



澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Outstanding Academic Papers by Students

學生優秀作品



University of Macau

Faculty of Science and Technology



澳門大學

UNIVERSIDADE DE MACAU

UNIVERSITY OF MACAU

iFinger – Study of Gesture Recognition Technologies & Its Applications

Volume II of II

by

Chi Ian, Choi, Student No: DB02828

Final Project Report submitted in partial fulfillment
of the requirements of the Degree of
Bachelor of Science in Software Engineering

Project Supervisor

Dr. Fai Wong, Derek
Dr. Sam Chao, Lidia

08 October 2014

DECLARATION

I sincerely declare that:

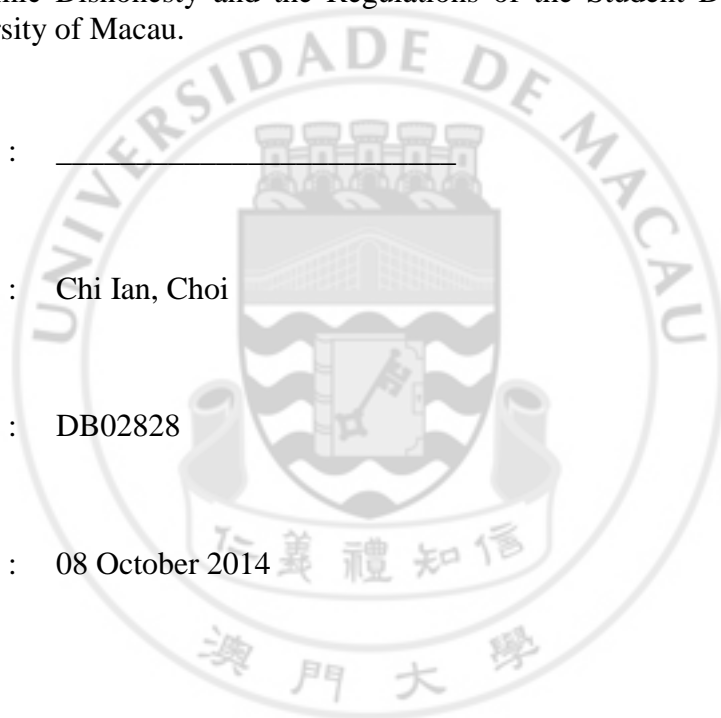
1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : _____

Name : Chi Ian, Choi

Student No. : DB02828

Date : 08 October 2014



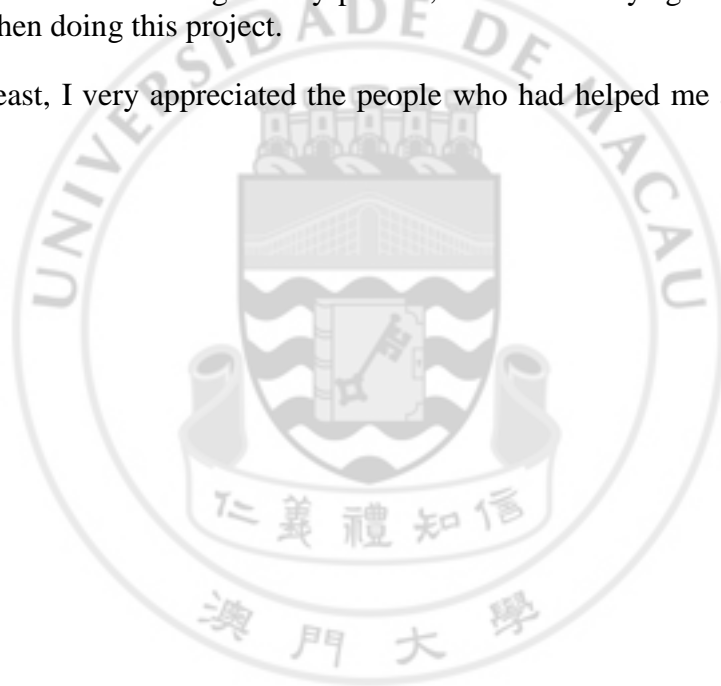
ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to UM for providing the opportunity to carry out a project as a partial fulfillment of the requirement for the degree of Bachelor of Software Engineering.

Throughout this project, I am very fortunate to receive the guidance and encouragement from my supervisor, Derek Wong and Lidia Chao. They let me learnt the procedure for doing research, methodology analyzing, design, implement, etc.

Secondly, I am deeply grateful for all doctors, professors and assistants who taught or helped me in the university life. I cannot finish this report without learnt those knowledge. Furthermore, the little achievement of the final year project is not only belonging to me. It also belongs to my partner, Ben. He always gives some creative suggestions when doing this project.

Last but not least, I very appreciated the people who had helped me at some time in the past.



ABSTRACT

Because of the development of Human–computer interaction (HCI), the method of interaction with the computer is becoming more and more freedom, which seek to use the human body to control computer naturally. However, the more natural action will introduce more ambiguity and require a more sensitive system.

Therefore, we develop a system which can extract the hand features from images using a common digital camera. So that we can calculate the human hands shape, motion, moving direction, moving speed, etc. In addition, we added some anti-shaking algorithm to stable the result. Hence, it can control computer to do some tasks only with human hands and show the result in real time.

We are adding some optimization and smoothing algorithms into the system in order to increase stability and sensitivity. Hence, users can interact with computer more intuitively and directly.

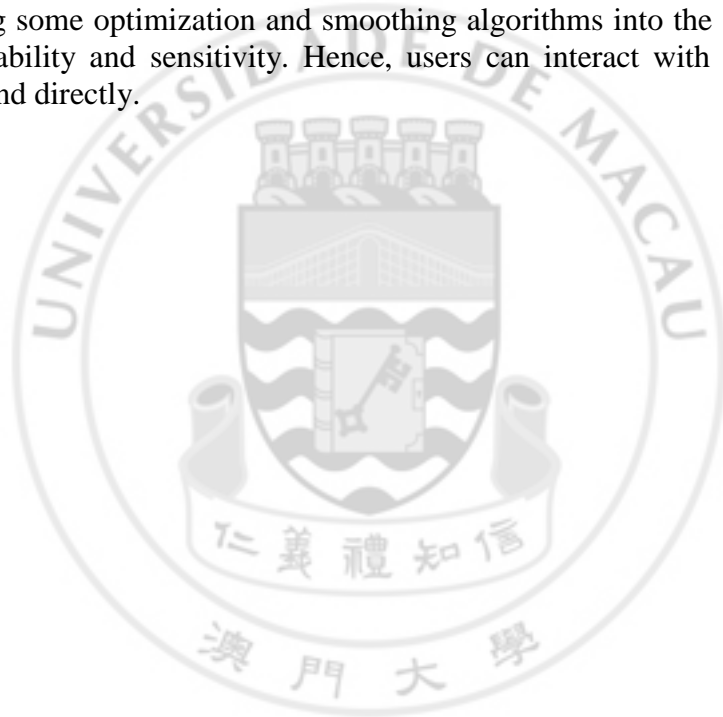


TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION	12
1.1 Overview	12
1.1.1 HCI History	12
1.1.2 Our Future Life	12
1.2 Gesture Recognition Devices	13
1.2.1 Kinect	13
1.2.2 Leap Motion	13
1.2.3 MYO Armband	14
1.2.4 Overall	14
1.3 Objectives	15
1.3.1 System Objectives and Motivation	15
1.3.2 System Environment	15
1.3.3 Devices Setup	16
1.4 Summary of Workload	17
1.4.1 Project Scope	17
1.4.2 Development Process	17
1.4.3 Work Distribution	20
1.5 Technologies Description	21
1.5.1 Difficult During Development	22
CHAPTER 2. RELATED WORK	23
2.1 Collaboration with a Robotic Scrub Nurse	23
2.1.1 Background	23
2.1.2 Gesture Recognition	23
2.1.3 Experiment Result	24
2.2 Turns Paper into a Touchscreen	25
2.2.1 Gesture Recognition	25
2.2.2 Other Functions	26
2.3 Gesture recognition for digits and characters	26
2.3.1 Background	26
2.3.2 Hand Detection	26
2.3.3 Fingertips Detection	26
2.3.4 Gesture Recognition	27
2.3.5 Experiment Result	27
2.4 A Two-Hand Multi-Point Gesture Recognition System Based on Adaptive Skin Color Model	27
2.4.1 Object Detection	27
2.4.2 Object Segmentation and Tracking	27
2.4.3 Features Extraction	27
2.4.4 Gesture Recognition	28
2.4.5 Experiment Result	28
CHAPTER 3. DESIGN	29

3.1 Overall System Design	29
3.2 Gesture Recognition	30
3.2.1 Separate Two Hands	30
3.2.2 Six Basic Gestures	31
3.2.3 Simulate Mouse Click	33
3.2.4 Simulate Keyboard Shortcuts	33
3.2.5 Virtual Keyboard	33
3.2.6 Unexpected Reaction	34
3.2.7 Recognize the Dynamic Gesture	34
3.3 History Move	35
3.4 Interface	36
3.4.1 Open the Start-up Screen	36
3.4.2 Main Interface	37
3.4.3 Setting YCbCr	37
3.4.4 Start Program	38
3.4.5 Working Window	38
3.4.6 Button Background Problem[14]	40
3.4.7 Game	41
3.4.8 Browser	42
3.4.9 Open File	44
3.4.10 Video Player	47
3.4.11 Help	49
3.4.12 About Us	49
3.4.13 Exit the Program	49
CHAPTER 4. IMPLEMENTATION	50
4.1 Two-Hand Separation	51
4.2 Gesture Recognition	52
4.3 Optimization	53
4.3.1 Transparent the Working Window	53
4.3.2 Working Window Auto Moving	54
4.4 Control Computer	55
4.4.1 Open Application	56
4.4.2 Open On-Screen Keyboard	56
4.5 Interface of iFinger	57
4.6 Setting	58
CHAPTER 5. TESTING AND EVALUATION	60
5.1 Normal Environments	60
5.2 Special Environment	60
5.2.1 Very Strong Light Source	60
5.2.2 Complex Background	61
5.3 Processing Time Testing	61

5.4 Testing Gestures	62
5.4.1 Testing Accuracy	62
5.5 Evaluation	64
 CHAPTER 6. DISCUSSION	 65
6.1 Satisfied Objectives	65
6.2 Unsatisfied Objectives	65
6.3 Future Work	66
 CHAPTER 7. CONCLUSIONS	 67
7.1 Overall	67
7.2 Acquisition	67
 CHAPTER 8. REFERENCES	 68



LIST OF FIGURES

Figure 1: Sci-fi movie - Iron man 2	12
Figure 2: Gesture recognition devices	13
Figure 3: Leap Motion valid area.....	13
Figure 4: MYO Armband.....	14
Figure 5: Camera position and valid area to be captured.....	16
Figure 6: Microsoft LifeCam	16
Figure 7: Work distribution	20
Figure 8: Robotic Scrub Nurse	23
Figure 9: Gesture table for surgery	24
Figure 10: Testing result for ACM research	24
Figure 11: Using paper as touchscreen	25
Figure 12: Height information for fingertip.....	25
Figure 13: Mask model	26
Figure 14: Radar scan	28
Figure 15: System flowchart.....	29
Figure 16: Before separate two hands.....	30
Figure 17: After separate two hands	30
Figure 18: On-Screen keyboard	34
Figure 19: Direction tracking.....	35
Figure 20: History path analysis	35
Figure 21: Interface flowchart	36
Figure 22: Start-up screen.....	36
Figure 23: Main interface.....	37
Figure 24: Setting window.....	37

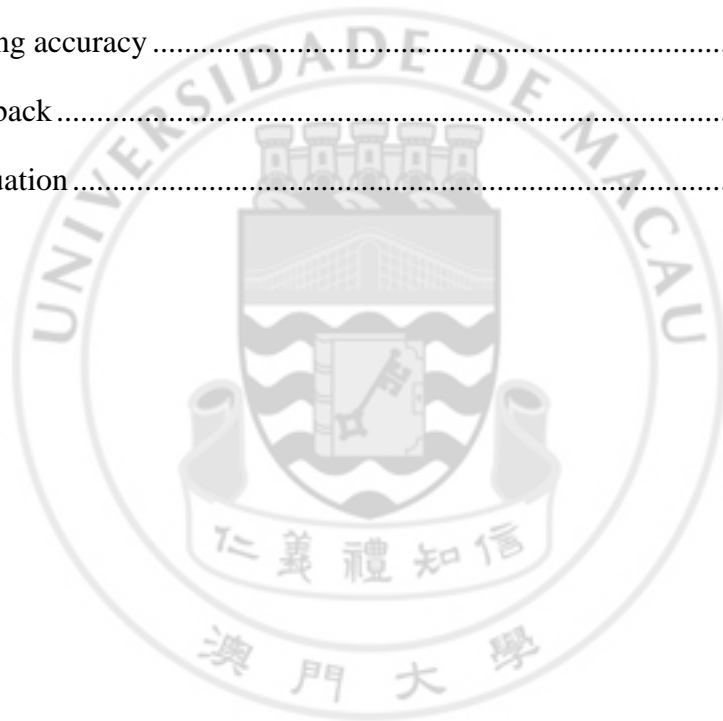
Figure 25: working window	38
Figure 26: Transparent the working window	38
Figure 27: Control mouse area.....	39
Figure 28: Moving the working window	39
Figure 29: Main interface.....	40
Figure 30: Cut the Rope.....	41
Figure 31 : Sliding in Game.....	41
Figure 32: Google Chrome	42
Figure 33: Original tab.....	43
Figure 34: Previous tab (upper) Next tab (lower)	43
Figure 35: Original page	43
Figure 36: Previous page (upper) Next page (lower).....	43
Figure 37: Open a new tab.....	43
Figure 38: Adobe Reader	44
Figure 39: Original reading page	45
Figure 40: Page down (upper) Page up (lower)	45
Figure 41: Original Size.....	46
Figure 42: Zoom in (upper) Zoom out (lower)	46
Figure 43: Windows Media Player	47
Figure 44: Stop video.....	48
Figure 45: Play video	48
Figure 46: Original volume.....	48
Figure 47: Decrease volume (upper) Crease volume (lower)	48
Figure 48: Help	49
Figure 49: About us	49
Figure 50: System flowchart.....	50
Figure 51: Two-hand separation flowchart.....	51

Figure 52: Gestures recognition flowchart	52
Figure 53: Working window auto moving flowchart	54
Figure 54: Control computer flowchart	55
Figure 55: Normal environments	60
Figure 56: Strong light source.....	61
Figure 57: Complex background	61



LIST OF TABLES

Table 1: Basic gestures	31
Table 2: Gesture table	33
Table 3: Main gestures in Game	41
Table 4 : Main gestures in Google Chrome	42
Table 5: Main gestures in Adobe Reader.....	44
Table 6: Main gestures in Window Media Player[15].....	47
Table 7: Testing accuracy	62
Table 8: Feedback.....	63
Table 9: Evaluation.....	64



CHAPTER 1. INTRODUCTION

1.1 Overview

HCI (Human-Computer Interface) [1] is one of hot topics in computer science. People always want to interact with machine by using the most natural way, such as: body gesture.

1.1.1 HCI History

In 1963, the first pointing device was released, which was using a light-pen to control the virtual object, including grabbing objects, moving them, changing size, and using constraints. The mouse was developed at Stanford Research Laboratory in 1965, which was the replacement for light-pen. Finally, mouse becomes famous in the 1970's. [2] The earliest gesture recognition tool always used sensor-based device as the input device. It is accurate but it has to pay higher cost and not comfortable to user.

1.1.2 Our Future Life

Nowadays, gesture is using in our daily life everywhere, which is the most natural way for communication between people. The Sci-fi movie affects people's expectation, where the computer can be controlled without wearing any sensor-based device but the human gesture in a natural way.



Figure 1: Sci-fi movie - Iron man 2

It seems amazing to interact with the computer in this way, but it requires many different technologies underneath. However, we will advance the gesture recognition technique with an aim to realize this kind of interactive technology.

1.2 Gesture Recognition Devices

There have many devices for gesture recognition in the market, such as: Kinect [3], Leap motion [4], and MYO Armband [5].



Figure 2: Gesture recognition devices

1.2.1 Kinect

Actually, Kinect is very successful tool in the area of body motion recognition. It can detect the skeleton of human body and its movement, by extracting the human object from the background. Indeed, the Kinect can provide deep information which is important data for different purposes. [3]

However, the stability is not good to detect a tiny movement, such as finger movement. [6] Even the body structure is shaking all the time. Furthermore, its price is quite expensive.

1.2.2 Leap Motion

It is a new tool for gesture recognition, which is on sale on 19, May 2013. Leap motion is a kind of box device which contains two digital cameras. The images will be transferred to the computer though a USB link and produce result by those images. The valid area for this device is illustrated in the following figure. [4]

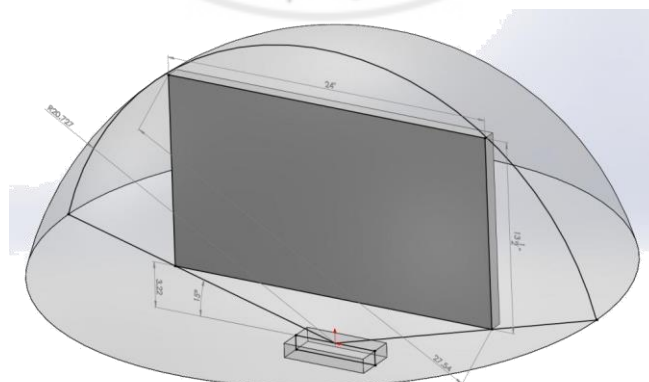


Figure 3: Leap Motion valid area

Based on the difference of two images, it will not consider the object exist if different is too less. Therefore, it will miss-track the hands when the position of hands higher than the maximum height. [7] Furthermore, this device needs an expensive computation powers to process the two images for identifying the hand gesture.

1.2.3 MYO Armband

MYO Armband [5] is another new device for gesture recognition, which expects to be sold in the spring of the year 2014. It uses a band to collect the reaction of the muscle, and converts it into a digital signal. Finally, transfer the signal to computer for recognizing the possible gesture.

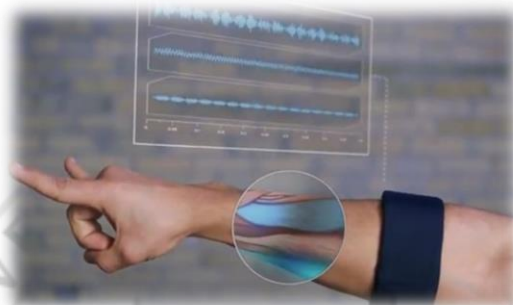


Figure 4: MYO Armband

However, the latest news for the MYO band is that it may reduce the number of gestures compare to the previous proposal. Actually, this technique is new but also hard to control. It is because human may do some actions that produces similar muscle signals, and as a result. It will lead to an unexpected control.

1.2.4 Overall

We can find some products for the gesture recognition. However, those products rely on some kind of specific design devices whose prices are normally very expensive. This makes it unsuitable be widely used. This project tries to find a solution way to achieve the similar tasks, but with a much lower and affordable cost.

1.3 Objectives

There are many products for the gesture recognition, but gesture control still hasn't widely applied. The main reason may cause by the cost of the device. Our aim is to develop a system to analyze the user gesture and apply to everywhere in an easy way instead of buying specifies hardware.

However, when we look into the other researches from the internet, we found that most of them using the gesture to control computer directly, or using the palm center to control the mouse. Also the quality of stability is poor, which the finger is shaking seriously. Therefore, we want to solve those problems by our effort.

1.3.1 System Objectives and Motivation

The cheapest way to perform gesture recognition is using the common camera with normal resolution. The system can process the image which captured by the camera, the processing time should be in real-time so that the user can get the result similar to using the mouse and keyboard. Furthermore, the system should include not only the normal activity of mouse, but also the keyboard activity, such as mouse click, mouse move, input character, and etc.

It is impossible to do many operations using one hand gesture only. The system should provide two hands gesture recognition for user to use. Besides two hands operation, the detection of moving gesture is important also. The average recognition accuracy of all kinds of gesture should larger than 90%.

In order to process the image effectively, we are using the OpenCV software package, which provide many algorithms for image processing. It also can minimize our programming effort so that we do not need to implement all the fundamentals from scratch, but base on it to further develop the recognition algorithms.

The interface, which is the main tool that interacts with the user. It should be user-friendly so that the user can understand how to manage the system without any doubt. We will collect the feedback from the users to evaluate the interface quality.

1.3.2 System Environment

There are two subprograms in our system, and both of them implemented in Microsoft Visual Studio 2010. One of them is developed for gesture recognition. Its programming language is C++ with OpenCV library. Another subprogram is implemented for the application interface. We use MFC as the programming language.

The program is executable in Windows 7 platform with Service Pack 1.

1.3.3 Devices Setup

We need to set up a camera in front of the computer, so that we can control the computer through this camera. The camera needs to shoot at a range of area for detecting the hand motion, which means the activity of user's hands should be within the valid area.

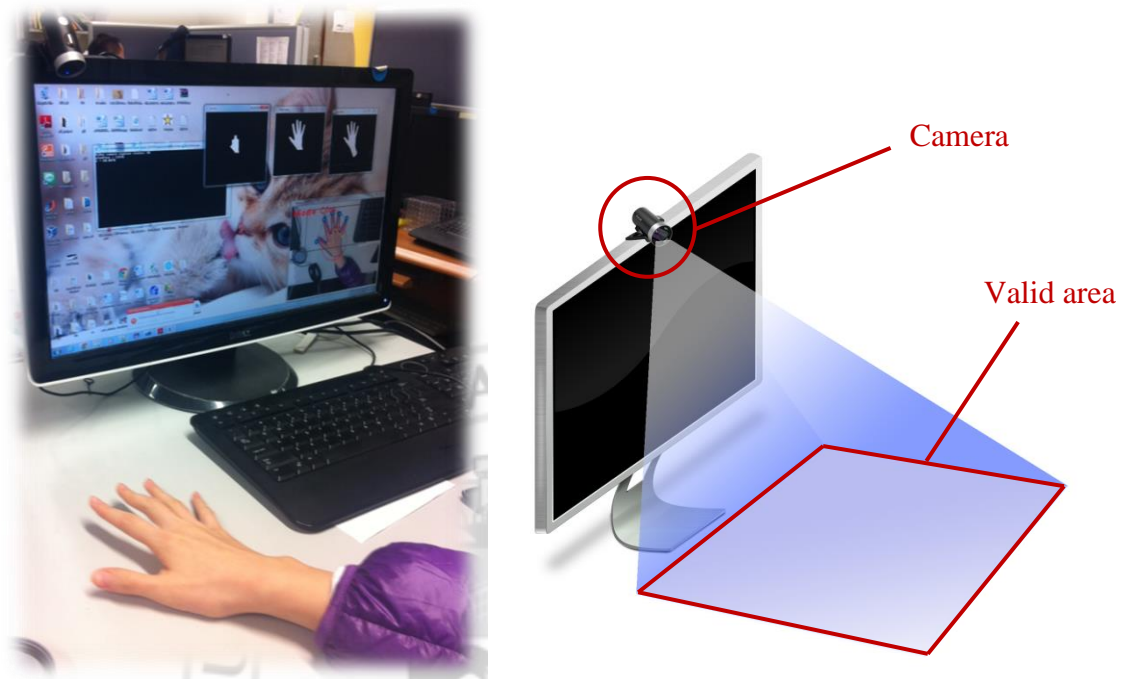


Figure 5: Camera position and valid area to be captured

The camera we use is the Microsoft LifeCam, the maximum resolution of the camera is 1920×1080 pixels and it provides 30 frames per second. However, in our system, the resolution is 640×480 pixels. [8]



Figure 6: Microsoft LifeCam

1.4 Summary of Workload

This section summarizes our project scope, the development process and the work distribution. In additions, the program scale, the performance and the interface will be also discussed in this section. Finally, we will talk about the difficulties in studying an realizing the gesture recognition system.

1.4.1 Project Scope

The project is developed for the hand gesture recognition in Microsoft Windows platform. The hand gesture will be processed after captured by the camera. Therefore, users can control the computer with their hands.

Furthermore, we also implement four applications to evaluate the usability of the gesture recognition. The applications include the Computer Game, PDF Reader, Video Player and the Internet Browser.

In the future, the system can continue to develop, and extend with more applications that make use of the developed gesture recognition model. For example, apply the gesture recognition technique to work with Google glass and mobile devices.

1.4.2 Development Process

We will specify our process for the system development; especially the Software Development Life Cycle model (SDLC) for this report will be described in detail.

Feasibility study

At the beginning of this project, we want to develop a system that can perform touchscreen on different objects, such as a touchscreen for wooden table. We reviewed some papers about the finger detection or object tracking methodologies. We found that it can do many interactions by using gesture when compared with touchscreen technique. According to the result of those researches reported in the literature, most of their gestures based application are very simple, so that we want to further apply them to advance applications and make it popular and easy to use.

Analysis

In order to minimize the cost of necessary devices, we will use one camera to capture the hand gesture. Compared to multi-camera recognition system, it is easy to set up in different environment, using less processing time and less computing power. Compared with multi-camera paradigm, a single camera approach is unable to use the ‘deep’ information for determining the precise gesture data. However, in this work, we are going to tackle this by improving the existing gesture approach in different analytical steps and algorithms.

We found that the OpenCV has provided many image processing algorithms. It can save our time that we do not need to implement those algorithms again. We had tried a simple hand recognition program in Microsoft Visual Studio 2010 with C++ project. It can detect the shape of hand, but the quality of that program is not well. However, our designed gesture recognition system is based on those programming prototypes in our project.

In Microsoft Visual Studio, we are able to create the interface using the MFC framework. It provides us the GUI wizard to set up the required interface. Therefore, we choose MFC for system interface development.

Design

In our system, the captured images are delivered to the computer for processing. Besides the one hand gesture, we suppose the user can make use of two hands for more operations. Therefore, it has more combination of gesture for user to use.

Based on the static gestures, we can add the dynamic ones, which will trigger when the moving speed and direction is satisfied.

We also provided four applications to use, such as Game, PDF Reader, Video Player and Browser. The user can use those applications to test the usability of our program. Actually, different applications have their shortcut of operation. Therefore, we decide to define a set of gestures for different applications. The user can change the setting if they like.

The default combinations consist 14 different kinds of gesture, as shown in Table 2: Gesture table in Chapter 3, 6 of them are dynamic gestures. However, the combination of gestures can create a huge number of gestures in general. We will provide a setting in the future to allow users to modify and associate the operation for specific gesture.

Implement

First of all, it will transfer the image color into grayscale image that contain hand shape only. Based on the grayscale image, we can extract the features of the image. We can analyze the distance of fingertip, moving speed for the gesture and the path of movement. The information becomes the cues to match the gesture from gesture table to perform the specific action.

However, we had met some serious problems during implementation.

1. When we try to control the mouse movement activity, we find that the mouse cursor cannot move smoothly.
2. The cursor is shaking even the users do not move their fingers as they thought. The reason relates to the transferring of the real object from the real world to the digital world. The edge of the object is different in each frame captured image.
3. After we optimized the processing time by modifying detection technique. It leads to unexpected reaction when changing between the gestures. For example, one finger changes to five fingers, two fingers gesture had been detected by the program. It is caused by the camera processing many images in one second.

In order to solve these problems, we added some optimization operations in the program. The detail solution is written by corresponding handler in Chapter 3.

Testing

In our testing, we will let the each user perform each gestures 20 times, and static the accuracy for the gestures. There are total 14 different gestures and 5 users are included in this testing. The accuracy of gesture testing is 90.86% on average. Most of the users satisfy to the interface and mouse control. Furthermore, the processing time for one image is 0.0825 seconds on average. This is fast enough to perform the real-time operations.

1.4.3 Work Distribution

System Description

Our system combines with two main programs, interface processing and gesture recognition processing. The interface is used to interact with the user, provided the application for user to use. The gesture recognition processing is the main program for feature extraction and controlling the computer.

Work Distribution

Our project included 2 members - Anika and Ben. The modules and the functions, and the work distribution are specified here:

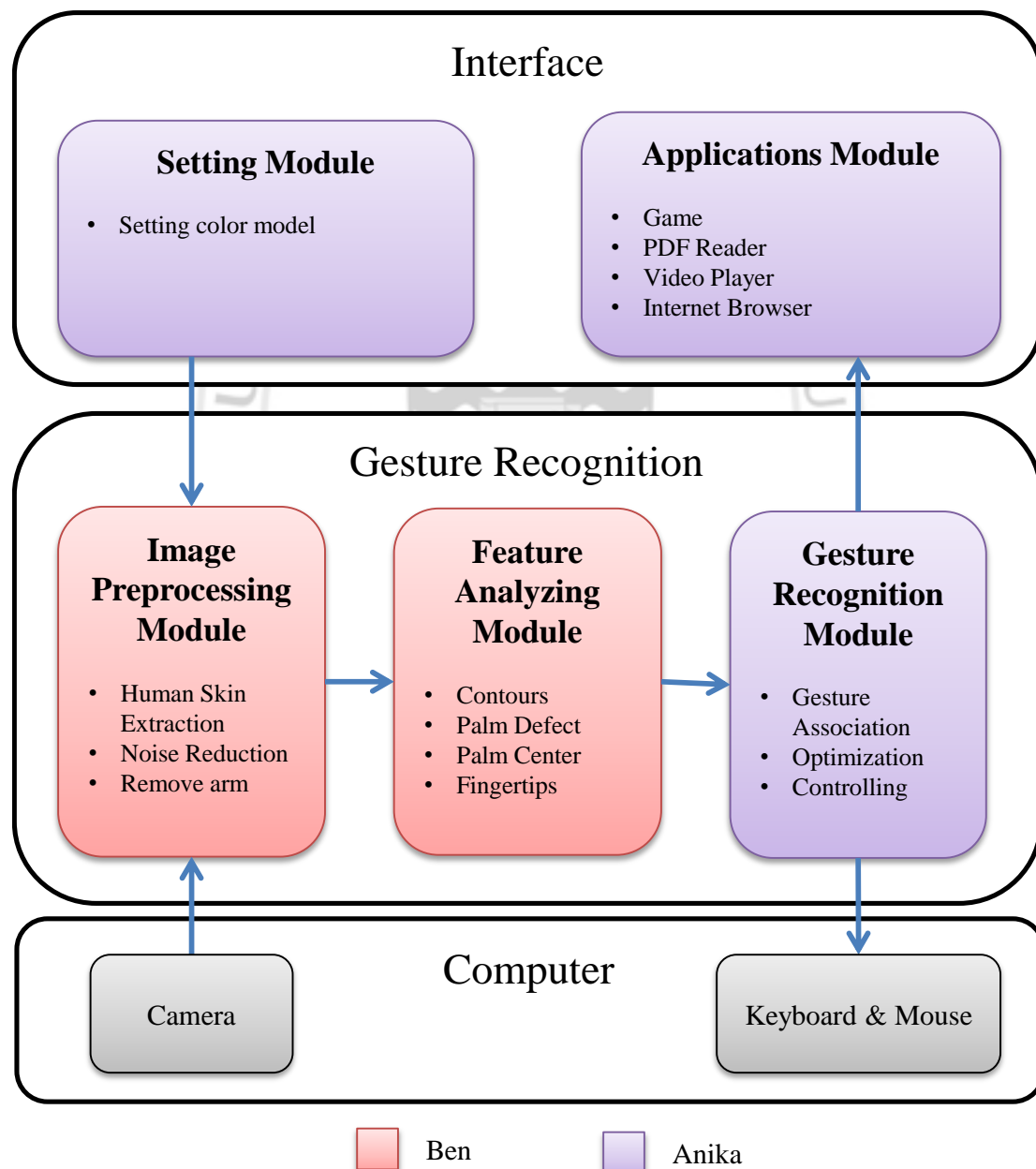
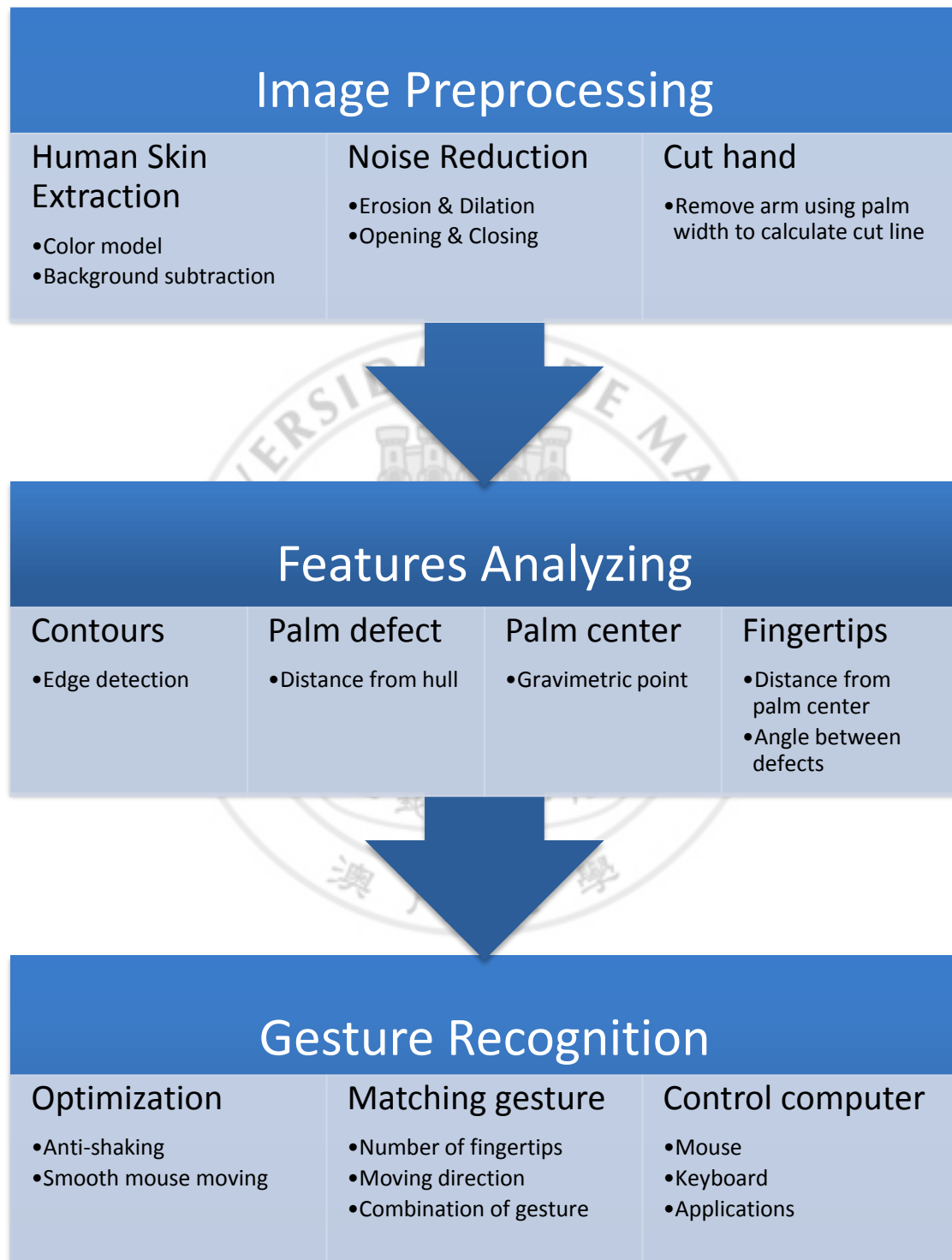


Figure 7: Work distribution

1.5 Technologies Description

This project includes 3 modules to process the gesture recognition. There are Image Preprocessing, Feature Analyzing and Controlling. Each of them has their methodology to realize the purpose, the detailed discussions are in the Chapter 3.



1.5.1 Difficult During Development

Hand gesture recognition is a hot topic but hard to make it perfectly. It has many difficult problems are unsolved by other people. During the development, the problem we met are:

1. Skin color: Different users may have different skin color; it is hard to make sure the system suitable for any user.
2. Environment: In the environment, the background, lighting effects and shadow are hard to predict. It will affect our accuracy when detecting the gesture.
3. Gesture setting: Different user has their habit when using gesture. It is hard to match the function to each gesture and adapt to everyone.
4. Shaking problem: The finger position shaking every frame since the camera refreshes the image, which transfer the real object into digital signals. Therefore, it is hard to make the shape of hand stable.
5. Innovation: In order to make our project different from others, we have created some new ideas to control the computer. However, the new idea may not be easily accepted by users. It should spend time to explain the functionality of the gesture.

However, the environment changing will affect our accuracy seriously. It is because all processing is based on the grayscale image which produced by color filtering. And the color filtering method is sensitive to the light source since it will render some color on the object. Therefore, we have a limitation that it has to adjust the color model when changing the environment.

CHAPTER 2. RELATED WORK

Nowadays, the gesture recognition is one of the hot topics in the world. Some famous companies are providing their system into different areas. There are also some researches released on the internet, which are about gesture recognition by using different methodologies.

In the following, we will introduce two researches from the famous companies, and two researches by other people.

2.1 Collaboration with a Robotic Scrub Nurse

2.1.1 Background

In May 2013, ACM released a research that using gestures to control the robot as a nurse. [9] The research purpose is they found that there are 31% of all communications in the operating room represent failures, with one-third of them having a negative effect on patient outcomes.

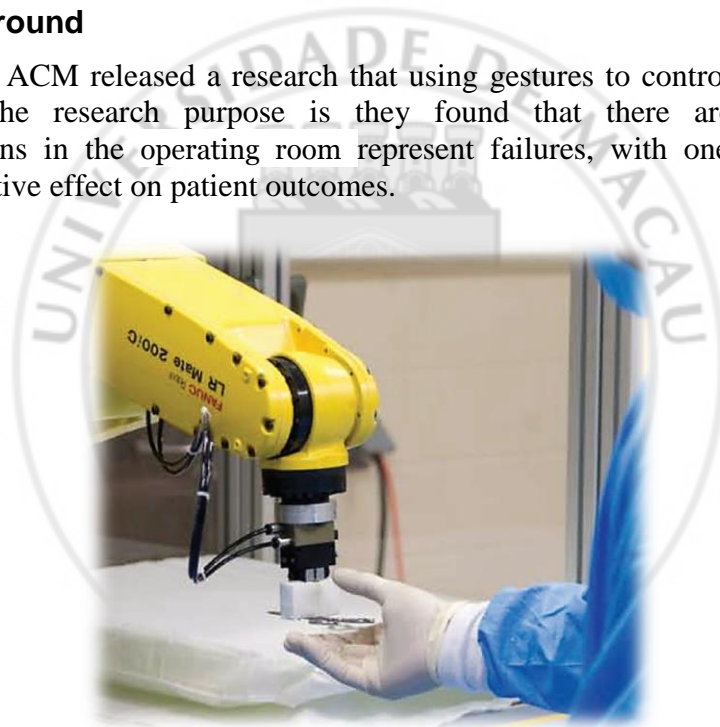


Figure 8: Robotic Scrub Nurse

2.1.2 Gesture Recognition

The research is using the Kinect sensor and segmented from the background through a depth-segmentation algorithm. The processing time for each image is about 160ms to recognize the gesture. The robot also need 2 seconds on average to transfer the tool to the doctor.

The system provided seven standard types of surgical instrument: scalpel, scissors, retractors, hemostats, clippers, forceps, and hooks. Combine to the movement, there are 5 static and 5 dynamic gestures.











 (a) Scalpel	 (b) Bandage Scissors	 (c) Hook Retractor	 (d) Forceps	 (e) Hemostat
 (f) Needle	 (g) Speech On/Off	 (h) Scissors	 (i) Sleep/Wake	 (j) Retractor

Figure 9: Gesture table for surgery

2.1.3 Experiment Result

They split the users into 3 groups to perform the tasks several times. Finally, the gesture-recognition accuracy is about 95.96% on average.

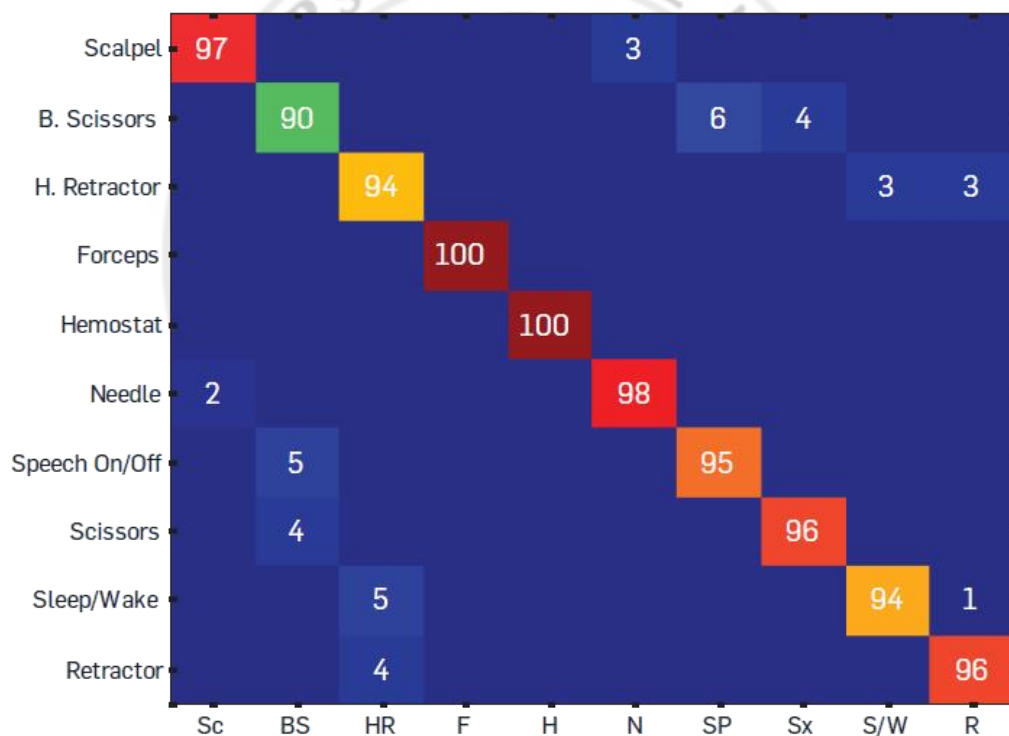


Figure 10: Testing result for ACM research

2.2 Turns Paper into a Touchscreen

In April 2013, Fujitsu has developed a technology that can detect finger and where is it touching in the real world, and turn into any surface. For example, a piece of paper turns into a touchscreen. The tools they used are ordinary webcam, plus a commercial projector. [10]



Figure 11: Using paper as touchscreen

2.2.1 Gesture Recognition

The method for recognition has used Binocular disparity principle, which can calculate the distance between object by the different angles. This system provides only 2 gestures for the user, the pointing and holding. The system can capture the distance for the user's finger, to predict the touch action.



Figure 12: Height information for fingertip

The system can detect the position of fingertip, even the book contains skin color image. It is because they are using two cameras to get the 3D position of the fingertip, which reduce the reliability to skin color.

2.2.2 Other Functions

Beside the click action, the system can capture images, and perform OCR to search information from the internet. It can draw some notes and manage the notes to the document.

However, the system is still under testing and the commercial version may on sale at 2014. [10]

2.3 Gesture recognition for digits and characters

2.3.1 Background

This research is a system to use the gesture as input digits and characters. [11] The process of recognition includes three main parts, which are hand detection, fingertips detection and gesture recognition.

2.3.2 Hand Detection

It is using color model to get the area of skin from the image, and then find out the area of the palm and using it to calculate the gravimetric point.

This research avoid capture the face because the color of the face is the same as the color of the skin.

2.3.3 Fingertips Detection

The method is using edge detection and the mask model to find the coordinate of the fingertips.

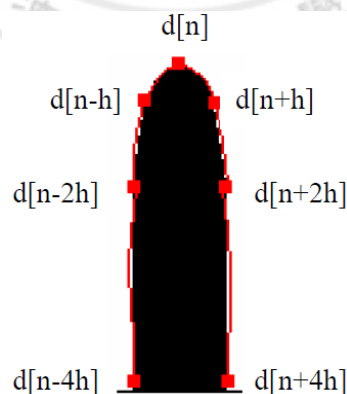


Figure 13: Mask model

2.3.4 Gesture Recognition

At last, it will find the appropriate gesture according to the angle of fingers and the relation of distance. This research can input the digits using one hand gesture, and input characters using two hands gesture.

2.3.5 Experiment Result

In the testing phase of the research, it summarized the result of digits recognition and character recognition. The system is overtraining for the author's hand, so that the author's accuracy is about 97%, but the other's accuracy is lower than the author.

2.4 A Two-Hand Multi-Point Gesture Recognition System Based on Adaptive Skin Color Model

In that research, the system can get the hand shape automatically in different environments. It has four main steps, which are object detection, object segmentation and tracking, feature extraction and gesture recognition. [12]

2.4.1 Object Detection

At First, it is using a camera to get an image, and then change it to a grayscale image. And then use Haar-Like Features database that had recorded some starting image. The system will find whether the grayscale image similar to the image which in the database so that it can detect the image whether exist hands.

2.4.2 Object Segmentation and Tracking

The method is using Opening and Closing algorithm to reduce noise and then use Connected Component Labelling to cut the hand. After that, it can calculate the gravimetric point and use it to track the amount of displacement of hands.

2.4.3 Features Extraction

There are two main parts, which is static feature and dynamic feature. Static feature is used the gravimetric point of hand to be a center and then create a Polar Hand Image to find which the fingertips are. Dynamic feature can calculate the direction of hands and the angle of movement according to the Gradient.

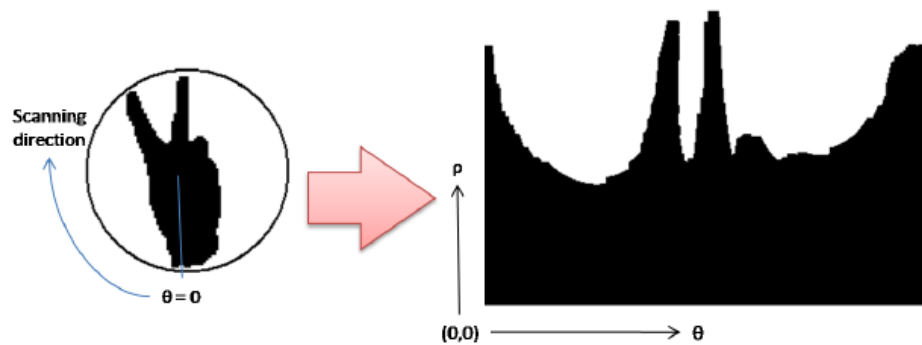


Figure 14: Radar scan

2.4.4 Gesture Recognition

That project is using the photo browser to be an example, it direct 3 kind of gesture for user to use, which are Slide, Zoom and Rotate.

2.4.5 Experiment Result

In that research, the system may miss the user's hands, and the reason has two main parts, which are the data of Haar-Like Features database that do not enough and the user's gesture do not match with the sample, so that the accuracy is about 80% and the hand recognition accuracy is about 89.3%.

CHAPTER 3. DESIGN

3.1 Overall System Design

The system is separated into two parts: interface part and processing part.

Interface part is controlling the interaction between software, such as: browser, video, virtual keyboard, etc. This part is completed by Anika.

Processing part is controlling the computer through gestures, which is the main part of our system. Ben has focused on the pre-processing of the image, extract features and some optimizations. Anika is dealing with the post-processing, control computer, application and some optimizations.

The following flowchart shows the workload of members in this project. Blue modules mean finished by Ben. Red modules mean finished with Anika. Purple modules mean finished by both of the member.

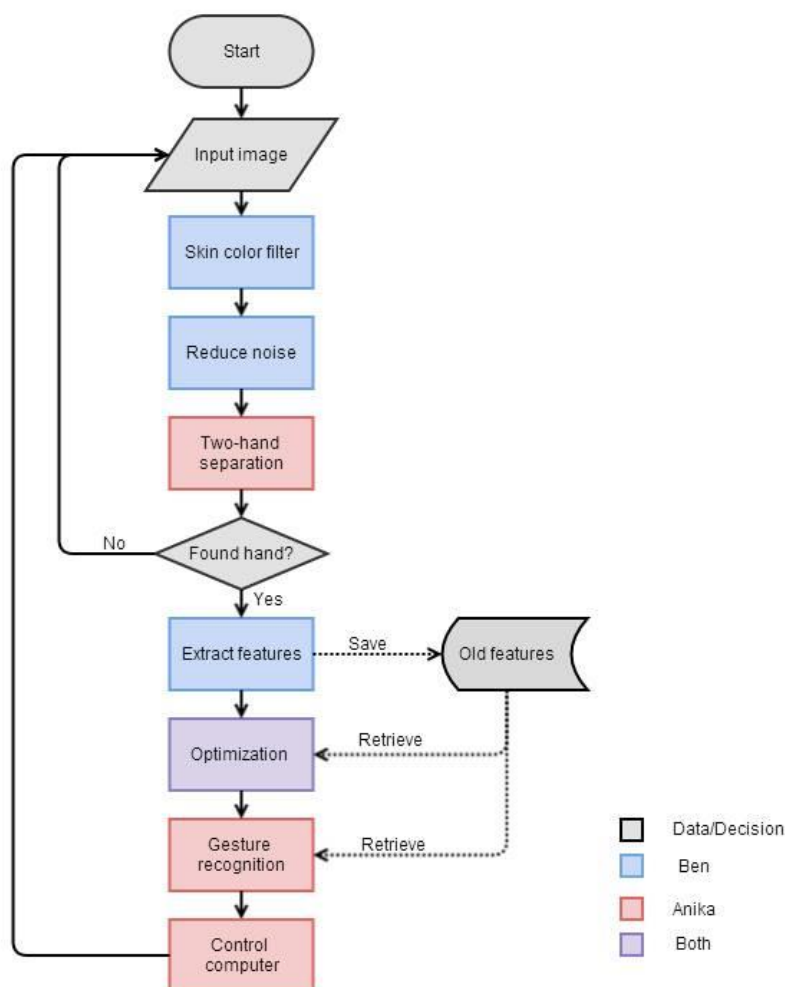


Figure 15: System flowchart

3.2 Gesture Recognition

Human can perform many different gestures with their hand, but it is hard to unify a standard of gesture for everybody. Normally, different systems have different set of gestures, which is the best for that system only.

In our planning, users can use two hands to control and we will provide static gestures and dynamic gesture for users to use.

3.2.1 Separate Two Hands

We provide the user that can use two hands to control the computer because the gestures of one hand is very few and limit, the combination of two hands can let users do more things.

When we get an image from a camera and have found out all the contours, we can know which the two biggest contours are, we define that will be the hands. Unfortunately, there are just two contours and we do not know which is left hand and which is right hand, it will have a problem to recognize. There is a method the solve it, we can calculate the center of two contours, we can find the correct answer according to the x-axis of the centers.

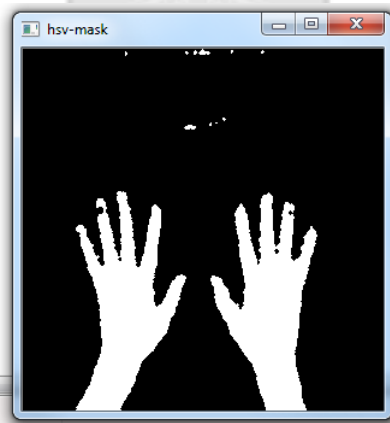


Figure 16: Before separate two hands

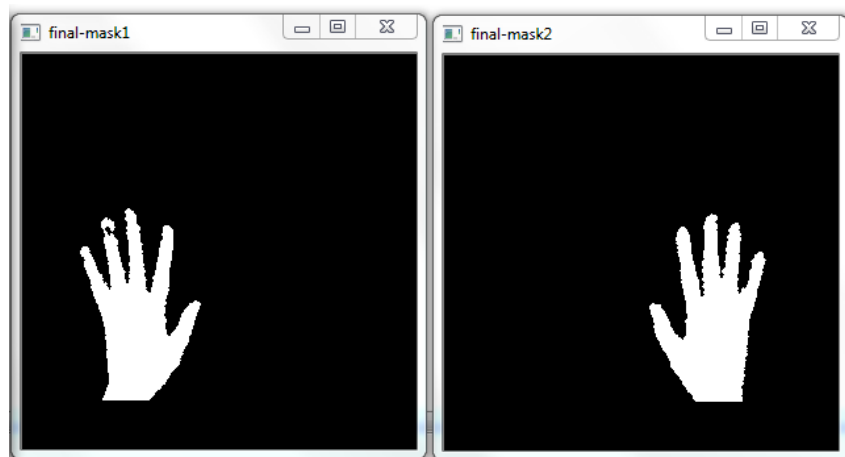


Figure 17: After separate two hands

3.2.2 Six Basic Gestures

We think that the gesture must be very simple for people to do, so we have six basic gestures, and then use them to do some combination. There is a table that explains how to recognize the gestures.









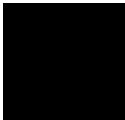















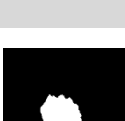

Number	Right Hand	Method
1		No finger
2		One finger
3		Two fingers Two fingers' distance more than 100 pixels
4		Two fingers Two fingers' distance less than 100 pixels
5		Three fingers
6		Five fingers

Table 1: Basic gestures

In our system, it has static gesture and dynamic gesture, also supports two hands recognition. Therefore, it can increase the number of gestures. Besides the gesture, our system can recognize some specific movements to do the operation.

There are the gestures that we had defined for user to use, we just define fourteen gestures because too many will make user confuse and hard to remember. Basically, we will add the setting for the user to match the gestures and function.

Number	Left Hand	Right Hand	Moving	Meaning
1			None	None
2			None	Control the position of the mouse
3			None	Left click
4			None	Right click
5			None	Middle click
6			Up/Down	Page up or Page down
7			None	None
8			Left/Right	Previous tab or next tab
9			Left/Right	Previous page or next page
10			Up/Down	Open or Close the On-Screen keyboard









11			None	Open a new tab
12			Left/Right	Zoom in or out
13			None	Video play or stop
14			Left/Right	Increase or decrease the volume

Table 2: Gesture table

3.2.3 Simulate Mouse Click

iFinger use camera to recognize finger's movement to do some position, however, there are some problems can affect the result, just like the light source, it will make the photos different although the hand haven't moved, we call it shaking. When we shake the forefinger like using the left click on the mouse normal, it is difficult to recognize whether it is left click or shaking, so that we use the thumb and forefinger's movement to recognize. When there is forefinger, it will be a cursor to point something at the desktop. When there are thumb and forefinger, it will do the left click.

3.2.4 Simulate Keyboard Shortcuts

iFinger not only can perform mouse actions, but also keyboard action. There are many shortcuts within an application, but they are always hard to remember and very complex. iFinger can let the user use their hands to make a simple gesture to substitute the shortcuts. When the user does difference gestures, it can recognize what the gesture they are, just like pressing the "Page Up", "Page Down", "Ctrl+N" etc.

3.2.5 Virtual Keyboard

There is an internal virtual keyboard call "On-Screen keyboard", we can use a gesture to open it and click the button like clicking keyboard. The user also can use a gesture to close the virtual keyboard. It can help user typing easily, and it is using an internal system, the user does not need to download additionally. The following picture can let us see that how it is working.

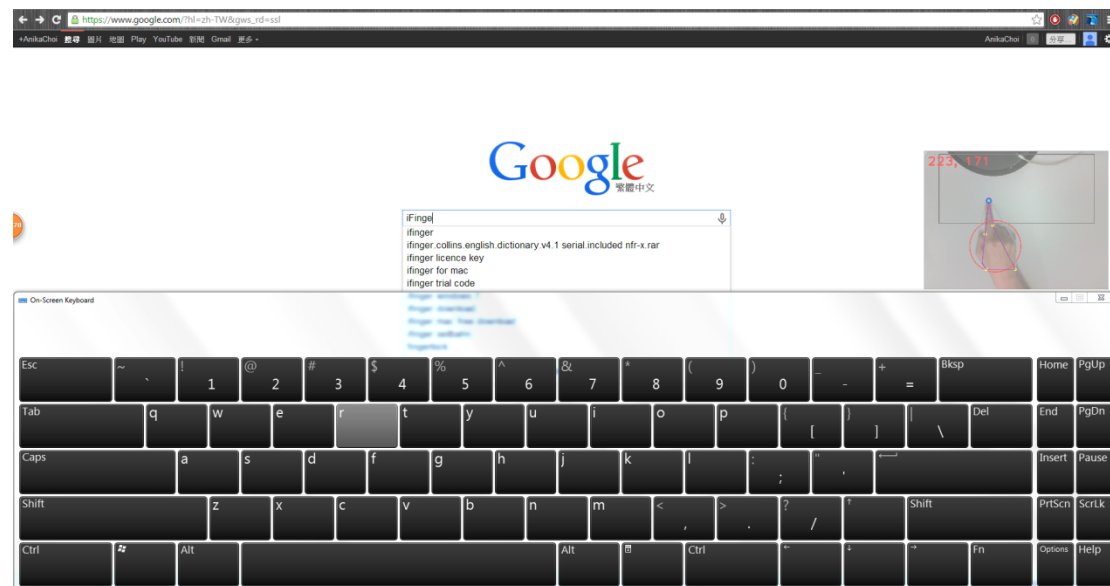


Figure 18: On-Screen keyboard

3.2.6 Unexpected Reaction

iFinger will analyze for each image and recognize the gesture, however, the run time of the system is very fast, when the user change a gesture to another gesture, it may have some unexpected reaction. For example, when the user is using one finger to make a gesture, after that he change to use three fingers to do other gesture, it may appear two fingers when he changes the gesture, it can make some mistake operation.

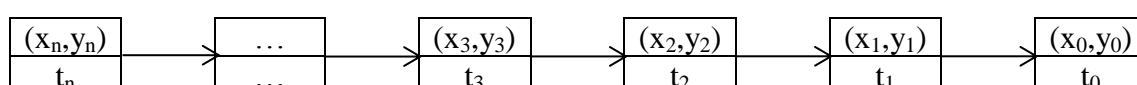
Finally, we do some flag to solve. When there is a gesture here, it will be saved as the previous gesture, when the gesture is done for two times in continuous, it will be accepted for the correct gesture and do the correct operation.

3.2.7 Recognize the Dynamic Gesture

In order to recognize the dynamic gesture, we have to know the moving direction, which means the user can keep the gesture and move with that direction. In order to detect the direction, we have to know the position and the time in each frame. First, we set a variable to summarize the processing time in each loop. Second, we create a data structure to record the pointing position and time.

```
struct Def_point_time {
    CvPoint point;
    double time;
...}
```

Finally, we can record them into a list and analyze the direction. The header of the list is the newest point of the program, and the last is the oldest point.



But how can we analyze the direction? The answer is we can use the time difference and distance to indicate moving speed. Based on the base operation of speed:

$$v = \frac{s}{t}$$

Therefore, we can calculate the moving speed and direction by using this method. For example, in the following figure, P_4 is the current point. We want to know where it came from in 400 ms ago. The program will check the time difference from the list to find out which point is bigger than 400. After that, it will calculate the distance between two points, and also the direction. Finally, the system can match the gestures.

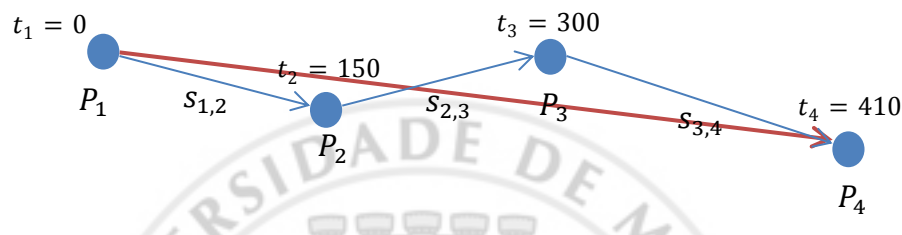


Figure 19: Direction tracking

3.3 History Move

Besides the gesture recognition, this is another operation for user to use. The system will remember the pointing path and analyze the shape of the path. However, we just finish a simple version of history path analysis.

Basically, we separate the screen into different section. The historic path will remember the section number to guess the shape of the path.

$$\text{Section list} = \begin{cases} \text{add new section, if section change} \\ \text{no change, otherwise} \end{cases}$$

For example, if we want to draw a character 'S' to represent 'Setting', we can define the path like this:

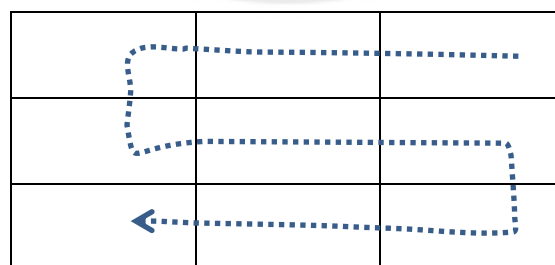


Figure 20: History path analysis

The program will detect the shape 'S' when the user pointing at the screen like this.

Actually, we want to make the history path more freedom. It means the user can draw 'S' shape everywhere on the screen. However, it may cause many unexpected reactions in the program. It has to balance the accuracy and usability, we choose accuracy as the main consideration.

3.4 Interface

iFinger combine with two main programs, processing interface and processing gesture recognition. Interface.cpp is using an MFC Application form the Microsoft Visual Studio. [13]

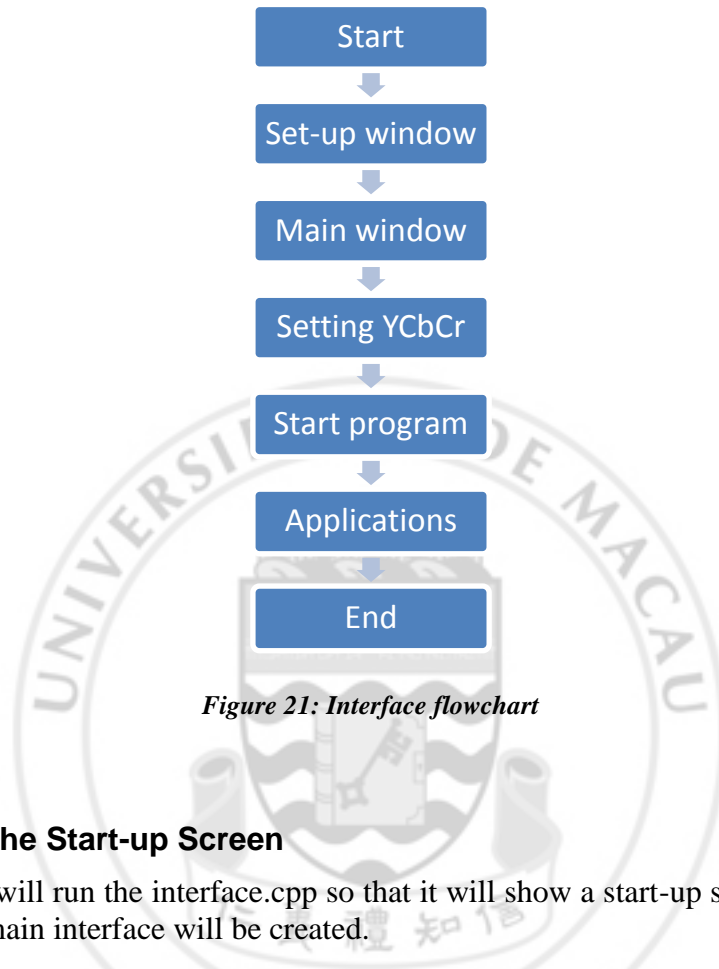


Figure 21: Interface flowchart

3.4.1 Open the Start-up Screen

First of all, it will run the interface.cpp so that it will show a start-up screen, after five seconds, the main interface will be created.



Figure 22: Start-up screen

3.4.2 Main Interface

After five seconds, the main interface has already been created. There are some buttons here and they are Start, Setting, Help, Information and Exit.



Figure 23: Main interface

3.4.3 Setting YCbCr

iFinger is using detect skin colour to recognize the gestures, however, there are some reason will influence the detection. First, the skin colour of people is different, someone is white skin, someone is black skin, and the default setting may not be matched with the user. Second, different place have different light, the light will influence the skin colour of people in an image, it also cannot detect correct.

Because of those reasons, we can let user to set the YCbCr numbers so that it can detect the skin colour more accurately. We use the read file method the save the YCbCr numbers because there are two programs here, they do not have the same variables. The user can set an optimal numbers to detect the hand before starting.

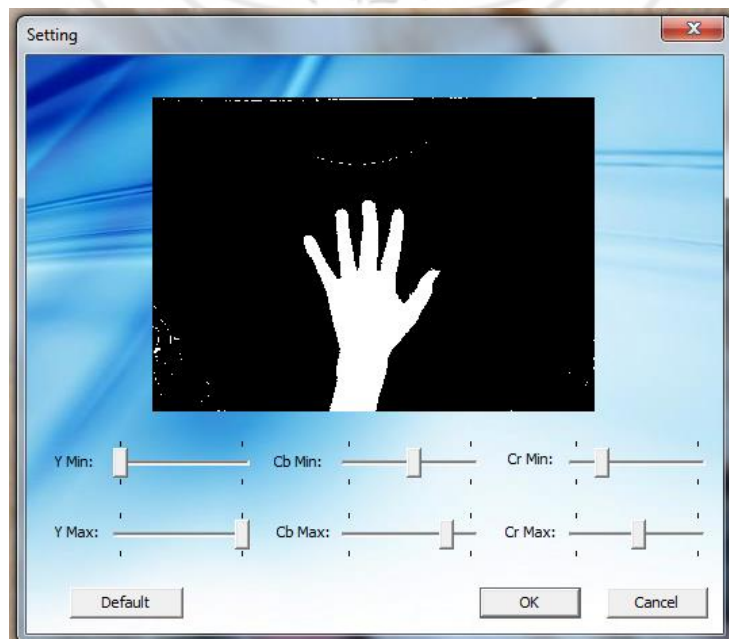


Figure 24: Setting window

3.4.4 Start Program

After that, there is a “Start” button, press it and it will create a new process and its thread to run the skinDetect.cpp, and then it will hide the “Start” button and show the other function buttons. There is a working window show on the right-bottom corner of the desktop, user need to show his fist in three seconds, the program will calculate his fist size, otherwise, it will have an auto size. The fist size can help the computer find the fingers because there is no finger will appear in that area.

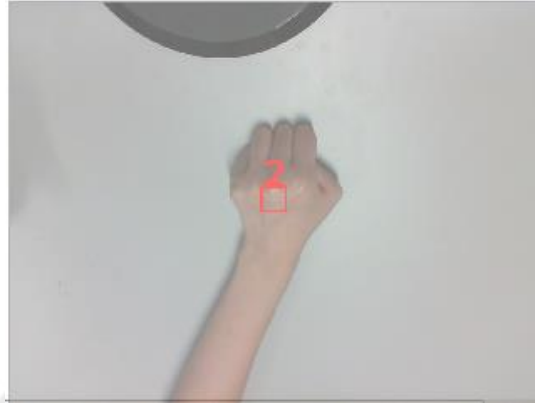


Figure 25: working window

3.4.5 Working Window

It is an important component because we capture the image from the camera; the user may confuse their hands position whether captured by camera. In order to show this information, we set a working window on the right-bottom corner of the desktop that can let the user know the valid area to detect their hands. However, iFinger is used to control the computer, but sometimes the working window covers the information and cannot do any operation under the area of the working window. Because of those reasons, we do the transparent to let users can read the information and see the working window at the same time.



Figure 26: Transparent the working window

iFinger is used the photo to recognize the gesture, when the gesture is out of the photo, it cannot recognize, so that if we set the photo size as the desktop size, it will have the problem here. Therefore, we set a size in the photo that can recognize the gesture and also can control the bottom side of the desktop.

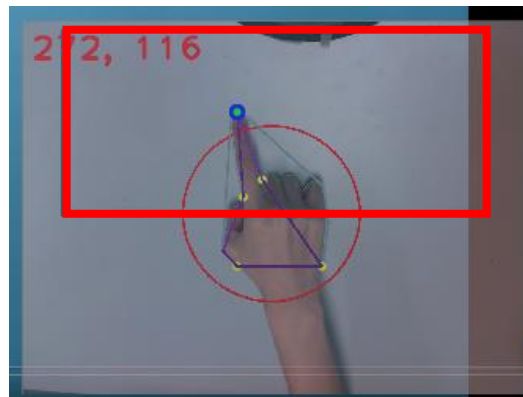


Figure 27: Control mouse area

In the above figure, the red rectangle presents the moving area for the cursor, which also the screen size. The position of the image has to change into the position of the screen.

Although the user can see the information under the working window, the user cannot click the area of the working window, it is also a problem here. Because of that reason, we set the working window move up when the cursor enter the area of working window, so that user can do actions at that area, when the cursor exit the area of working window, it will go back to the original area, the working window will not influence the user' action.

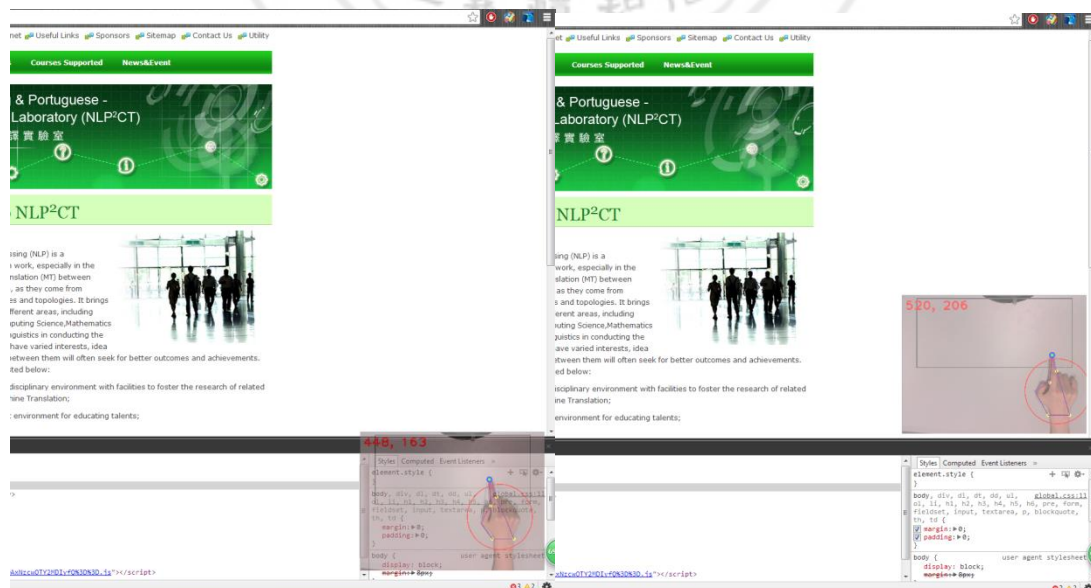


Figure 28: Moving the working window

3.4.6 Button Background Problem[14]

iFinger can control the computer by user's hands, so that user can use the computer more convenient. After clicking the "Start" button and the user show his fist to the camera, the interface has some buttons let user to experience the software, such as Game, Browser, Open File and Video Player, we want the user can familiar with gestures thought these applications.



Figure 29: Main interface

As we can see the button on the interface, the background cannot be transparent, so it is so strange when the window is different background colour with the background colour of the buttons. Luckily, my professors give a resource of NLP2CT lab and it can help me to set the background of the buttons to be transparent. The method is creating a new class "CHoverButton" and inheritance the class "CBitmapButton", and then set the background colour to be transparent, when I create the buttons, I assign their type is "CHoverButton" and the background of the buttons will be transparent.

3.4.7 Game

We want to select a game that is simple and easy to understand. The Cut the Rope is using moving and sliding to control the action. User can use their fingers rather than mouse to feed candy to a little green monster to clear the level.

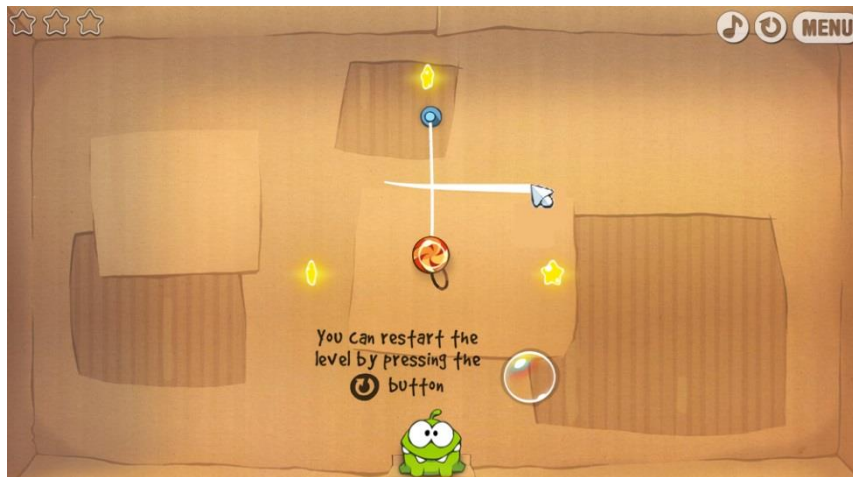


Figure 30: Cut the Rope

Number	Left Hand	Right Hand	Moving	Meaning
2			None	Control the position of the mouse
3			None	Left click

Table 3: Main gestures in Game

As we can see here is the interface of the Cut the Rope, there is a rope and a sugar, and then the user can do the Number 2 gesture to control the position of the cursor to an optimal position. After that, user can do the Number 3 and move and it will slide it as cutting to cut the rope.

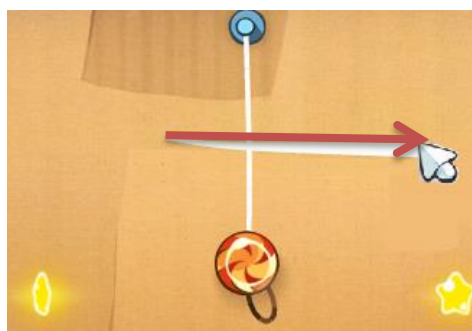


Figure 31 : Sliding in Game

3.4.8 Browser

In fact, there are many browsers on the internet, which are provided many shortcuts for the users, but the users are hard to remember the shortcuts. For example, back to the previous page is Alt+Left arrow. In this system, the user can use the browser more intuitively.

For the browser, Google Chrome, we provide some main shortcuts for it, such as page up, page down, change tab, open a new tab, previous page, next page etc.



Figure 32: Google Chrome







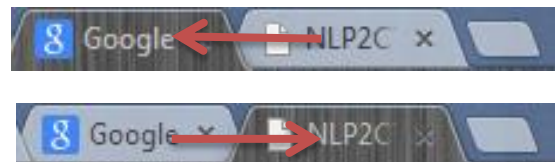
Number	Left Hand	Right Hand	Moving	Meaning
8			Left/Right	Previous tab or next tab
9			Left/Right	Previous page or next page
11			None	Open a new tab

Table 4 : Main gestures in Google Chrome

We think most of you are using browser will open many tabs for reading different website, user can do the Number 8 gesture to change to previous tab or go to next tab. When a user does the Number 8 gesture and move to left, the browser will change to the previous tab. When a user does the Number 8 gesture and move to the right, the browser will change to next tab.



Figure 33: Original tab



*Figure 34: Previous tab (upper)
Next tab (lower)*

The user may visit many websites in the same tab and they also can do the Number 9 gesture to back to the previous page or go to the next page. When a user does the Number 9 gesture and move to left, the browser will back to previous page. When a user does the Number 9 gesture and move to the right, the browser will go to the next page.

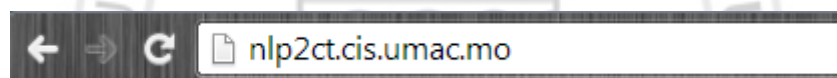
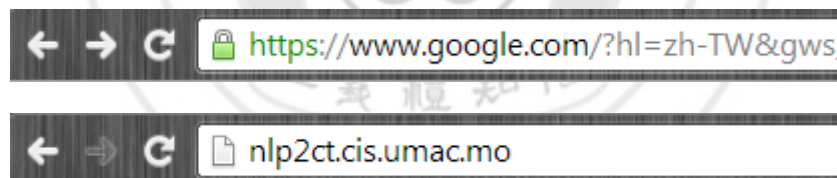


Figure 35: Original page



*Figure 36: Previous page (upper)
Next page (lower)*

When a user is visiting a website, he wants to keep that website and visit it later, but he also wants to visit another website, he can use Number 11 gesture to open a new tab in a browser.



Figure 37: Open a new tab

3.4.9 Open File

These functions allow the user to open some of the file from their computer. In order to simplify our work, we define it as open a PDF file. However, it can open an image instead.

Based on the type of file, the system will select the appropriate application to run the file. Each application has some corresponding functions for users to use. In a PDF file, it can allow the user to scroll up and scroll down the article, and etc.



Figure 38: Adobe Reader





Number	Left Hand	Right Hand	Moving	Meaning
6			Up/Down	Page up or Page down
12			Left/Right	Zoom in or out

Table 5: Main gestures in Adobe Reader

When we are reading a paper, it's always more than one page to read, so we can use the Number 6 gesture to move the page up or down.



Figure 39: Original reading page

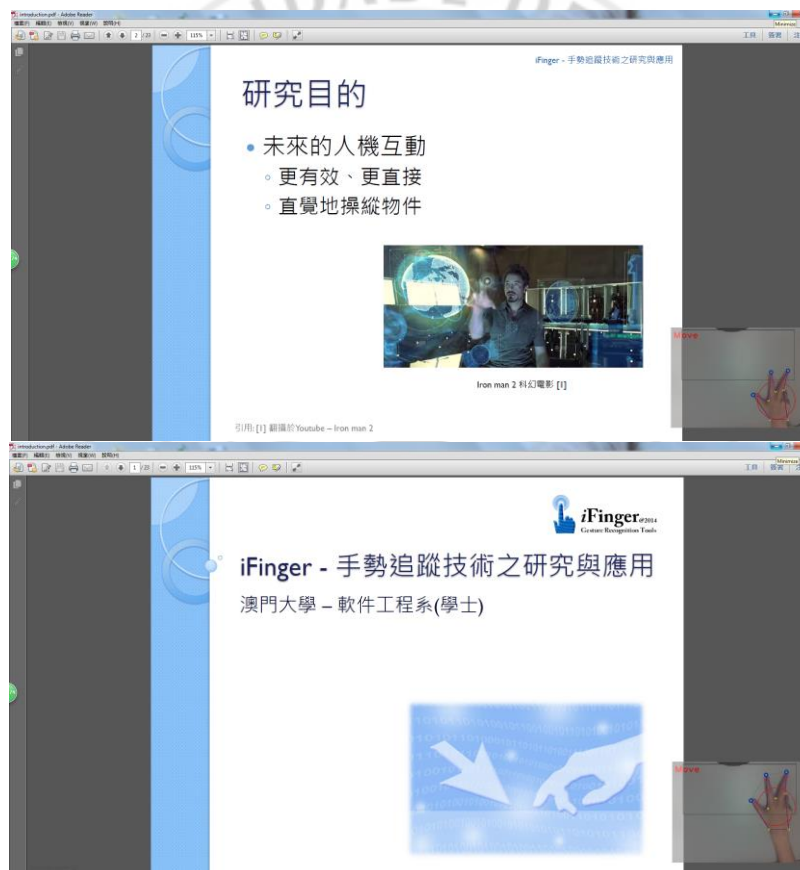


Figure 40: Page down (upper)
Page up (lower)

If there are some words are too small to read, we can use the Number 12 gesture to zoom in or out, the gesture is simulated as touch screen, so that user will remember it easily.



Figure 41: Original Size

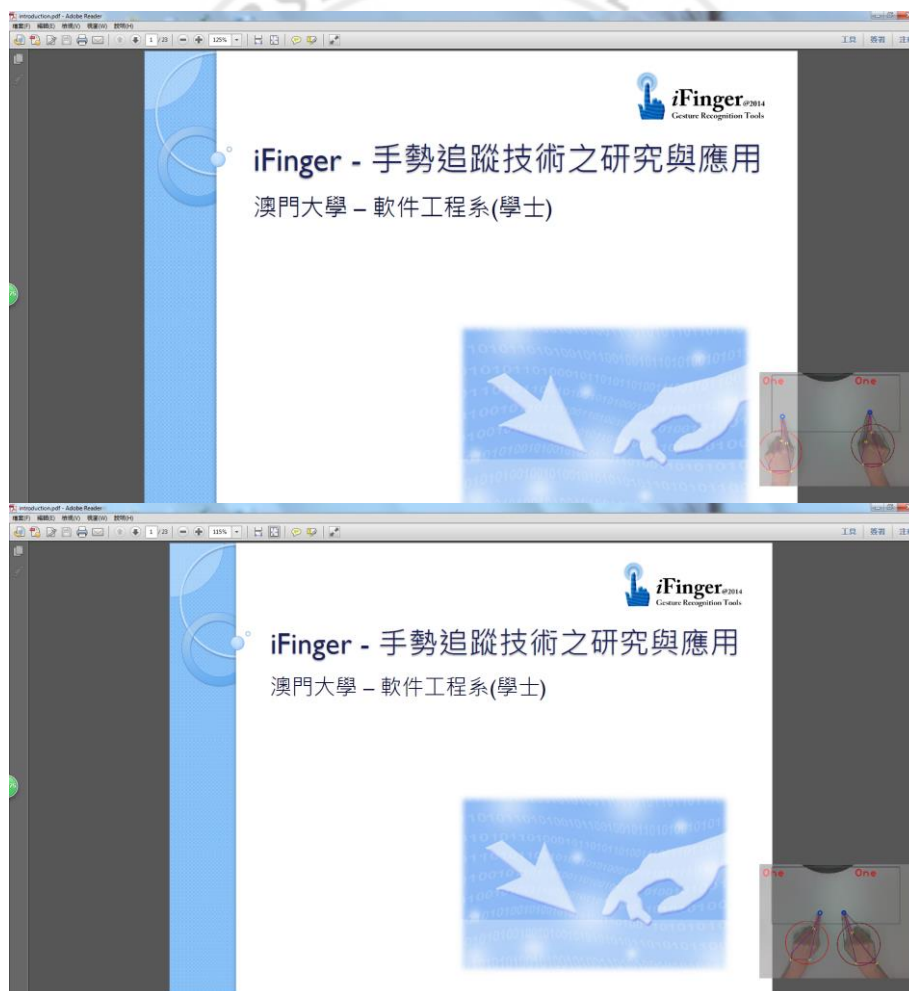


Figure 42: Zoom in (upper)
Zoom out (lower)

3.4.10 Video Player

Usually, watching video is one of the methods for study or entertainment. For example, mother is looking the video to learn how to cook, however, her hands are wet and holding some foods, they will not want to make their mouse and keyboard dirty, if they use iFinger, it can solve the problem, they do not need to touch the mouse and keyboard also can stop or continue the video and the volume.



Figure 43: Windows Media Player





Number	Left Hand	Right Hand	Moving	Meaning
13			None	Video play or stop
14			Left/Right	Increase or decrease the volume

Table 6: Main gestures in Window Media Player[15]

This application can let our mother try, when she is cooking and watching the teaching video, the video is going too fast and she cannot follow, she can do the Number 13 gesture to stop the video. When she wants to play the video, she can do the Number 13 gesture again and the video will go on.



Figure 44: Stop video

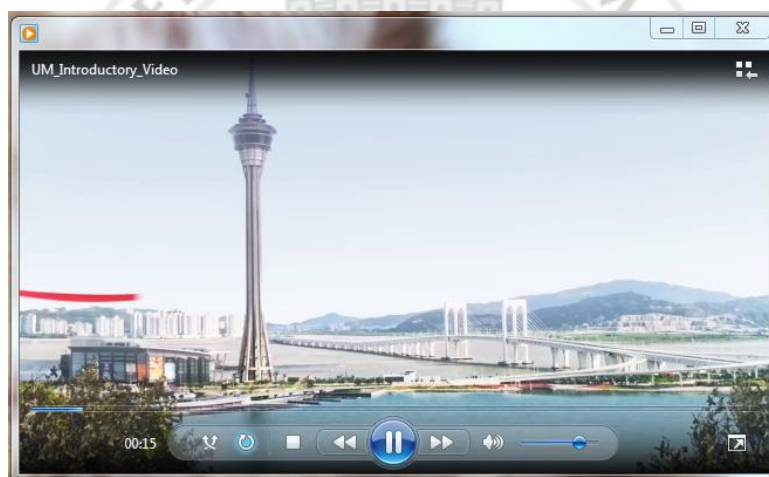


Figure 45: Play video

Sometime the volume will be too low, you want to put it higher, you can do the Number 14 and move right to be louder, otherwise, you can move left to be lower.



Figure 46: Original volume



*Figure 47: Decrease volume (upper)
Increase volume (lower)*

3.4.11 Help

After clicking the “Help” button, it will appear a window, there is a table that can let users review all gestures and it can prevent the user misunderstand the meaning of gestures, even after they are modify the function of gesture. The user can click the previous button and next button to switch.

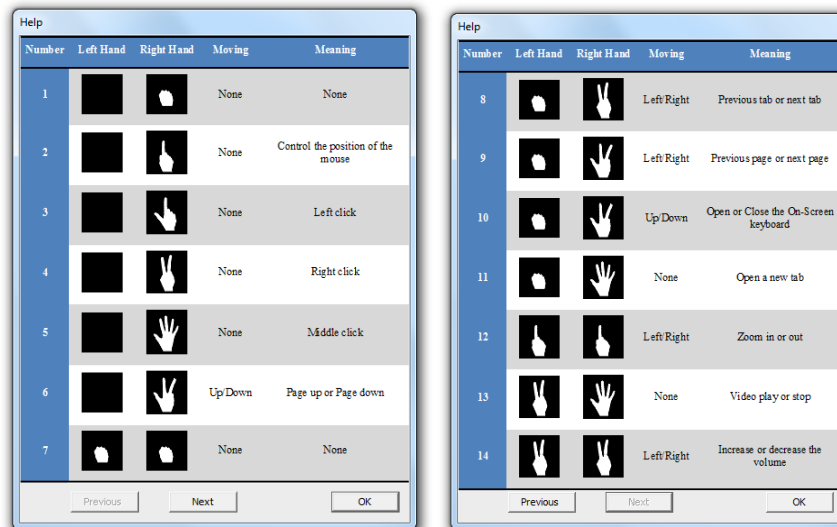


Figure 48: Help

3.4.12 About Us

“About us” can get some information about iFinger. It includes the author information and the version of iFinger.



Figure 49: About us

3.4.13 Exit the Program

Because of opening two programs, the skinDetect.cpp must be closed at the same time. It uses the process and thread to open, so that when user press the “Exit” button, it need to terminate the process and close the handle of process and thread.

CHAPTER 4. IMPLEMENTATION

In this chapter, we will deeply talk about the procedure of iFinger. We will have some codes and flowchart to explain the procedure. The following is flowchart of iFinger. As we can see, my work is Two-hand separation, Optimization, Gesture recognition, Control Computer and interface of iFinger.

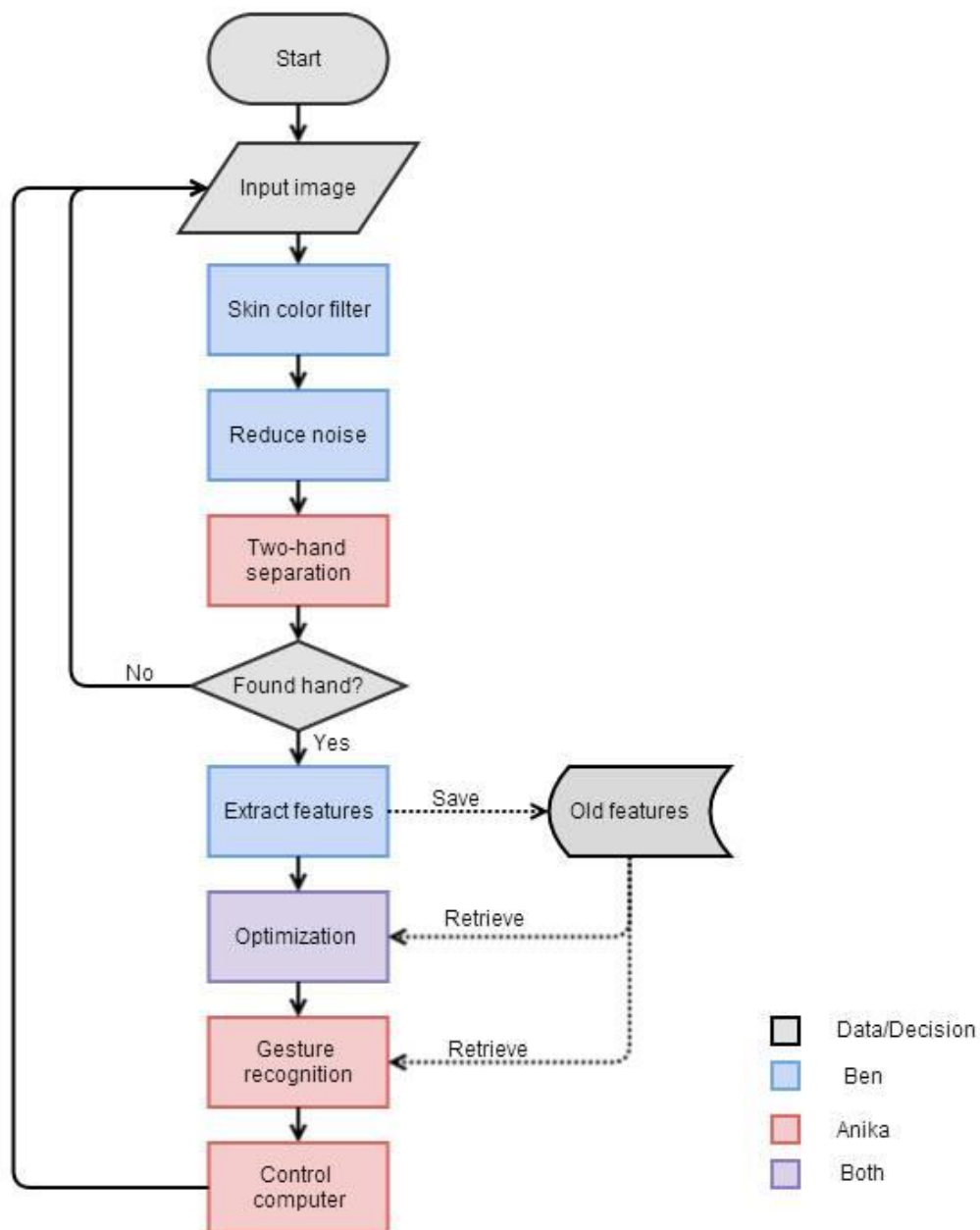


Figure 50: System flowchart

4.1 Two-Hand Separation

At first, my partner will do some pre-processing of the images so that it will return a gray scale image and the date of contours. When I have this information, I can find two biggest contours in the image, if the area of them are bigger than 2000, it means that the contours are valid, otherwise, it may be no hands or just one hand here. Now the contours are valid so we use findCenter() to find the center of the contours.

```
CvPoint center;
CvSize sz = cvGetSize(cvQueryFrame( capture));
IplImage* hsv_mask1 = cvCreateImage( sz, 8, 1);
IplImage* bi_dist = cvCreateImage( sz, 8, 1); //find center image

center = findCenter(hsv_mask1, bi_dist, sz);
```

After calculating the center of the contours, we can use their x-axis to analyze. If the x-axis of the first contours is smaller than the x-axis of the second contours, it means that the first contours is left hand and second contours is right hand, otherwise, the first contours is right hand and second contours is left hand.

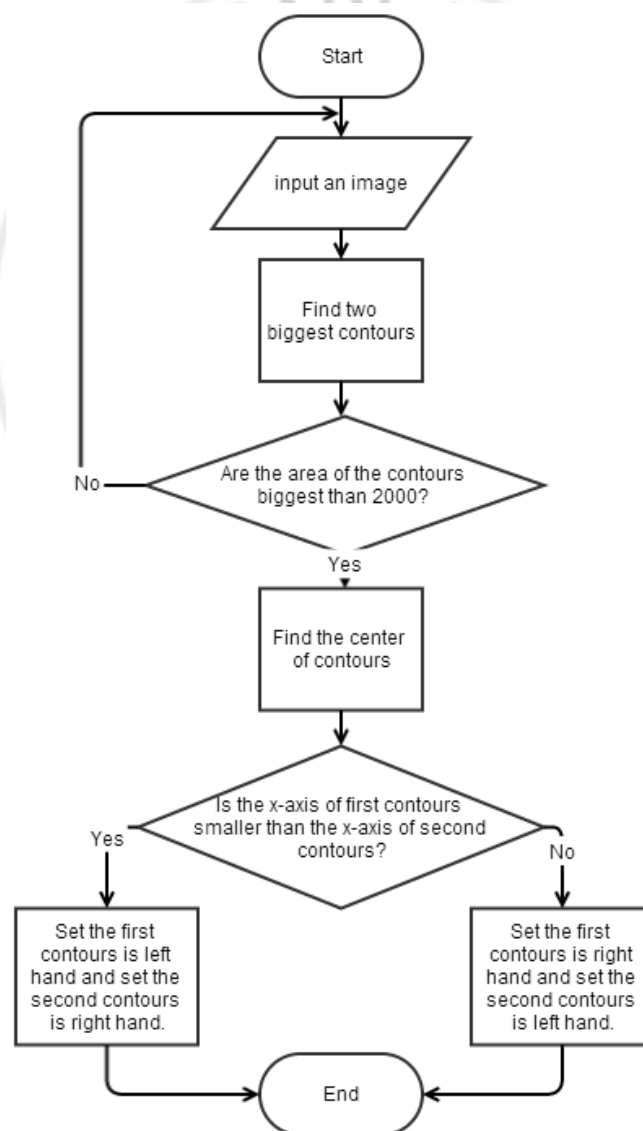


Figure 51: Two-hand separation flowchart

4.2 Gesture Recognition

It is because iFinger can use two hands to control the computer, it must separate two parts to recognize. First part is to recognize one hand gestures and I can get the finger numbers by my partner's analyze, after that it can know whether the finger is valid. If it is valid, the recognition of gesture is a success, and it will do the corresponding motion later. Second part is to recognize two hands gestures,

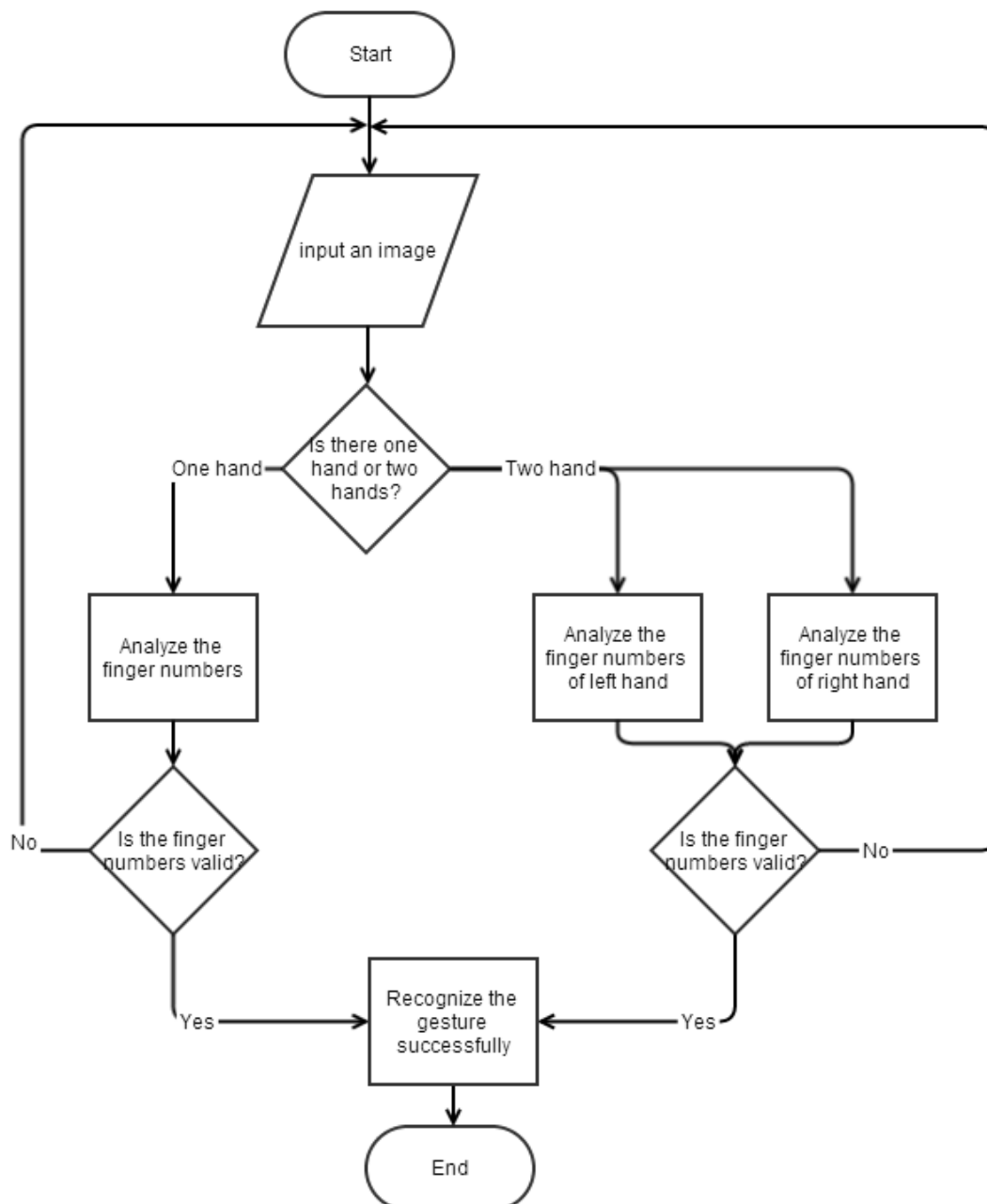


Figure 52: Gestures recognition flowchart

4.3 Optimization

There are some tasks we have done for the optimization, they are transparent the working window, working window auto moving. There optimization can make user convenient and will not influence user the control the computer.

4.3.1 Transparent the Working Window

The working window shows the current status of the gesture from the camera. However, it will cover the information behind the working window. We search the internet for the transparent method[16], [17]. We found:

```
bool TransWindow(HWND window, unsigned char opacity)
{
    if(initi == NULL)
    {
        HMODULE dynmall = LoadLibrary(L"user32");
        pSetLayeredWindowAttributes = (PSLWA) GetProcAddress(dynmall,
"SetLayeredWindowAttributes");
        initi = true;
    }
    if(pSetLayeredWindowAttributes == NULL)
    {
        return false;
    }
    SetLastError(NULL);

    SetWindowLong(window, GWL_EXSTYLE, GetWindowLong(window, GWL_EXSTYLE) |
WS_EX_LAYERED);

    if(GetLastError())
    {
        return false;
    }

    return pSetLayeredWindowAttributes (window, RGB(255, 255, 255), opacity,
LWA_COLORKEY|LWA_ALPHA);
}
```

This method can change the window become transparent, so that the user can see the information behind the working window and see the capture of the camera at the same time.

4.3.2 Working Window Auto Moving

Besides the transparent setting, we also implement the method that will let the working window move away when the cursor pointing into the area of the working window. The working window will return back to the default position if the cursor leaves the area. The optimization can let user can see the status of the camera and also can do some action at the working window and it will not influence the user' action.

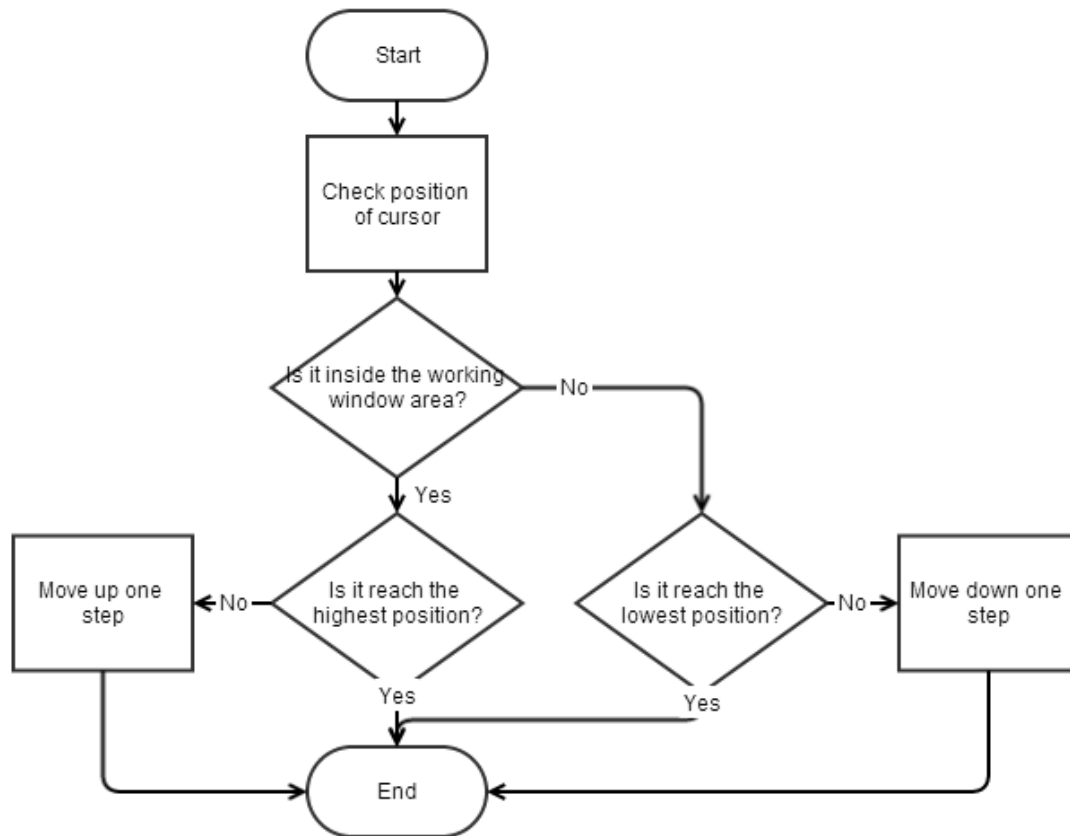


Figure 53: Working window auto moving flowchart

The following are the codes about working window auto moving:

```

//move window if the mouse pointing into the area
if( xCursor > srcX && xCursor < srcX+srcWidth/2 && yCursor > srcY && yCursor <
srcY+srcHeight/2)
{
    if( currentStep != step){
        locationY -= moveSize;
        cvMoveWindow("Webcam", srcX, locationY);
        currentStep++;
    }
} else {
    if( currentStep != 0){
        locationY += moveSize;
        cvMoveWindow("Webcam", srcX, locationY);
        currentStep--;
    }
}
}

```

4.4 Control Computer

Our main idea is to use iFinger to control computer so that do not need to use mouse and keyboard. iFinger just use a camera capture some pictures, user can do some gesture to simulate mouse click and keyboard shortcuts to control the computer.

However, the camera will capture many images continuous in a short time, it will recognize that the user does that gesture many times and it will do the corresponding motion many times. For this problem, we set a flag to control it. If the user makes a gesture and it is valid, it will check that whether the action is the first time performs. If it is the first time, it will do the corresponding motion and told the flag that it had been done. If it is not the first time, it will ignore the request until there is another action which is the first time.

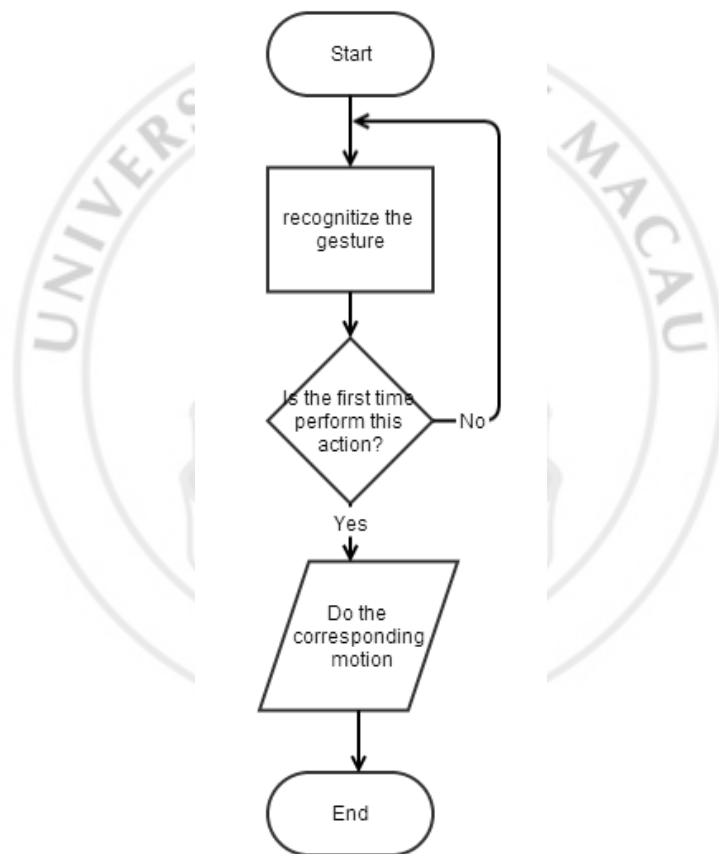


Figure 54: Control computer flowchart

4.4.1 Open Application

In C++, it has two main ways to run a program, which are:

1. `system(path)`
2. `ShellExecute(NULL, "open", path, NULL, NULL, SW_SHOWNORMAL) [16]`

The first method is to run the program as doing in cmd. However, the path is absolute path, which has to know the location of the application executable file. It cannot open the file if the path is missing or wrong.

Therefore, the second method can call the defined software based on that computer setting. For example, to open a link in a browser, we just have to provide the link only; the browser is selected by that computer.

In this interface, Game is installed in the setup file, so that is can use the first method to open. However, the other applications should open based on that computer setting, so that they have to use the second method.

4.4.2 Open On-Screen Keyboard

We want there is a keyboard that can let user enter words by clicking in iFinger. At first, I do it as before using `system()` and `ShellExecute()`, but the On-Screen keyboard is an always on top system, it makes me have some trouble to close it. After that we search the website and find this code can open the On-Screen keyboard and close it. When the user is using iFinger and want to close the On-Screen keyboard, he need to do Number 10 gesture and move down. [1, 2, 3, 4, 5] [6]

Open On-Screen keyboard:[18]

```
Wow64DisableWow64FsRedirection(FALSE);
shellExInfo.cbSize = sizeof(SHELLEXECUTEINFO);
shellExInfo.fMask = SEE_MASK_NOCLOSEPROCESS;
shellExInfo.hwnd = NULL;
shellExInfo.lpVerb = L"open";
shellExInfo.lpFile = L"C:\\Windows\\System32\\osk.exe";
shellExInfo.lpParameters = NULL;
shellExInfo.lpDirectory = NULL;
shellExInfo.nShow = SW_SHOW;
shellExInfo.hInstApp = NULL;
ShellExecuteEx(&shellExInfo); // start process
GetProcessId(shellExInfo.hProcess); // retrieve PID
oskOpen = true;
srcY = window.bottom/2-srcHeight/2-30;
locationY = srcY;
cvMoveWindow("Webcam", srcX, srcY);
```

Close On-Screen keyboard:

```
TerminateProcess(shellExInfo.hProcess, 1);
CloseHandle(shellExInfo.hProcess);
```

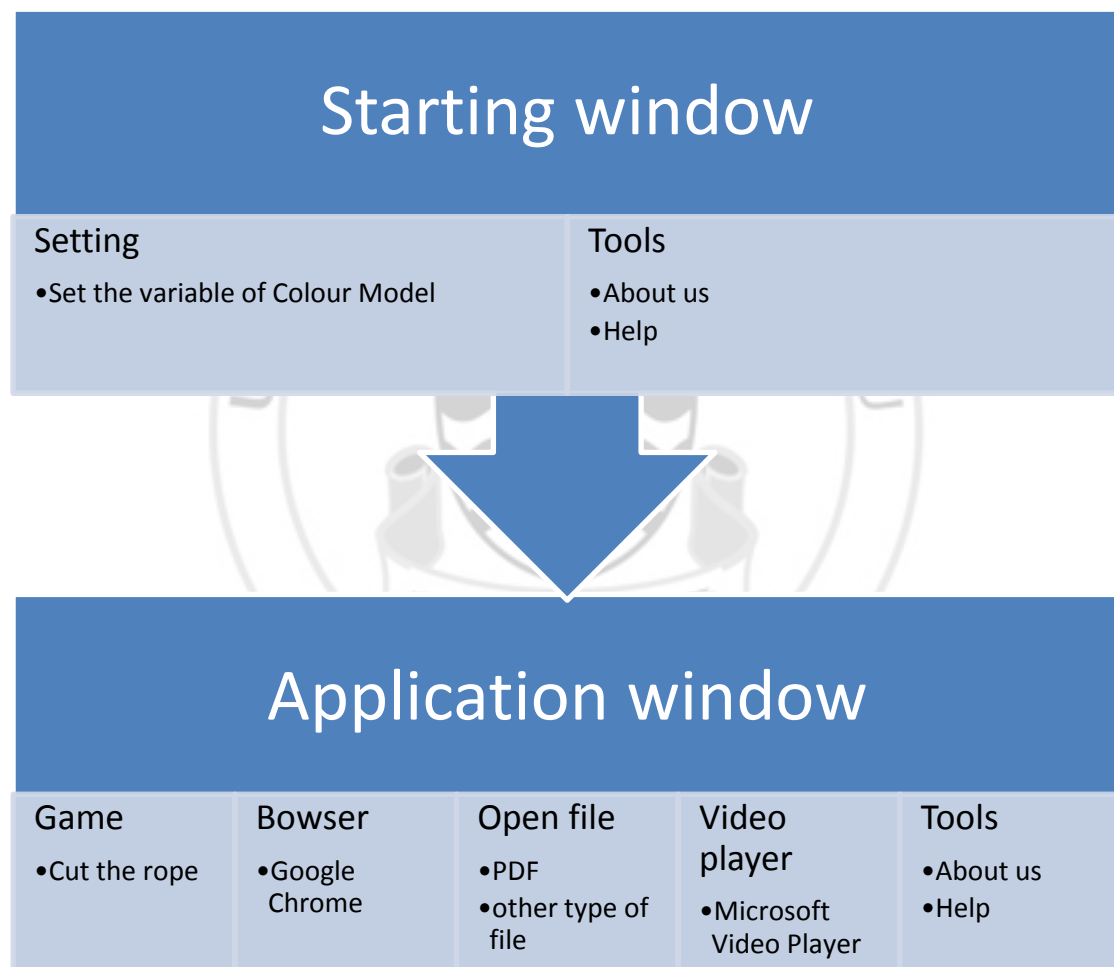
4.5 Interface of iFinger

We design an interface for iFinger because we want user can have an interesting experience for using the system. There are some pop up dialogs here, so that I define them to individual classes that will be easy to manage.[15], [19]

At first, there is a start-up screen and it will show the starting window five seconds later.

```
CstartingDlg startingDlg = new CstartingDlg;
startingDlg.Create(IDD_STARTING_DIALOG);
startingDlg.ShowWindow(SW_SHOW);
Sleep(5000);
startingDlg.ShowWindow(SW_HIDE);
```

The following is the main structure about the interface.



4.6 Setting

In the setting dialog, we can see an image capture by the camera, the system will terminate when there is no camera here, so that we need to check the camera whether connect the computer successfully. If it cannot detect any camera here, it will have a pop up message to tell the user and then it will close the system. If there is a camera here, it can run the system normally. After that, it will get the YCbCr value by reading file, and then set it to the slider.

```
CString string;
CStdioFile input("YCbCr.txt", CFile::modeRead);
input.ReadString(string);
YMin=_ttoi(string.GetString());
YMinSlider.SetRange(0,255);
YMinSlider.SetPos(YMin);
```

The sliders also do not have transparent background in Microsoft Visual Studio, so I also need to rebuild another class to change it. Here is a class “CTransparentSlider” which inherit the default class “CSliderCtrl”, and then we have searched the website and it can help me to solve the problem.[20]

```
void CTransparentSlider::OnCustomDraw(NMHDR* pNMHDR, LRESULT* pResult)
{
    LPNMCUSTOMDRAW lpcd = (LPNMCUSTOMDRAW)pNMHDR;
    CDC *pDC = CDC::FromHandle(lpdc->hdc);
    switch(lpcd->dwDrawStage)
    {
        case CDDS_PREPAINT:
            *pResult = CDRF_NOTIFYITEMDRAW;
            break;
        case CDDS_ITEMPREPAINT:
            if (lpcd->dwItemSpec == TBCD_THUMB)
            {
                *pResult = CDRF_DODEFAULT;
                break;
            }
            if (lpcd->dwItemSpec == TBCD_CHANNEL)
            {
                CClientDC clientDC(GetParent());
                CRect crect;
                CRect wrect;
                GetClientRect(crect);
                GetWindowRect(wrect);
                GetParent()->ScreenToClient(wrect);
                if (m_dcBk.m_hDC == NULL)
                {
                    m_dcBk.CreateCompatibleDC(&clientDC);
                    m_bmpBk.CreateCompatibleBitmap(&clientDC,
                        crect.Width(), crect.Height());
                    m_bmpBkOld = m_dcBk.SelectObject(&m_bmpBk);
                    m_dcBk.BitBlt(0, 0, crect.Width(), crect.Height(),
                        &clientDC, wrect.left, wrect.top, SRCCOPY);
                }
                //This bit does the tics marks transparently.
                CDC SaveCDC;
                CBitmap SaveCBmp, maskBitmap;
                //set the colours for the monochrome mask bitmap
                COLORREF crOldBack = pDC->SetBkColor(RGB(0,0,0));
                COLORREF crOldText = pDC->SetTextColor(RGB(255,255,255));
                CDC maskDC;
                int iWidth = crect.Width();
                int iHeight = crect.Height();
                SaveCDC.CreateCompatibleDC(pDC);
                SaveCBmp.CreateCompatibleBitmap(&SaveCDC, iWidth, iHeight);
                CBitmap* SaveCBmpOld = (CBitmap
                *)SaveCDC.SelectObject(SaveCBmp);
                //fill in the memory dc for the mask
                maskDC.CreateCompatibleDC(&SaveCDC);
                //create a monochrome bitmap
                maskBitmap.CreateBitmap(iWidth, iHeight, 1, 1, NULL);
                //select the mask bitmap into the dc
```

```

        CBitmap* OldmaskBitmap = maskDC.SelectObject(&maskBitmap);
        //copy the oldbitmap data into the bitmap, this includes the
tics.
        SaveCDC.BitBlt(0, 0, iWidth, iHeight, pDC, crect.left,
crect.top, SRCCOPY);
        //now copy the background into the slider
        BitBlt(lpdc->hdc, 0, 0, iWidth, iHeight, m_dcBk.m_hDC, 0, 0,
SRCCOPY);
        maskDC.BitBlt(0, 0, iWidth, iHeight, &SaveCDC, 0, 0,
SRCCOPY);
        pDC->BitBlt(0, 0, iWidth, iHeight, &SaveCDC, 0, 0,
SRCINVERT);
        pDC->BitBlt(0, 0, iWidth, iHeight, &maskDC, 0, 0, SRCAND);
        pDC->BitBlt(0, 0, iWidth, iHeight, &SaveCDC, 0, 0,
SRCINVERT);

        //restore and clean up
        pDC->SetBkColor(crOldBack);
        pDC->SetTextColor(crOldText);
        DeleteObject(SelectObject(SaveCDC, SaveCBmpOld));
        DeleteDC(SaveCDC);
        DeleteObject(maskDC.SelectObject(OldmaskBitmap));
        DeleteDC(maskDC);
        *pResult = 0;
        break;
    }
}

```



CHAPTER 5. TESTING AND EVALUATION

The quality of hand detection result is relative to the light color, light reflection, background noise and shadow. Therefore, we will test these features one by one. Finally, we will collect some user responses and summarize the accuracy.

5.1 Normal Environments

In order to have a better result, it may need a clear background and stable light source. For the setting of the camera, it should turn off auto force and white balance, which will affect the result.

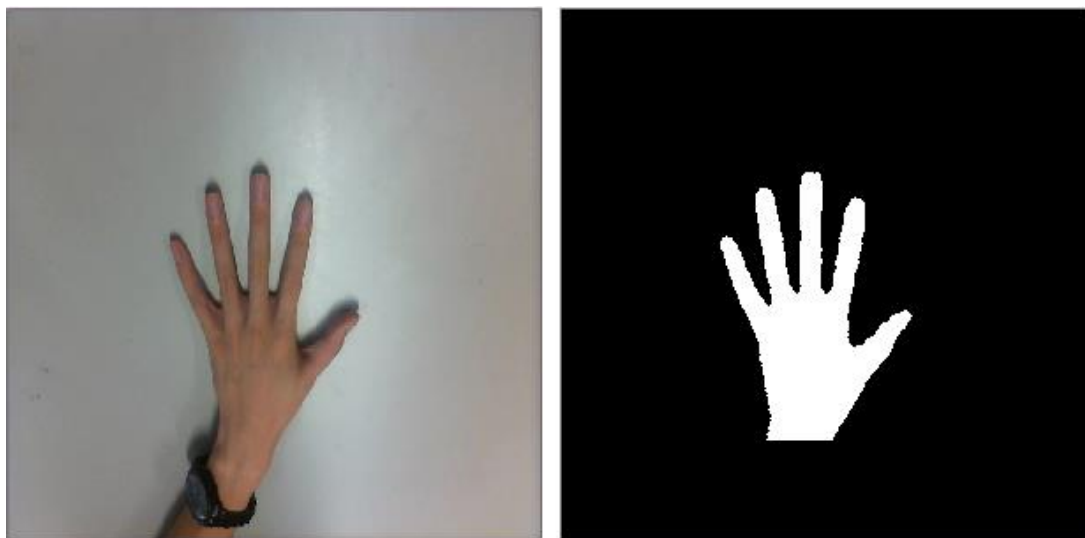


Figure 55: Normal environments

5.2 Special Environment

Different environments have different effects to the recognition. The following will show the result based on the same setting, but in different environments.

5.2.1 Very Strong Light Source

When the light source is very strong, the color of the skin will be different. Let me emphasize one point which cannot fix by adjusting the color model variable since the light source makes the skin color change. It is not similar to the dark environment, which can fix by modifying variable.

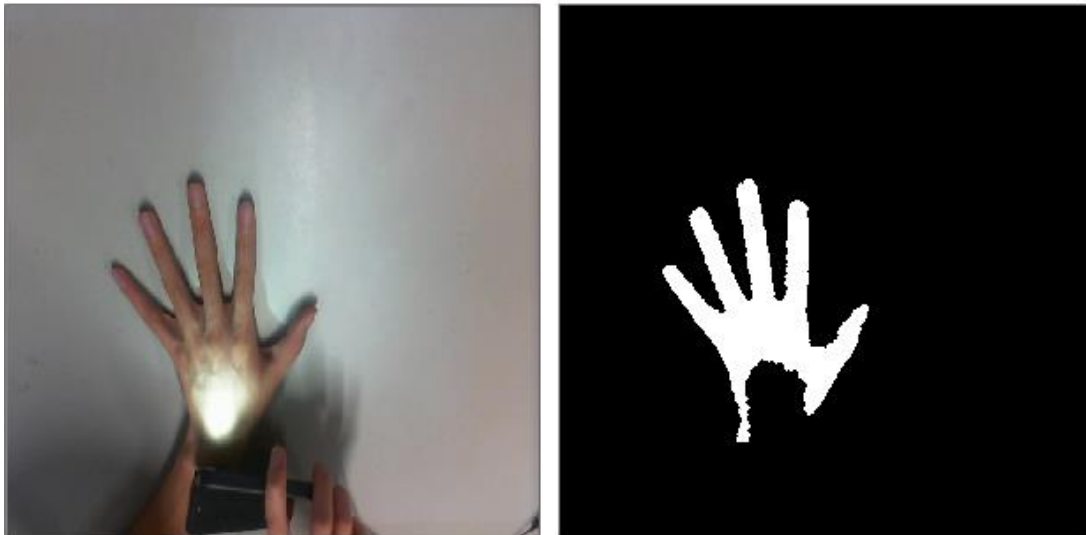


Figure 56: Strong light source

5.2.2 Complex Background

The result is poor when the background has some objects that similar to hand color. For example, the brown color table is not a good choice. Besides background reason, if the environment has a yellow light source, which can render the background objects in little yellow color. Therefore, the performance will also be decreased.



Figure 57: Complex background

5.3 Processing Time Testing

We can calculate the processing time for each loop. The data show that it will use 0.07 seconds for one hand image, which means it can process 14.3 frames in one second. For two hands image, it will use 0.095 seconds for one image, which can process 10.5 frames in one second.

5.4 Testing Gestures

First of all, let me recall the gestures table Table 2: Gesture table, which in the Chapter 3.

5.4.1 Testing Accuracy

In our testing, we will let the users perform each gestures 20 times, and static the accuracy for each gesture. There are total 14 different gestures and 5 users are included in this testing.

No.	User 1	User 2	User 3	User 4	User 5	Total	Accuracy
1	20/20	20/20	20/20	20/20	20/20	100/100	100%
2	20/20	19/20	19/20	18/20	19/20	95/100	95%
3	18/20	20/20	17/20	20/20	18/20	93/100	93%
4	20/20	20/20	18/20	19/20	17/20	94/100	94%
5	19/20	18/20	18/20	18/20	18/20	91/100	91%
6	18/20	18/20	17/20	19/20	18/20	90/100	90%
7	20/20	20/20	20/20	19/20	20/20	99/100	99%
8	18/20	19/20	16/20	18/20	18/20	89/100	89%
9	18/20	17/20	18/20	16/20	17/20	86/100	86%
10	15/20	16/20	17/20	16/20	17/20	81/100	81%
11	18/20	19/20	19/20	18/20	16/20	90/100	90%
12	17/20	18/20	15/20	16/20	16/20	82/100	82%
13	20/20	19/20	18/20	17/20	18/20	92/100	92%
14	17/20	19/20	18/20	17/20	19/20	90/100	90%

Table 7: Testing accuracy

After doing the testing, we ask some question for the user. They can give 5 marks as the highest mark if satisfy the function.

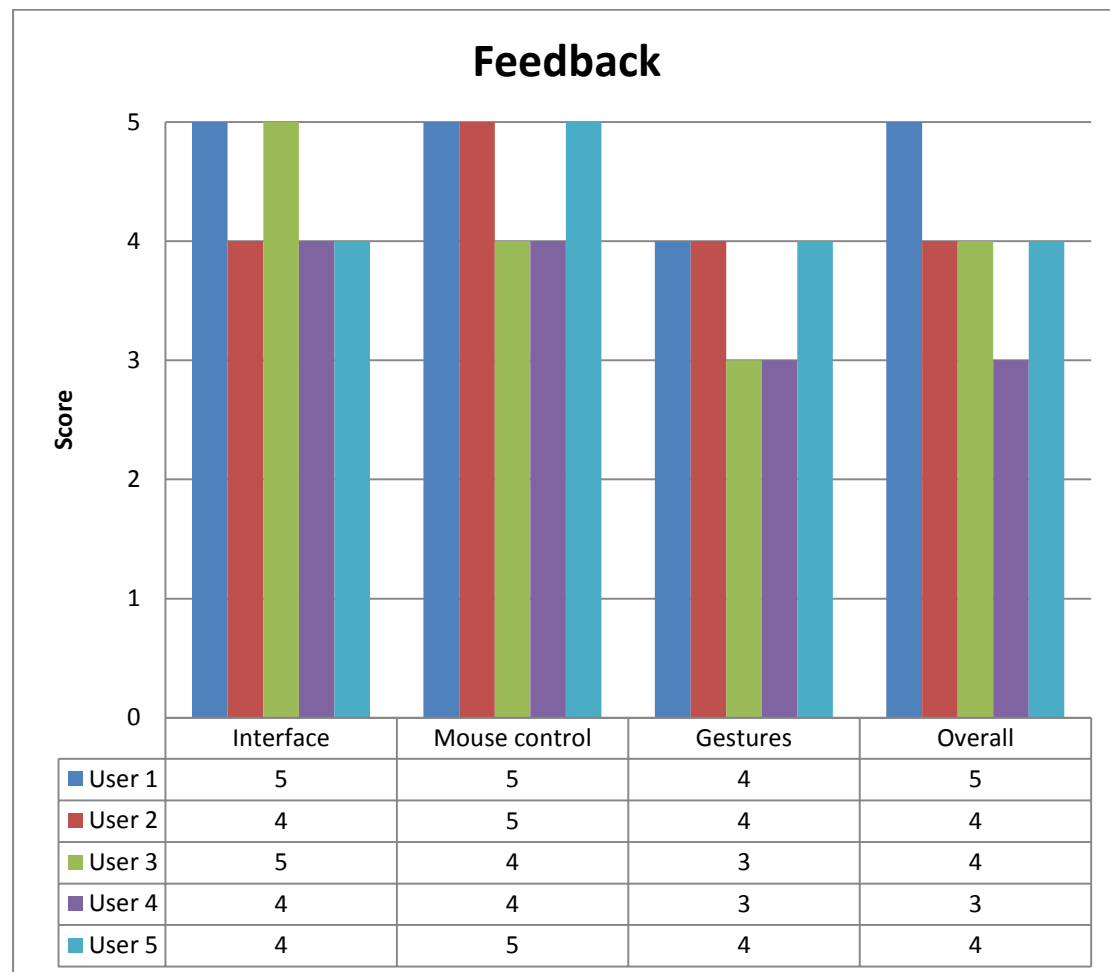


Table 8: Feedback

5.5 Evaluation

We had summery the objective and the criteria in Chapter 1, and we can compare with testing results in the following.

No.	Objective	Criteria	Testing result
1	Skin extraction	The hand shape can extract correctly	Done
2	Processing time	Processing time for each image should lower than 0.150 seconds	The average processing is 0.0825 seconds
3	Mouse action	Perform the mouse action, such as left click, right click, middle click and move	Done
4	Virtual keyboard	Allow user insert characters, though their fingers	Done
5	Two hand gesture recognition	The user can use two hands to perform actions	Done
6	Accuracy	The average accuracy of detection should higher than 90%	The average accuracy is 90.86% for all gesture
7	Interface	A user-friendly interface, the user has no doubt for the function of the button	The average score is 4.4 (highest: 5)
8	User experience	The users think it is great and easy to use, which will evaluate by users	The average score is 4.0 (highest: 5)

Table 9: Evaluation

In the above table, we can see most of objectives are satisfied with the criteria. The processing time is fast so that it can perform real-time processing. Besides, the performance for mouse control is very nice.

CHAPTER 6. DISCUSSION

For the Vision-Based HCI, the difficulties of recognition are how to balance the processing time and quality, stability and resolution. [1] It means increased quality cause processing time raise, or decrease processing time cause quality drop. Same as stability and resolution, it has to make a choice to balance the result.

6.1 Satisfied Objectives

We focus on the processing speed and stability in this project, which can let the user perform any operation easily. We conclude some objectives that we feel satisfied to our objectives, such as Processing time, Mouse action, Virtual keyboard, Two hand gesture recognition and Interface

First of all, indeed, the program has a very good processing time, which about 0.0825 second/frame. When there is single hand gesture, it only cost about 0.07 second/frame. It is enough to perform real time processing.

The following, the second and the third, the user can perform mouse and keyboard action with their finger. In addition, the shaking problem is almost solved, which increase the stability a lot. Therefore, the user can control them to finish some simple tasks.

The fourth, this program support two hands gesture recognition. The users can perform the action when they use specific gesture.

The last, the interface is user-friendly, which is given by the user. Most of them can know the function of the button without reading the text. Besides, all buttons in the interface have a feedback operation, which will enlarge the button when pointing to it, etc.

6.2 Unsatisfied Objectives

“You cannot sell the cow and drink the milk.” it means to optimize something, it will affect other factor. The processing time is fast, which cause the quality drop down a little; stable the fingertip, the resolution will decrease, etc.

The following objectives, we think that we can be improved more: Skin extraction, Accuracy and User experience.

From the result of testing, the system is needed to adjust the color model in order to adapt different light sources and backgrounds. Both of them are the mainly effect to our detection since the shaking problem will appear again when hand sharp is not clear.

The other objective, we don't satisfied is the accuracy. It is because the moving directions will affect the performance. Moving left or right has better performance than moving up or down. It is because the space moving left or right is larger; hence, the accuracy is higher. Furthermore, the user will turn their hand in different angle, which may cause unexpected control.

The last, user experience is not so good compare to our expectation when using this system. The main reason is that it does not have the gesture setting for the specific user. The action of the gesture is set by us, but it is difficult to adapt to different users. Another reason may be the accuracy, it still occurs some unexpected controls that will barriers the user further action to the computer.

6.3 Future Work

To be honest, the skin color model is not enough to extract the hand sharp absolutely, which need to adjust the color range when using in a new environment. The result given by the color model is affected by the lighting source, the environment and shadow. We are going to find out the combination of methodology to produce a system that is more stable, can adjust the color model automatically, so that it can be used in the different environment.

For the accuracy problem, we should optimize the detection method also in the future. It still does not have good results for finding palm center and cutting arm. Besides, we should add an algorithm to turn the user's hand into correct angle. In the future, we will find out the suitable methodology to solve those problems.

In order to make our program easier to use, we are going to add gestures setting in the future. It will give a chance to the users to set gestures according to their behavior. However, it is a time consuming task to format all gestures and allow setting by the user.

CHAPTER 7. CONCLUSIONS

7.1 Overall

In short, this project allows users to use their gestures to control computer, it can apply to many applications with their provided shortcut. It looks like a simple task, but it is hard to make it perfectly since the stability and performance are hard to balance.

In this project, we used a common camera with normal resolution. Therefore, this technique can apply into many different places with a cheap price.

The system support 2-hand gestures and moving gestures, which can maximize the usability of gesture. In addition, we add some smoothing technique to make the result become more stable. Therefore, we can apply the gesture into an application to make them easy to use.

However, we had met some problems that have not solved in this report. We will pass them into future work. The unsolved problems are:

1. Auto adjusts skin color
2. Complex background

Those problems are related to the hand sharp recognition. We can apply it in different environments when we solve those problems. However, it is very hard to solve those problems.

7.2 Acquisition

During this project, we figure out how to do a qualify project. We begin from research, problem space specification, analyze methodology, management, work distribution, user-friendly interface design, testing, and finally finish the project. We have learnt so much in this progress, study others research, find out algorithms, raise problems and solve it again and again.

Actually, those progresses cannot finish by us only; also supervisors give us many useful suggestions in this project. Otherwise, we just finish a meaningless program, but not a useful project.

After finishing this project, we know that it is not the end of our study, but it is the last chance to study in university. However, we use all project relative knowledge that we have learnt in university. We are deeply grateful for all doctors, professors and assistants. We will try our best and overcome all barriers in the future.

CHAPTER 8. REFERENCES

- [1] F. Karray, M. Alemzadeh, J. A. Saleh, and M. N. Arab, “Human-computer interaction: Overview on state of the art,” 2008.
- [2] B. A. Myers, “A brief history of human-computer interaction technology,” *interactions*, vol. 5, no. 2, pp. 44–54, 1998.
- [3] T. Leyvand, C. Meekhof, Y.-C. Wei, J. Sun, B. Guo, and others, “Kinect identity: Technology and experience,” *Computer*, vol. 44, no. 4, pp. 94–96, 2011.
- [4] H. Hodson, “Leap Motion hacks show potential of new gesture tech,” *New Scientist*, vol. 218, no. 2911, p. 21, 2013.
- [5] Phys.org, “MYO armband to muscle into computer control,” *Phys.org*, p. 2, 2014.
- [6] B. Bonnechere, B. Jansen, P. Salvia, H. Bouzahouene, L. Omelina, J. Cornelis, M. Rooze, and S. Van Sint Jan, “What are the current limits of the Kinect sensor,” in *Proc 9th Intl Conf. Disability, Virtual Reality & Associated Technologies, Laval, France*, 2012, pp. 287–294.
- [7] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, and J. Sodnik, “An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking,” *Sensors*, vol. 14, no. 2, pp. 3702–3720, 2014.
- [8] “LifeCam Cinema Features.” [Online]. Available: <http://www.microsoft.com/hardware/en-us/p/lifecam-cinema#details>.
- [9] M. G. Jacob, Y.-T. Li, G. A. Akingba, and J. P. Wachs, “Collaboration with a robotic scrub nurse,” *Communications of the ACM*, vol. 56, no. 5, pp. 68–75, 2013.
- [10] S. Brown, “Very Cool: Fujitsu Creates Technology That Allows Anything to Be a Touchscreen,” *Science*, 2013.
- [11] 張廷瑜, “手勢數字碼辨識,” 崑山科技大學電機工程研究所學位論文, pp. 1–59, 2009.
- [12] C.-W. Chang and C.-H. Chang, “A two-hand multi-point gesture recognition system based on adaptive skin color model,” in *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, 2011, pp. 2901–2904.
- [13] *Visual Studio C++ 2010-MFC编程入门*.
- [14] “MFC setting image background for dialog.” [Online]. Available: <http://cppandmfc.blogspot.com/2013/03/mfc-setting-image-background-for->

dialog.html.

- [15] “Windows Media Player shortcuts.” [Online]. Available:
<http://windows.microsoft.com/zh-tw/windows/media-player-keyboard-shortcuts#1TC=windows-7> .
- [16] “ShellExecute function.” [Online]. Available: [http://msdn.microsoft.com/en-us/library/windows/desktop/bb762153\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/bb762153(v=vs.85).aspx).
- [17] “Transparent Windows.” [Online]. Available:
<https://www.youtube.com/watch?v=XU7ZI6SnnC4>.
- [18] “kill process started with shellexecuteex.” [Online]. Available:
<http://stackoverflow.com/questions/13193719/kill-process-started-with-shellexecuteex> .
- [19] “Set Video in MFC Dialog.” [Online]. Available:
<http://forrobot.blogspot.com/2012/08/opening-imageaviwebcam-on-mfc-dialog.html>.
- [20] “MFC Transparent Scrollbar.” [Online]. Available:
http://blog.7cdi.com/aaaa/showtopic_126.html.

