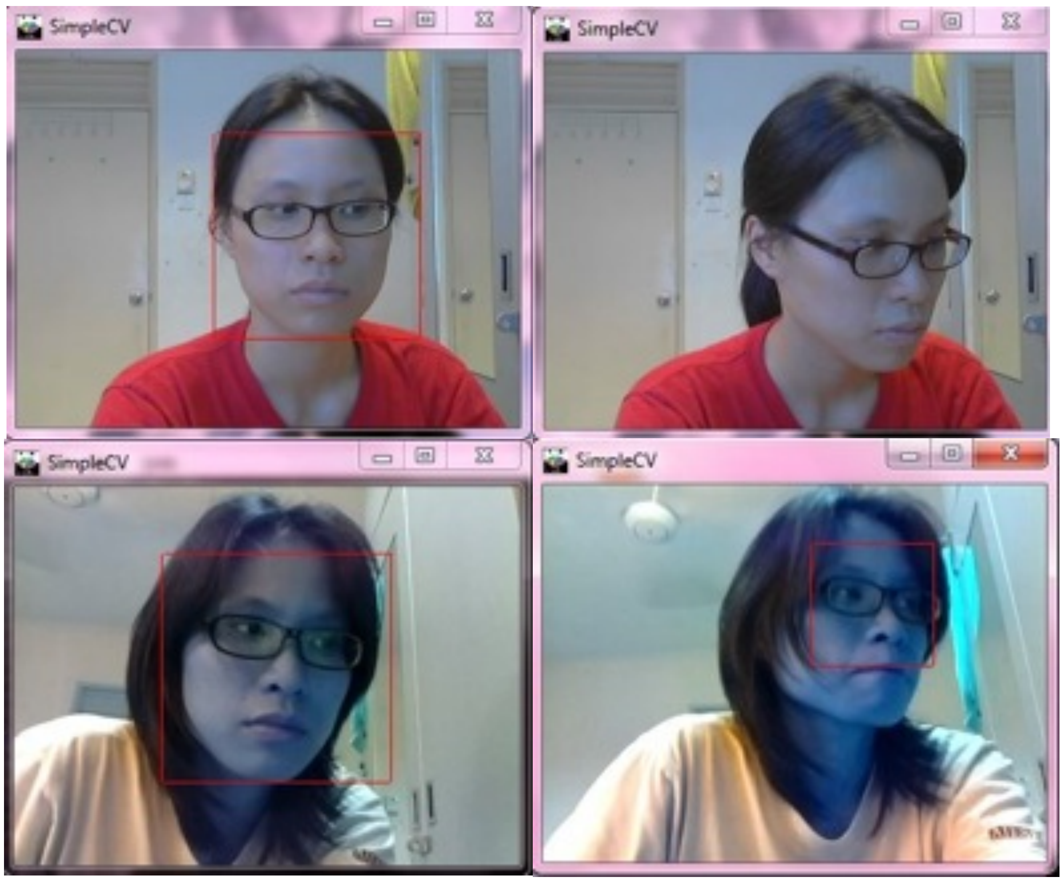


FIGURE 14: FACE DETECTION SENSITIVITY IN VARIOUS HEAD ANGLE

4.2.5 Face Recognition

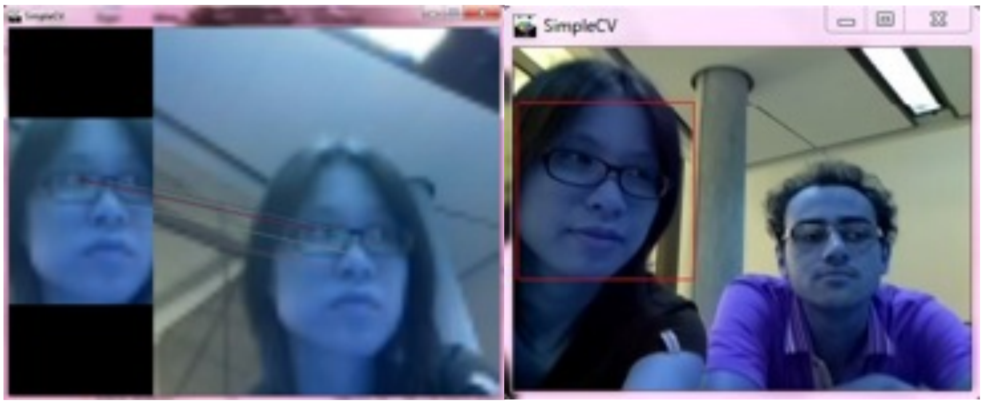


FIGURE 15: FACE RECOGNITION USING KEY POINTS MATCHING TECHNIQUE

An effective face recognition algorithm is very complicate and huge complex data needed to be trained and stored inside the system. With limited processing power of raspberry pi, face recognition algorithm is developed using simple key points matching technique. A sample image of subject of interest need to be loaded beforehand so that the distinguish of the features points can be identified and could be used to compared against with the images captured by the video stream.

Once faces are detected using the Haar-like feature facial detection approach mentioned earlier, the key points that describe the uniqueness of a face are extracted and compared against with the template stored beforehand. This method of recognition is effective and robust compared to feature detection technique discussed earlier. This is because the algorithm doesn't affected by the color features of the subject but sufficient light is needed for decent quality of images to be captured. Nonetheless this algorithm only offers decent performance in recognizing the targeted person due to its simplicity.

When the face detection, key points matching for face recognition and basic tracking algorithms are combined, the system can be used to track a targeted person over time. However the time taken for processing has seen to be increased when the system tracks in multiple faces environment. The limited processing power of raspberry pi affects the overall performance of tracking in terms of speed and accuracy.

4.3 Communication between two raspberry pis

4.3.1 Wireless communication

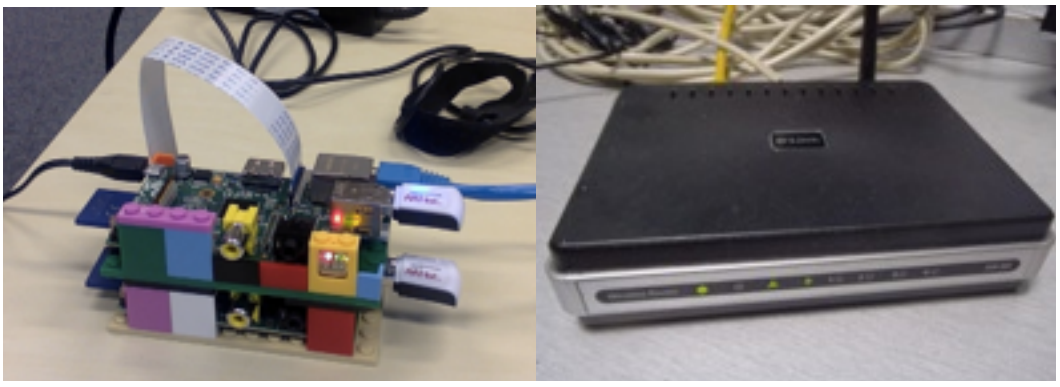
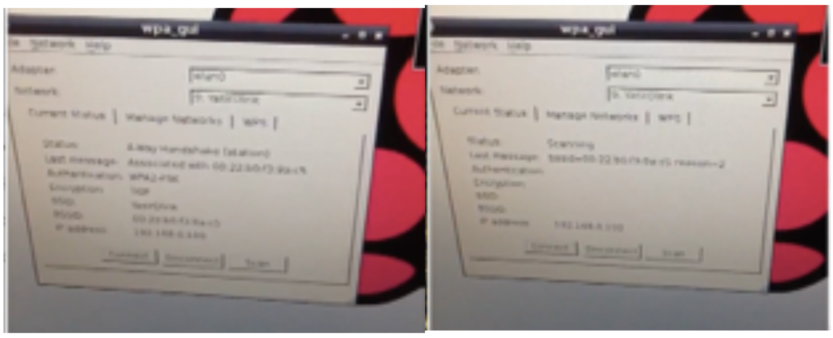


FIGURE 16: RASPBERRY PI TO PI COMMUNICATION VIA WIRELESS ROUTER

In order for the two raspberry pis to communicate to each other directly without using any add-on hardware (eg. transmitter and receiver), a wireless ad-hoc network is to be established and shared between the raspberry pis via a wireless router in order for them to forward data to one and another.

The raspberry pis are able to connect wirelessly to the WLAN provided by the university using a wifi dongle. However, the attempt to connect the two raspberry pis to a wireless router (D-Link DIR-300) and a Celcom broadband is failed. No internet signal is detected by the wifi dongle when it attempted to connect to the broadband. On the other hand, the communication channel is established and dropped consecutively in several seconds until the system hangs. The root cause of the error happened has not been found. It is suspected that this is due to the hardware configuration misalignment between raspberry pi and the router since the dongle can connect to WLAN in the university.



```
wpa_gui
Adapter:          wlan0
Network:          [illegible]
Current Status | Manage Networks | WPS
Status:           4-Way Handshake (station)
Last message:     Associated with [illegible]
Authentication:   WPA2-PSK
Encryption:       [illegible]
SSID:             [illegible]
BSSID:            [illegible]
IP address:       [illegible]
```

```
wpa_gui
Adapter:          wlan0
Network:          [illegible]
Current Status | Manage Networks | WPS
Status:           Scanning
Last message:     [illegible]
Authentication:
Encryption:
SSID:
BSSID:
IP address:       [illegible]
```

FIGURE 17: INTERNET MESSAGE SHOWN IN ROUTER CONNECTION ATTEMPT

4.3.2 Wired Communication through Ethernet

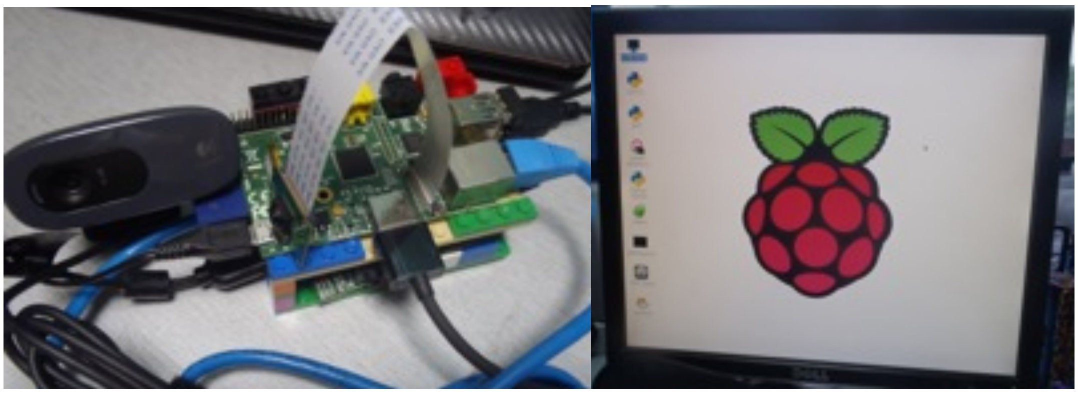
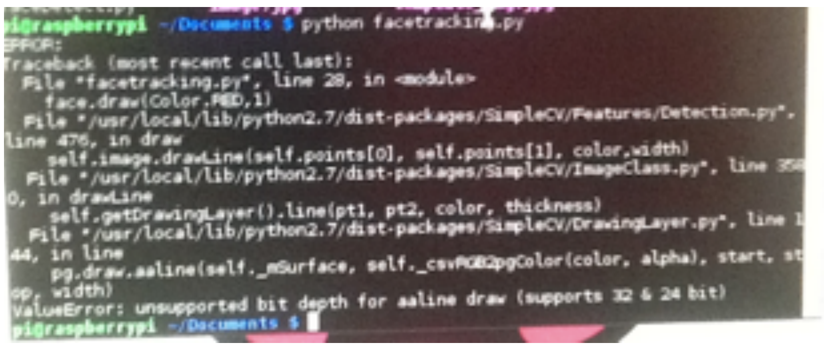


FIGURE 18: COMMUNICATION THROUGH ETHERNET WITH HDMI DISPLAY

The first attempt for wired communication between two raspberry pis is established by direct ethernet connection with the first pi connected to HDMI display (with keyboard and mouse connected to first pi). Raspberry Pi camera module is used instead of another webcam in order to minimize the power usage of the first pi. In the meanwhile the second pi is connected to webcam for tracking. Communication channel is establish via SSH network protocol. The data is transferred between two raspberry pis through SFTP. However, the tracking cannot be done with the first pi even though the tracking algorithm works well when the raspberry pi desktop is displayed using virtual network computing (VNC) server with the laptop. It is suspected that the power is insufficient for the raspberry pi to accomplish the image processing while connecting to HDMI display.


```
i@raspberrypi ~/Documents $ python facetracking.py
ERROR:
raceback (most recent call last):
  File "facetracking.py", line 28, in <module>
    face.draw(Color.RED,1)
  File "/usr/local/lib/python2.7/dist-packages/SimpleCV/Features/Detection.py",
line 476, in draw
    self.image.drawLine(self.points[0], self.points[1], color,width)
  File "/usr/local/lib/python2.7/dist-packages/SimpleCV/ImageClass.py", line 35
0, in drawLine
    self.getDrawingLayer().line(pt1, pt2, color, thickness)
  File "/usr/local/lib/python2.7/dist-packages/SimpleCV/DrawingLayer.py", line 1
44, in line
    pg.draw.aaline(self._mSurface, self._csvRGB2pgColor(color, alpha), start, st
op, width)
ValueError: unsupported bit depth for aaline draw (supports 32 & 24 bit)
pi@raspberrypi ~/Documents $
```
FIGURE 19: ERROR MESSAGE SHOW WHEN TRACKING DONE WITH HDMI DISPLAY

4.3.3 Wired Communication via router

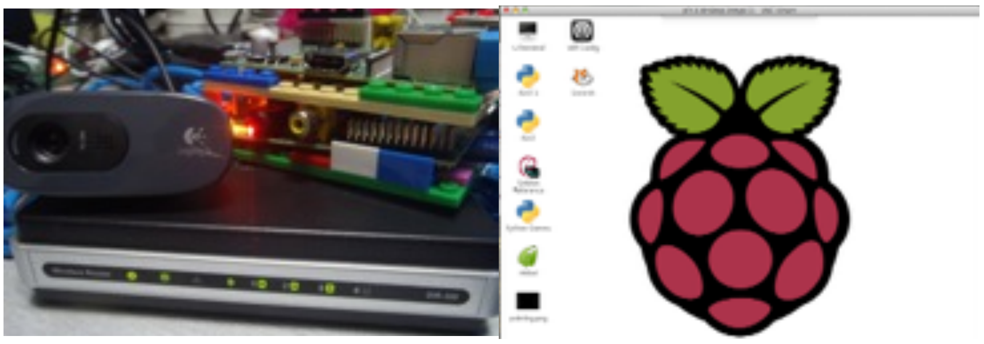


FIGURE 20: COMMUNICATION VIA ROUTER WITH VNC REMOTE DESKTOP DISPLAY

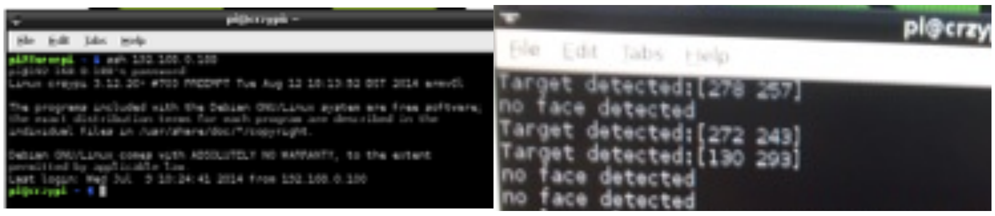
Since the tracking cannot be done with raspberry pi connected to HDMI display, a workaround approach is used to resolve this issue. The two raspberry pis and a laptop are connected to a router through ethernet respectively. SSH and VNC server are used to display and control the first raspberry pi remotely. On the other hand, the communication channel between the two pi is established using SSH via the router.

However since the raspberry pi system doesn't support graphical remote desktop display like Windows and OS X. When the first pi is connected to the second via SSH, the desktop of the second pi cannot be shown in the first pi. Only the terminal of the second pi is displayed within the terminal of the first pi. Therefore the display of tracking outcome from the second pi can only be done on the terminal with coordinate indication. Data points collected from the second pi is compiled in a text file and sent back to the first pi for further processing usage.

On top of that, the SSH communication between the laptop and the first pi via a router is not stable as compared to the SSH communication with the ethernet ports connected directly without passing through a router. The communication channel is connected up for half and hour but it is likely to be disconnected afterward. It is found that the router needed to be reset from time to time so that the communication channel can be stabilized for communication.

Aside from the communication issue, the tracking performance of the first pi is deteriorated when automated SFTP is initiated within the tracking script. Time taken for the raspberry pi to process and transfer the file is prolonged by at least 10s compared to the overall performance before. Nonetheless, image captured by the first pi before the subject leaves the webcam frame can successfully passed to the second pi to initialize the second pi to start tracking on the target.

The tracking script on the second pi is run from startup but the tracking algorithm will only be initiated when the data needed for tracking is transferred across from the first pi. Once the tracking is initiated on the second pi, the tracking result is printed on the terminal to display the coordinate of the target when it is detected.



```
pi@raspberrypi ~ $ ssh 192.168.0.100
pi@192.168.0.100's password:
Linux crzypi 3.12.20+ #700 PREEMPT Tue Aug 12 18:13:52 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jul  9 10:24:41 2014 from 192.168.0.100
pi@crzypi ~ $
```

```
Target detected:[278 257]
no face detected
Target detected:[272 243]
Target detected:[130 293]
no face detected
no face detected
```

FIGURE 21: SSH COMMUNICATION & TRACKING RESULT ON SECOND RASPBERRY PI

Chapter 5

Recommendation & Conclusion

By today computing technology standard, the CPU of Raspberry Pi has comparable slow processing speed and low computing performance. Since we have limited computational power, the solution proposed is to have task sharing or data distribution among multiple Raspberry Pis so that the amount of data needed to be stored in the system can be significantly reduced.

The v4l2 developed by the Linux community is also not compatible to most ordinary webcams. Control of buffer content or data flow from these webcams is a challenge faced in using Raspberry Pi for this project. Although the camera module released by the Raspberry Pi has no compatibility issue but it is not suitable to be used in SimpleCV because in SimpleCV framework only USB devices are recognized.

SimpleCV provides limited computer vision library for its user. Nonetheless using these library to perform object tracking in Raspberry Pi offers decent performance. This system can be further optimized and upgraded using OpenCV to gain more comprehensive computer vision library. Advanced detection and tracking techniques can be used in OpenCV to boost the tracking performance in overall.

As conclusion, the possibility of implementing object tracking using low cost embedded system hardware has been proven with this project. Even though object tracking in real time is difficult tasks for one Raspberry Pi to handle, data sharing or task delegation is a good way in resolving those limitations.

References

[1] A. Yilmaz, O. Javed and M. Shah, "Object tracking: A survey," *ACM Computing Surveys,* vol. 38, no. 4, 2006.

[2] N. Kehtarnavaz and M. Gamadia, Real-Time Image and Video Processing From Research to Reality, Morgan & Claypool, 2006 .

[3] S. Liu, A. Papakonstantinou, H. Wang and D. Chen, "Real-Time Object Tracking System on FPGAs," in *Application Accelerators in High-Performance Computing (SAAHPC)*, July 2011.

[4] M. Bramberger, J. Brunner, B. Rinner and H. Schwabach, "Real-Time Video Analysis on an Embedded Smart Camera," in *Real-Time and Embedded Technology and Applications Symposium*, May 2004.

[5] M. Wei and A. Bigdeli, "Implementation of a Real-Time Automated Face Recognition System," in *IEEE International Symposium on Communications and Information Technologies*, October 2004.

[6] N. N. Hoang and J. L. Kai, "Small Embedded Vision System Using AVR-8 bit Microcontroller Atmega64 and OV6620 CMOS Image Sensor," in *Proceedings of 2007 CACS International Automatic Control Conference* , National Chung Hsing University, Taichung, Taiwan, November, 2007.

[7] K. Arshak and F. Adepoju, "Object tracking in the GI tract: A novel microcontroller approach," in *Advanced Motion Control, 2006. 9th IEEE International Workshop*, Istanbul, Turkey, 2006.

[8] L. Xiaofeng, R. Diqi and Y. Songyu, "FPGA-based Real-Time Object Tracking for Mobile Robot," in *Audio Language and Image Processing (ICALIP), International Conference*, Shanghai, November 2010.

[9] Z. Elisa, "Real-time markerless tracking of objects on mobile devices," November 2009.

[10] Ladvien, "OpenCV on Raspberry Pi," 24 April 2013. [Online]. Available: http://letsmakerobots.com/node/36947. [Accessed 10 February 2014].

[11] G. A. Stafford, "Object Tracking on the Raspberry Pi with C++, OpenCV, and cvBlob," 9 February 2013. [Online]. Available: http://programmaticponderings.wordpress.com/2013/02/09/opencv-and-cvblob-with-raspberry-pi/. [Accessed 13 February 2014].

[12] D. Graziano, "First boot with the Raspberry Pi," 13 February 2014. [Online]. Available: http://howto.cnet.com/8301-11310_39-57618810-285/first-boot-with-the-raspberry-pi/. [Accessed 18 February 2014].

[13] Raspberry Pi Foundation, “MODEL B” [Online]. Available: http://www.raspberrypi.org/product/model-b/. [Accessed 10 April 2014]

[14] P. Abrahamsson et al., “Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Rasperry Pi Cloud Cluster Experiment,” in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference*, Bristol, December 2013

[15] J. Lamer, D. Cymbalak and F. Jakab, “Computer vision based object recognition principles in education,” in *Emerging eLearning Technologies and Applications (ICETA), 2013 IEEE 11th International Conference*, Stara Lesna, October 2013. pp. 253-257.

[16] Computer Vision Algorithm Implementations,” [Online]. Available: http://www.cvpapers.com/rr.html. [Accessed 10 April 2014]

[17] K. C. Yong, A. S. Prabuwono and R. Sulaiman, “Visitor face tracking system using OpenCV library,” in *Research and Development (SCOReD), 2009 IEEE Student Conference*, UPM Serdang, November 2009. pp.196-199

[18] R. S. Deepthi and S. Sankaraiah, “Implementation of Mobile Platform using QT and OpenCV for Image Processing Applications,” in *Open Systems (ICOS), 2011 IEEE Conference*, Langkawi, September 2011. pp.284-289

[19] Y. X. Wu, “Research on bank intelligent video image processing and monitoring control system based on OpenCV, ” in *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference*, August 2009. pp. 211-214

[20] D.G. Calin, “How to Detect and Track Object Using Matlab,” 9 October 2013. [Online]. Available: http://www.intorobotics.com/how-to-detect-and-track-objects-using-matlab/. [Accessed 12 April 2014]

[21] S. Matuska, R. Hudec and M. Benco, “The Comparison of CPU Time Consumption for Image Processing Algorithm in Matlab and OpenCV,” in *Proceedings on 9th International Conference 2012 ELEKTRO*, Rajeck Teplice, May 2012. pp75-79

[22] "OpenCV vs. Matlab vs. SimpleCV," [Online]. Available: http://simplecv.tumblr.com/post/19307835766/opencv-vs-matlab-vs-simplecv. [Accessed 8 April 2014].