



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 I of II**

*by*

**Chi Hong, Ao, Student No: D-B0-2838-1**

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:

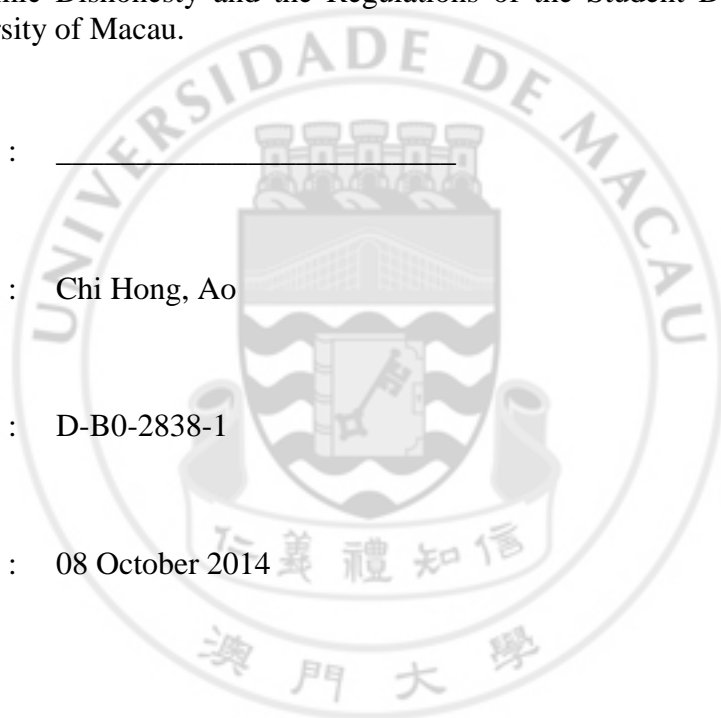
1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts 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 Hong, Ao

Student No. : D-B0-2838-1

Date : 08 October 2014



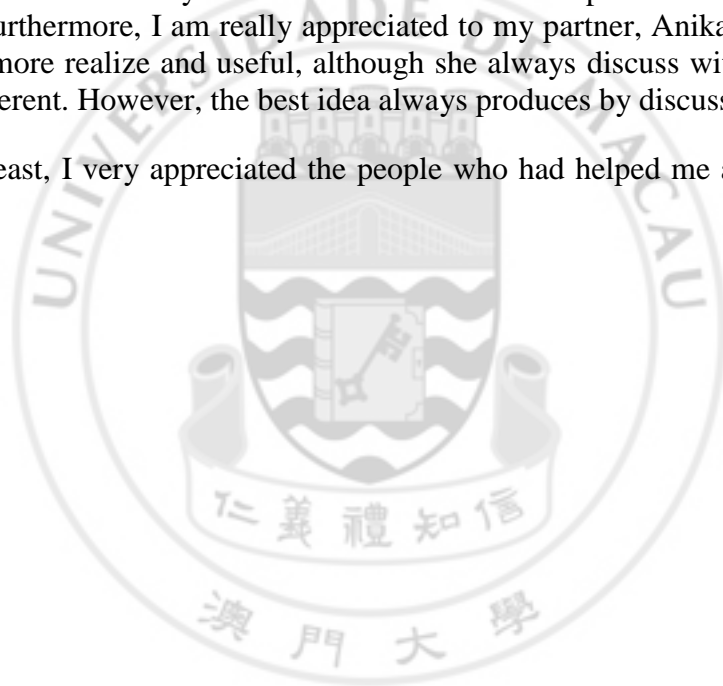
## ACKNOWLEDGEMENTS

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 always provide many useful suggestions to make the project become more professional. Also, they will indicate the direction how to research, study and develop a software system systematically and to be meaningful.

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, I am really appreciated to my partner, Anika. She made my idea become more realize and useful, although she always discuss with me when the opinion is different. However, the best idea always produces by discussion.

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

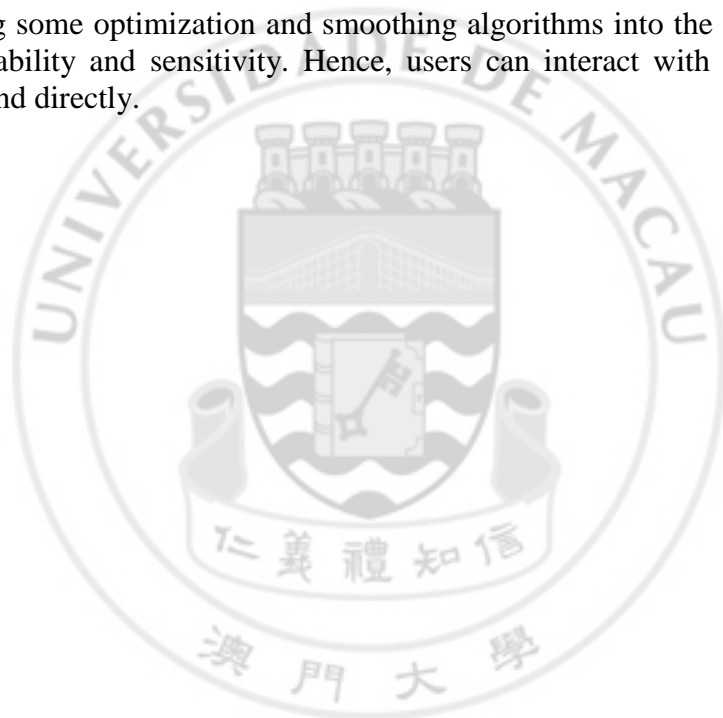


## ABSTRACT

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

<b>CHAPTER 1. INTRODUCTION</b>	<b>11</b>
<b>1.1 Overview</b>	<b>11</b>
1.1.1 HCI History	11
1.1.2 Our Future Life	11
<b>1.2 Gesture Recognition Devices</b>	<b>12</b>
1.2.1 Kinect	12
1.2.2 Leap Motion	12
1.2.3 MYO Armband	13
1.2.4 Overall	13
<b>1.3 Objectives</b>	<b>13</b>
1.3.1 System Objectives and Motivation	14
1.3.2 System Environment	14
1.3.3 Devices Setup	14
<b>1.4 Summary of Workload</b>	<b>15</b>
1.4.1 Project Scope	15
1.4.2 Development Process	16
1.4.3 Work Distribution	18
<b>1.5 Technologies Description</b>	<b>19</b>
1.5.1 Difficult During Development	20
<b>CHAPTER 2. RELATED WORK</b>	<b>21</b>
<b>2.1 Collaboration with a Robotic Scrub Nurse</b>	<b>21</b>
2.1.1 Background	21
2.1.2 Gesture Recognition	21
2.1.3 Experiment Result	22
<b>2.2 Turns Paper into a Touchscreen</b>	<b>23</b>
2.2.1 Gesture Recognition	23
2.2.2 Other Functions	24
<b>2.3 Gesture recognition for digits and characters</b>	<b>24</b>
2.3.1 Background	24
2.3.2 Hand Detection	24
2.3.3 Fingertips Detection	24
2.3.4 Gesture Recognition	25
2.3.5 Experiment Result	25
<b>2.4 A Two-Hand Multi-Point Gesture Recognition System Based on Adaptive Skin Color Model</b>	<b>25</b>
2.4.1 Object Detection	25
2.4.2 Object Segmentation and Tracking	25
2.4.3 Features Extraction	25
2.4.4 Gesture Recognition	26
2.4.5 Experiment Result	26

<b>CHAPTER 3.</b>	<b>FINGER RECOGNITION</b>	<b>27</b>
<b>3.1</b>	<b>Human Skin Extraction</b>	<b>27</b>
3.1.1	HSV Color Model	27
3.1.2	YCbCr Color Model	27
3.1.3	Background Subtraction	28
3.1.4	Skin Color Filter	29
<b>3.2</b>	<b>Noise Reduction</b>	<b>30</b>
3.2.1	Erosion and Dilation	30
3.2.2	Opening and Closing	30
3.2.3	Reduce Noise	31
<b>3.3</b>	<b>Extract Features</b>	<b>31</b>
3.3.1	Hand Features	32
<b>3.4</b>	<b>Cut Hand</b>	<b>33</b>
3.4.1	Palm Center Extraction	33
3.4.2	Hand Defects	34
3.4.3	Fingertip Extraction	35
<b>3.5</b>	<b>Optimize Fingertips Position</b>	<b>36</b>
3.5.1	Anti-shaking Algorithm	36
3.5.2	Smooth Mouse Moving Algorithm	37
<b>CHAPTER 4.</b>	<b>IMPLEMENTATION</b>	<b>38</b>
<b>4.1</b>	<b>Color Filter and Hand Shape Extraction</b>	<b>38</b>
<b>4.2</b>	<b>Hand Features Extraction</b>	<b>39</b>
4.2.1	Hand shape	39
4.2.2	Extract Palm Center	39
4.2.3	Cut hand	41
4.2.4	Hand defects	42
4.2.5	Fingertips	43
<b>4.3</b>	<b>Optimization</b>	<b>44</b>
4.3.1	Anti-shaking Algorithm	45
4.3.2	Smooth Mouse Moving Algorithm	45
<b>CHAPTER 5.</b>	<b>TESTING AND EVALUATION</b>	<b>47</b>
<b>5.1</b>	<b>Normal Environments</b>	<b>47</b>
<b>5.2</b>	<b>Special Environment</b>	<b>47</b>
5.2.1	Very Strong Light Source	48
5.2.2	Complex Background	48
<b>5.3</b>	<b>Processing Time Testing</b>	<b>49</b>
<b>5.4</b>	<b>Testing Gestures</b>	<b>49</b>
5.4.1	Testing Accuracy	51
<b>5.5</b>	<b>Evaluation</b>	<b>53</b>

<b>CHAPTER 6. DISCUSSION</b>	<b>54</b>
6.1 Satisfied Objectives	54
6.2 Unsatisfied Objectives	54
6.3 Future Work	55
<b>CHAPTER 7. CONCLUSIONS</b>	<b>56</b>
7.1 Overall	56
7.2 Acquisition	56
<b>CHAPTER 8. REFERENCES</b>	<b>57</b>

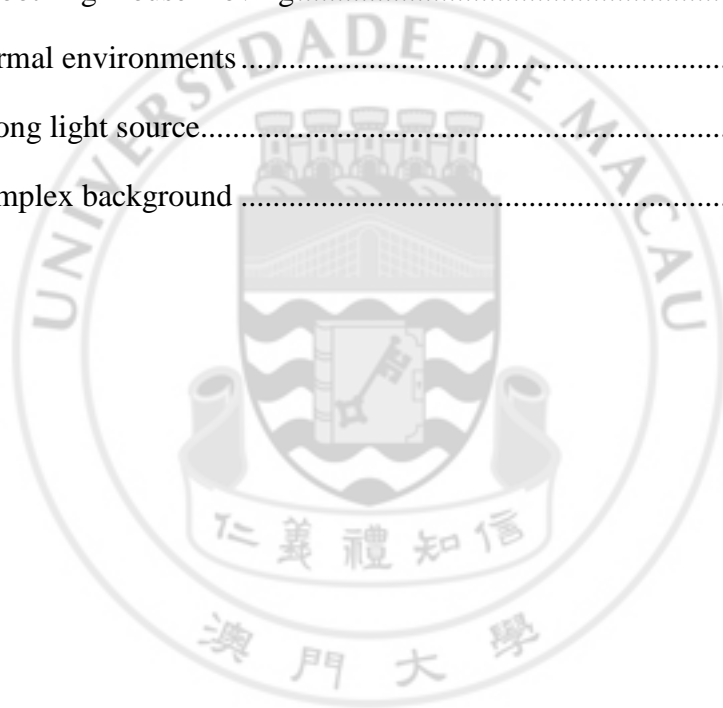




## LIST OF FIGURES

Figure 1: Sci-fi movie - Iron man 2 .....	11
Figure 2: Gesture recognition devices .....	12
Figure 3: Leap Motion valid area.....	12
Figure 4: MYO Armband.....	13
Figure 5: Camera position and valid area to be captured.....	15
Figure 6: Microsoft LifeCam .....	15
Figure 7: Work distribution .....	18
Figure 8: Robotic Scrub Nurse .....	21
Figure 9: Gesture table for surgery .....	22
Figure 10: Testing result for ACM research .....	22
Figure 11: Using paper as touchscreen .....	23
Figure 12: Height information for fingertip.....	23
Figure 13: Mask model .....	24
Figure 14: Radar scan .....	26
Figure 15: HSV color model.....	27
Figure 16: YCbCr color model .....	28
Figure 17: Background Subtraction .....	28
Figure 18: Testing color model.....	29
Figure 19: Erosion.....	30
Figure 20: Dilation.....	30
Figure 21: Opening .....	30
Figure 22: Closing.....	31
Figure 23: Hand .....	32
Figure 24: Cut hand .....	33

Figure 25: Palm center .....	34
Figure 26: Hand defect.....	34
Figure 27: Fingertip extraction .....	35
Figure 28: Mouse smoothing ( $\lambda=2$ ) .....	37
Figure 29: Extract palm center flowchart .....	40
Figure 30: Cut hand flowchart .....	41
Figure 31: Fingertip extraction flowchart .....	43
Figure 32: Anti-shaking flowchart.....	45
Figure 33: Smoothing mouse moving.....	46
Figure 34: Normal environments .....	47
Figure 35: Strong light source.....	48
Figure 36: Complex background .....	48



## LIST OF TABLES

Table 1: Gesture table .....	50
Table 2: Testing accuracy .....	51
Table 3: Feedback .....	52
Table 4: Evaluation .....	53



## 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*

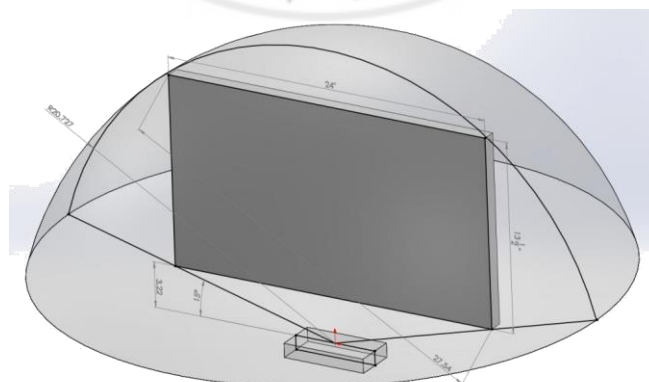
### 1.2.1 Kinect

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]

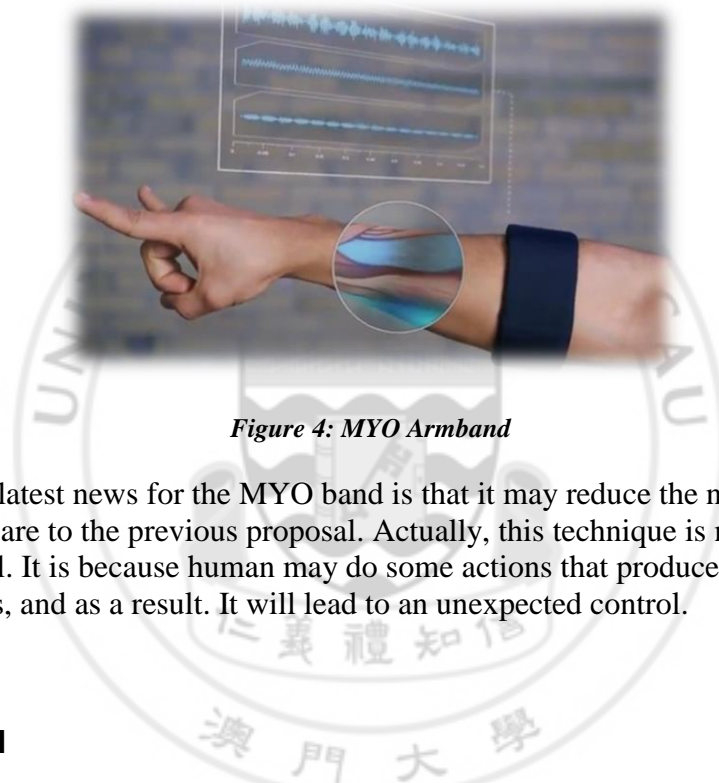


*Figure 3: Leap Motion valid area*

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.

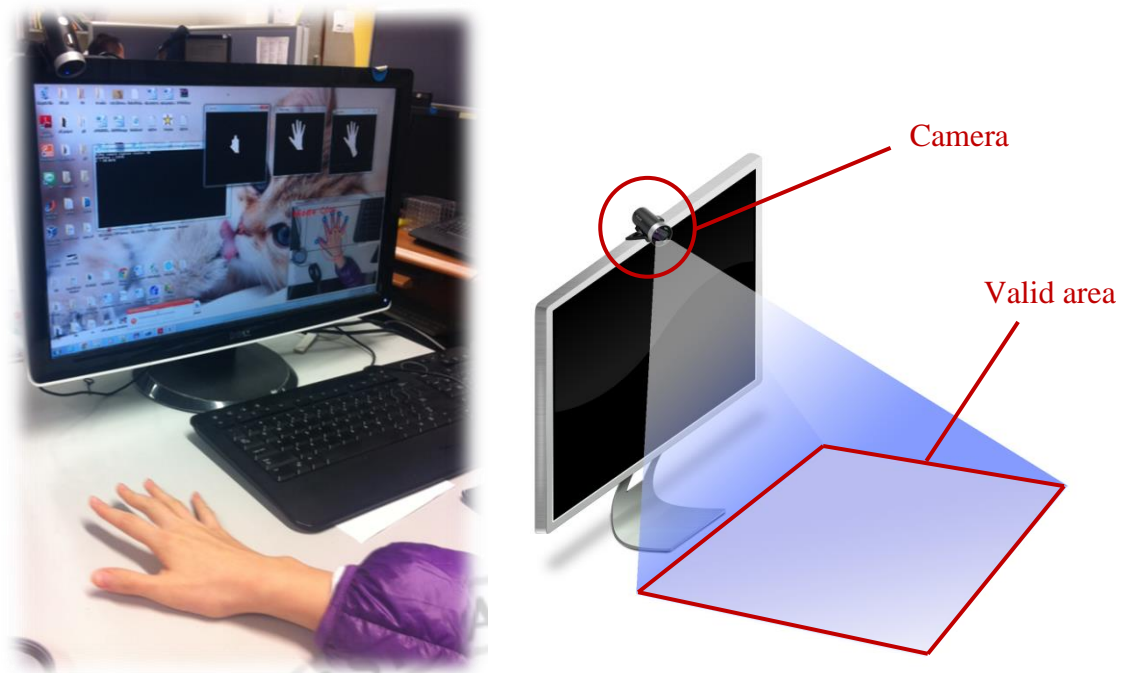
### **1.3.2 System Environment**

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 user.

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 1, 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.

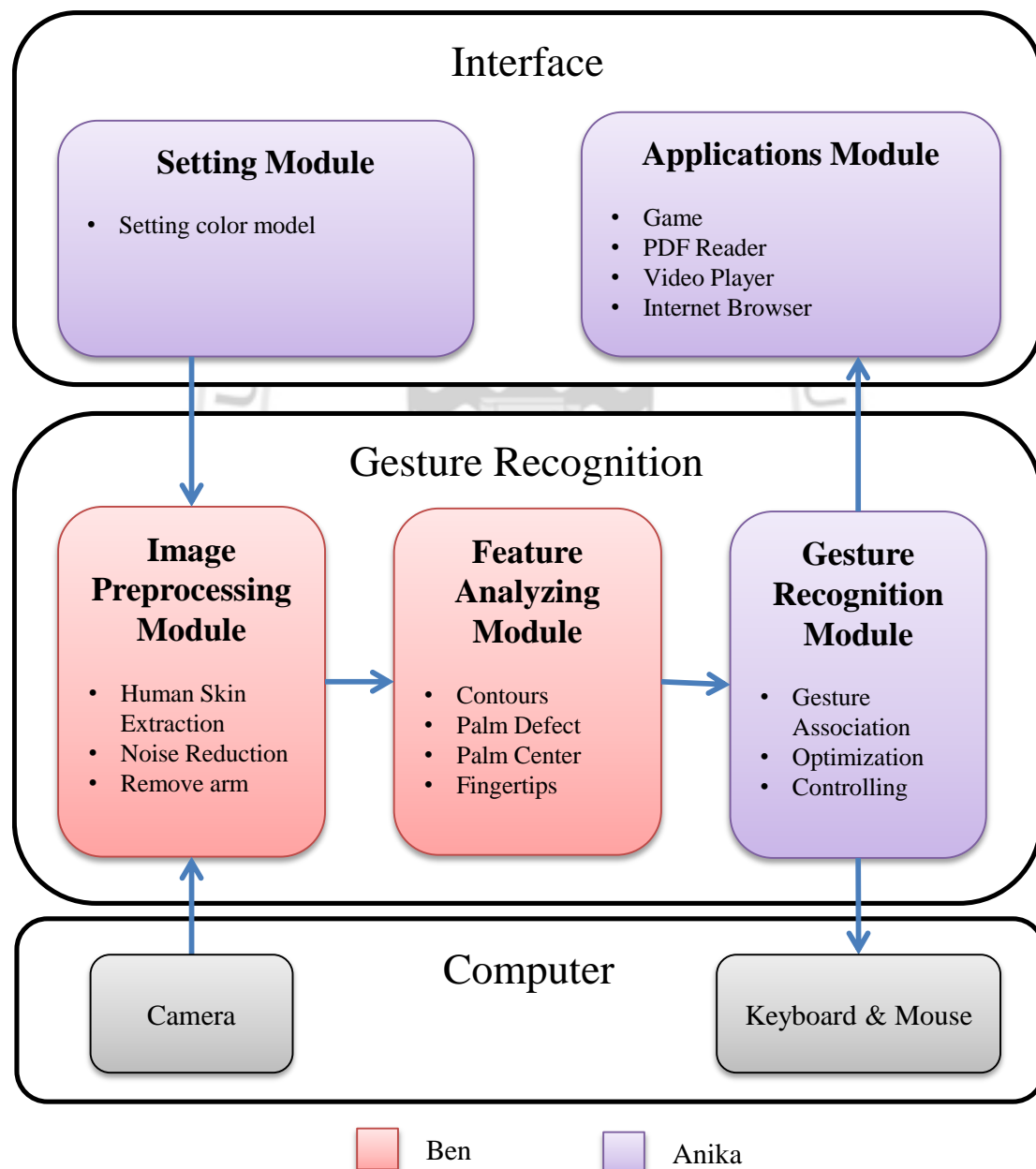
### 1.4.3 Work Distribution

#### 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

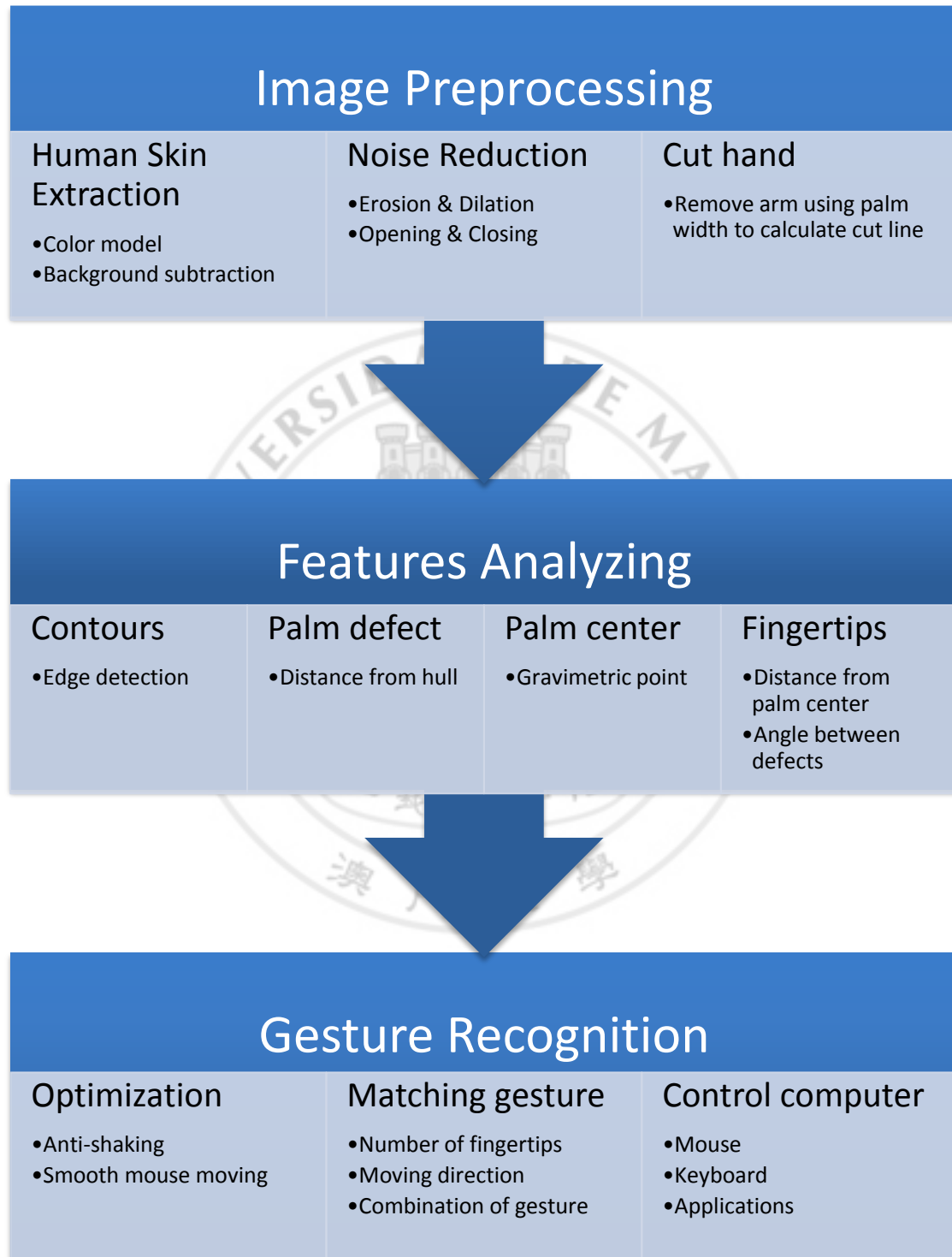
Our project included 2 members - Anika and Ben. The modules and the functions, and the work distribution are specified here:



*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

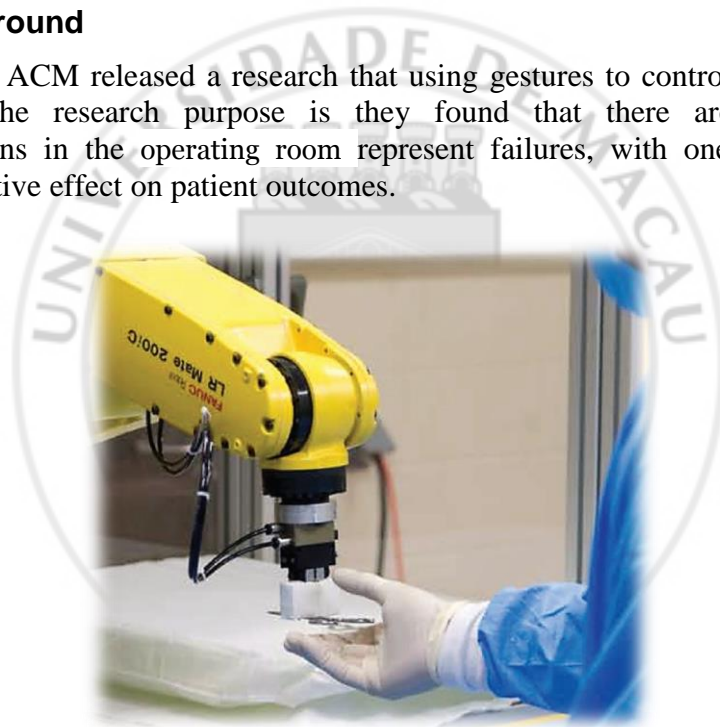
Nowadays, the gesture recognition is one of the hot topics in the world. Some famous companies are providing their system into different areas. There are also some researches released on the internet, which are about gesture recognition by using different methodologies.

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.

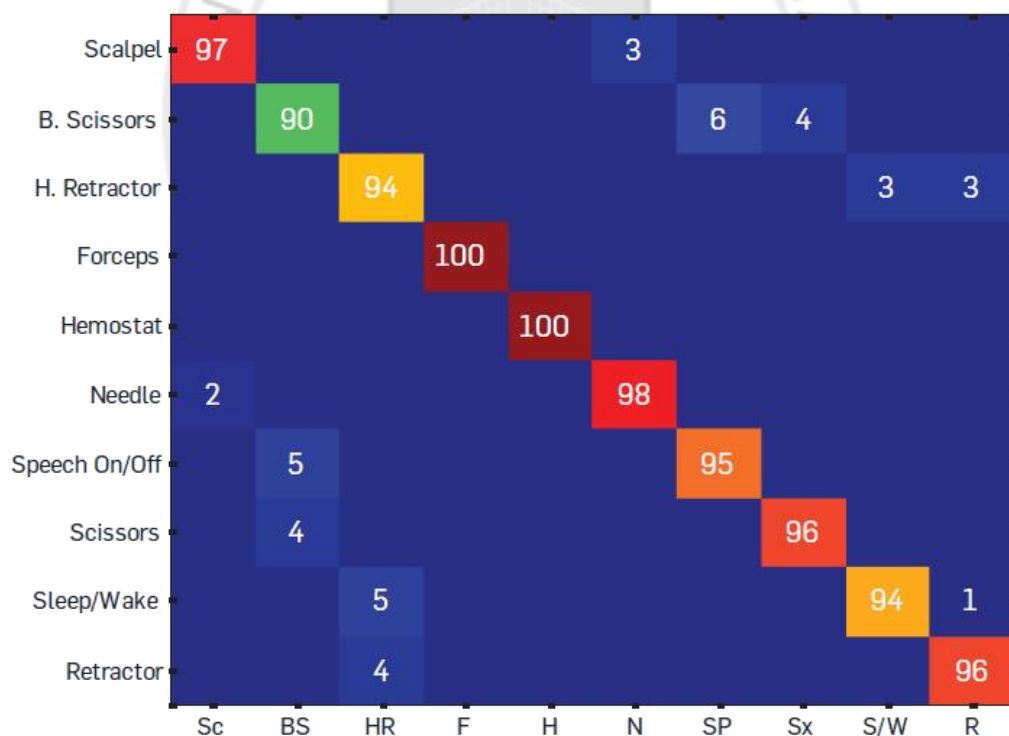
The system provided seven standard types of surgical instrument: scalpel, scissors, retractors, hemostats, clippers, forceps, and hooks. Combine to the movement, there are 5 static and 5 dynamic gestures.

 (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.

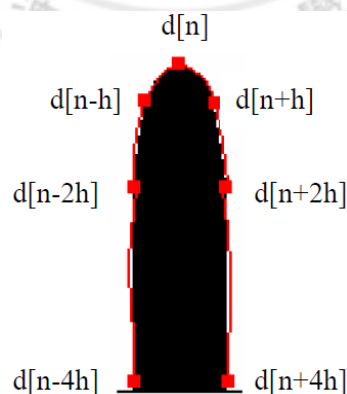
### 2.3.2 Hand Detection

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 avoids 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**

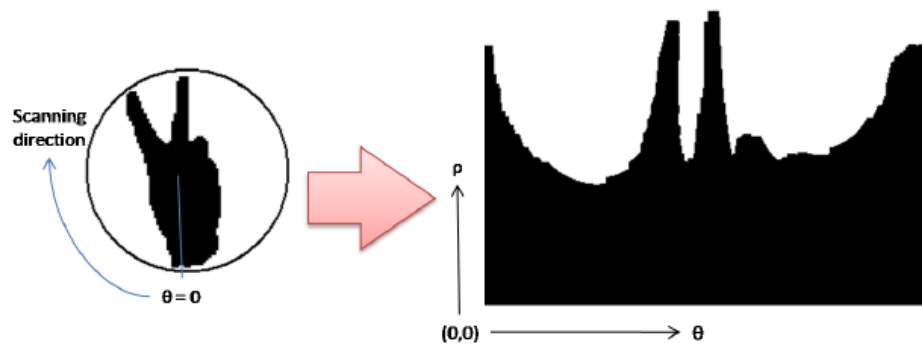
At First, it is using a camera to get an image, and then change it to a grayscale image. And then use Haar-Like Features database that had recorded some starting image. The system will find whether the grayscale image similar to the image which in the database so that it can detect the image whether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

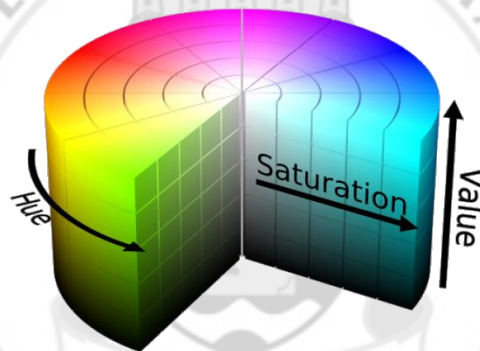
### 3.1 Human Skin Extraction

Visual-Based HCI mainly rely on the capture an image from the real world, and then extract the shape of the object from that image. [1] Main method to extract shape features is using the Color Model or Background subtraction.

There are many different color models can be used, such as: RGB, HSV, HSL, YCbCr, etc. However, the mainly color models for extract skin color are HSV and YCbCr.

#### 3.1.1 HSV Color Model

HSV can be explained as Hue, Saturation and Value. It can used to extract the skin color since HSV can set the range of Hue to our skin color very easily. [13] [14]

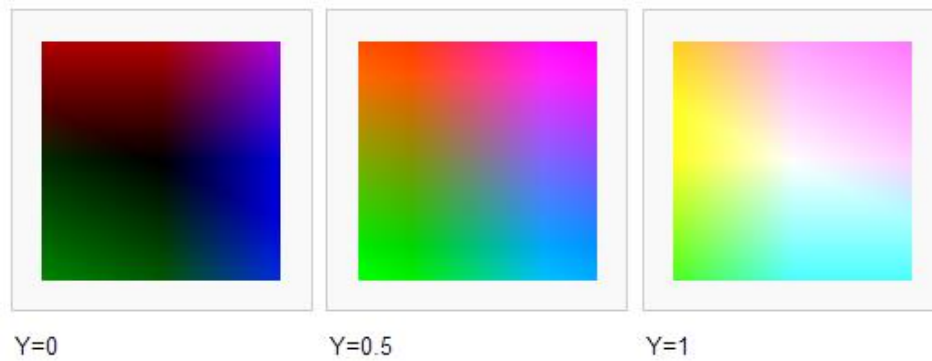


*Figure 15: HSV color model*

However, HSV has poor performance when using it in a white light source environment since HSV is too sensitive to white color. Moreover, the conversion cost is high.

#### 3.1.2 YCbCr Color Model

The other color model for human skin is YCbCr, which is a nonlinear color model. Y is stand for Luminance, Cb is stand for the Blue-difference Chroma, Cr is stand for the Red-difference Chroma. [13][15]

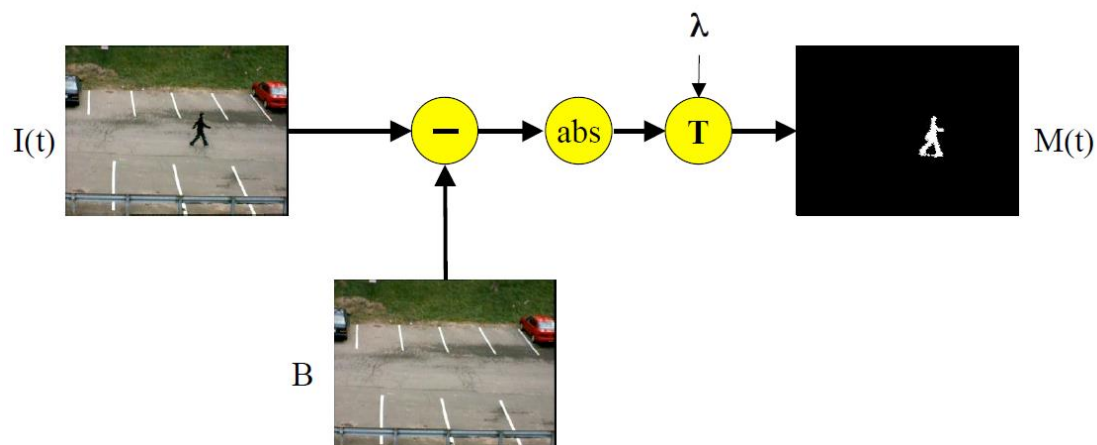


*Figure 16: YCbCr color model*

Using this color model is good for skin detection because we can select a range of skin color, and allow Y from 0 to 1. However, we will compare the result between HSV and YCbCr later.

### 3.1.3 Background Subtraction

Background subtraction is an old but useful method to extract object from an image. The basic idea is set up a background image at the beginning, the object can be extracted by using subtraction between the current image and background image. If the result bigger than the threshold, that pixel will turn to white color. [16]



*Figure 17: Background Subtraction*

However, it is very sensitive to the shadow, which means it will consider the shadow of hands as the moving object. Therefore, it is not suitable for our project.

### 3.1.4 Skin Color Filter

Based on the description of human skin extraction, we design using color model to extract the skin. Therefore, we can compare the result by using different color models for finding better option.



*Figure 18: Testing color model*

We can conclude the detection result is that HSV color model cannot include fingernail as part of hand. It is more sensitive to white and black, which means it has great ability to ignore the shadow, even dark skin and reflect light on the fingernail.

The YCbCr color model may include some part of shadow as hand, but it can include the dark skin and fingernail very well. However, the fingernail is very important, which can help us to stable the position of the fingertips.

Hence, we selected YCbCr as our color model to filter the skin color.

### 3.2 Noise Reduction

There are two basic functions to reduce noise, which are Erosion and Dilation.

#### 3.2.1 Erosion and Dilation

In order to do these two operations, it should build an element first. The element is determining the smoothing degree of object. For easy to understand the algorithm, we use a 3x3 square to perform those operations.

Erosion is very useful to reduce the noise, extract center point and separate the continuous pixels. We denote Erosion as  $A \ominus B$ . [17]



*Figure 19: Erosion*

Dilation is the reverse of Erosion, which can connect pixels and expand the pixel. We denote Dilation as  $A \oplus B$ . [17]

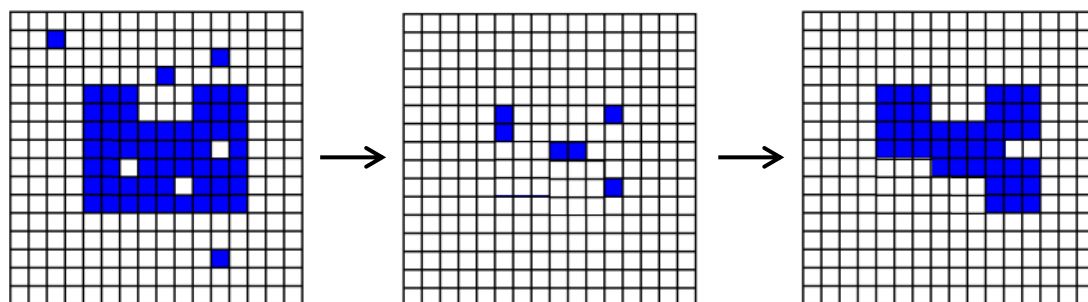


*Figure 20: Dilation*

#### 3.2.2 Opening and Closing

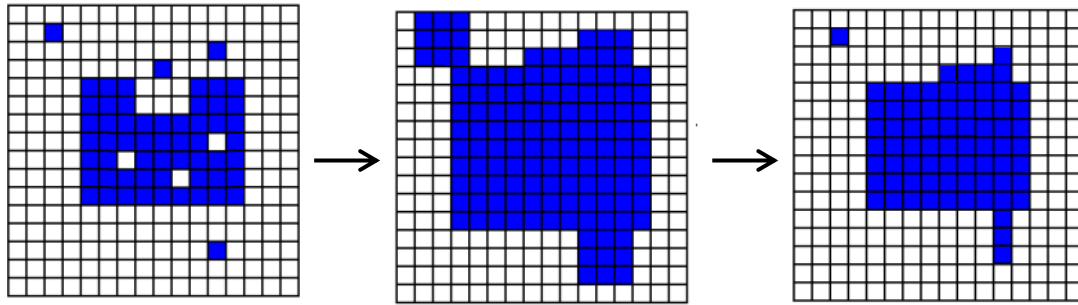
Opening and Closing are the combination of Erosion and Dilation.

Opening is executed Dilation after executing Erosion, which is useful to remove outside noises. We denote Erosion as  $A \circ B = (A \ominus B) \oplus B$ . [17]



*Figure 21: Opening*

Closing is executing Erosion after executing Dilation, which can fill the hold inside the object. We denote Erosion as  $A \circ B = (A \oplus B) \ominus B$ . [17]



*Figure 22: Closing*

In our system, the Closing operation is not important since the noises inside the shape can ignore easily. On the other side, to connect noises outside the shape is not good for detection. Therefore, we will use Opening operation in this project.

### 3.2.3 Reduce Noise

The color filter method can change the skin color into white, otherwise change to black. Therefore, some pixels are satisfying the range of skin color in the background. We can use opening and closing to group white pixels and estimate small piece of white area.

In our program, we will do the Opening operation to remove the noise in the image in the beginning. After removing the noise, we will select the biggest continue white area as hand area. In order to smooth the hand white area, the element of Opening operation is a circle.

### 3.3 Extract Features

After extracting white area, removed noise and performed smoothing. We can extract the finger features base on the grayscale image. There are some extraction's methods are provided in the internet. It is important that how making fingertip stable and increase accuracy in this phase.

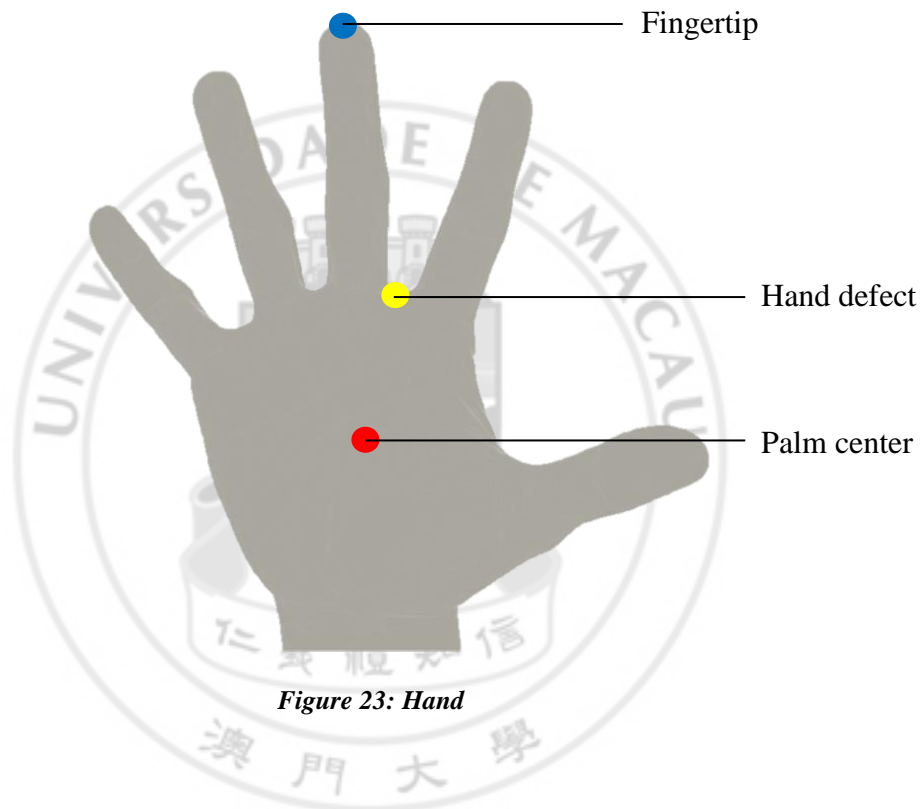
There are some features that can be detected though the hand shape. Such palm center, finger, palm defect and palm size.



### 3.3.1 Hand Features

Hand contains some features to detect the position of fingertips. We can use those features to find out the position of fingertips. For example:

1. The position of fingertip has the longest distance from palm center
2. The fingertip is nearly a sector.
3. The finger is thin and between defects normally.
4. Palm is nearly a circle.

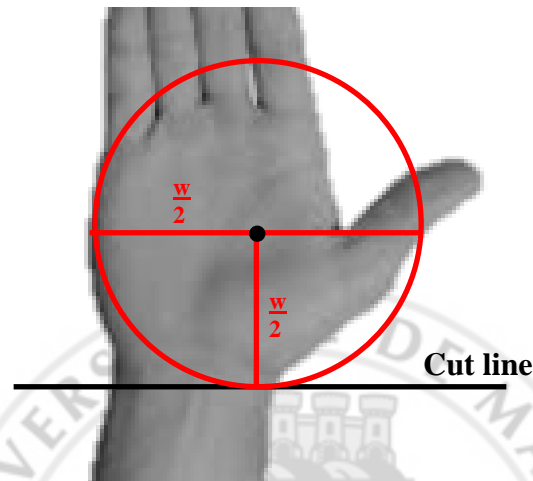


*Figure 23: Hand*

### 3.4 Cut Hand

After filtering the skin color, we found that the arm is detected at the same time, which affect the fingertip extraction method, and also increase the processing time in following steps. Therefore, we will cut the hand from the mask image. [17]

The basic idea is to find out the maximum width of palm as diameter to draw a circle. The bottom pixel of the circle is the cut line for cutting hand.



*Figure 24: Cut hand*

After cutting the hand from grayscale image, the center point of palm much more stable and it is good for hand feature extraction.

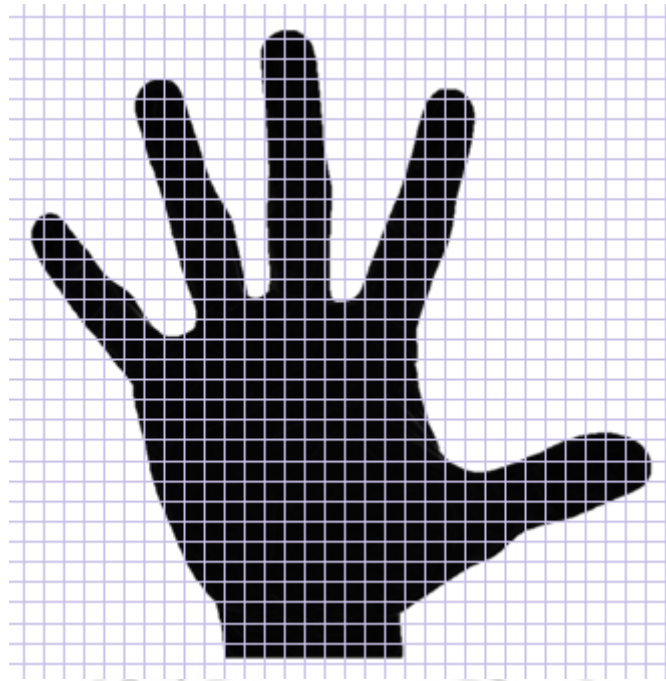
#### 3.4.1 Palm Center Extraction

The palm center is the main feature that can help us to find out the position of fingertips. The easiest way is to continue doing Erosion operation until one pixel left, thus the palm center is found. However, this method is using most of running time to process the image.

Therefore, we can find the gravimetric point from the grayscale image. This method is much faster and simpler. The equation we used is following:

$$\begin{cases} x_{center} = \frac{\sum x}{\text{number of point}}, \text{if pixel}(x, y) \text{ is white} \\ y_{center} = \frac{\sum y}{\text{number of point}}, \text{if pixel}(x, y) \text{ is white} \end{cases}$$

However, this method introduces many addition operations since it scans the whole image once, which is still not fast enough. Therefore, in order to speed up the processing time, we skip some pixels to get the similar result.

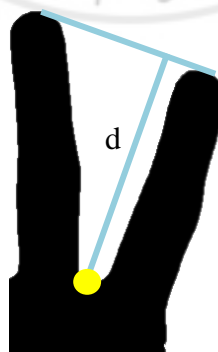


*Figure 25: Palm center*

Basically, the number of skipping pixels can speed up the processing time but reduce the accuracy of real gravimetric point. In order to balance the speed and accuracy, we have tried different setting for the skipping size. Finally, the accuracy does not drop too much when setting the size as 10.

### 3.4.2 Hand Defects

Hand defect is one of the main features for fingertip detection. The method is found out the maximum distance between two hull points. The point with maximum distance may be the defect of hand.



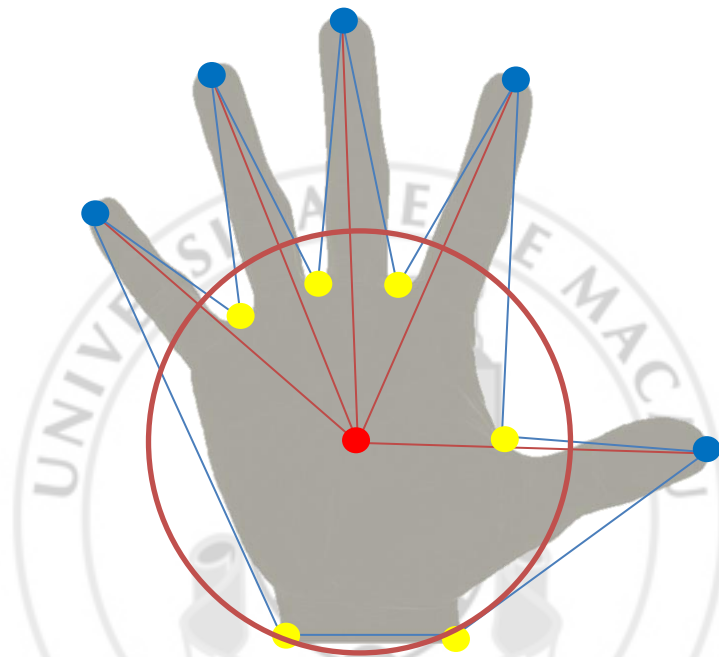
*Figure 26: Hand defect*

However, the distance of defect should larger than the minimum requirement. Otherwise, it will produce may defects even their distance is very small.

### 3.4.3 Fingertip Extraction

In fact, we had tried many different ways to find the fingertip. We summarize some features to identify the position of fingertips.

1. The finger always between palm defects and the angle is smaller than 45 degrees.
2. The fingertip always has the longest distance from palm center.
3. There are some points between fingertips along the point of the shape.



*Figure 27: Fingertip extraction*

Based on the previous methods, we found out the shape of the hand, the palm center point and hand defect points. In addition, we add defect points on the cut line to let the fingertip detection works.

The extraction criteria are the following:

1. Find every point from the shape between two defects, which point has a maximum distance from the center.
2. Check the points from the array, which within a range of angle by defects.
3. The distance between fingertip and the center should larger than the range of the circle.
4. Not allow fingertip to appear nearby the cut line and border of the window.

### 3.5 Optimize Fingertips Position

The mainly reason of optimization is that we want to stable the position of fingertips in order to increase the user experience.

The fingertips position has a shaking problem since the image is changing every time by the camera. We can set a threshold to overcome the shaking problem. However, to stable the position of fingertips and keep the sensitive don't drop too much are hard to balance.

#### 3.5.1 Anti-shaking Algorithm

The shaking problem is caused since the image captured from the camera is different, even it is the same object and absolutely static. It is because the camera translates the color from real object into digital signals. Therefore, the edge of the object is not stable in every image, which make the position of fingertips in not stable.

We implement the anti-shaking method for some features, such as: hand defect, palm center and cut line. The method for anti-shaking is simple but useful, which is:

$$points = \begin{cases} new\ points, & if\ distance(new, old) > threshold \\ old\ points, & if\ distance(new, old) < threshold \end{cases}$$

For example, the new position of palm center will compare the distance old point of palm center. If the distance not changes too much, use the old point. This method can apply to other features, which can make the detection more stable.

However, the most difficult part is the hand defect anti-shaking because the defect points may not on the shape. Therefore, we introduce more code to solve this problem, which search the distance between contour point and defect point.

### 3.5.2 Smooth Mouse Moving Algorithm

One important function of our project is to control the mouse moving activity through hand gesture. At first, the pointer is twinkling in the screen, which is caused by the cursor is updated every frame.

In order to smooth the mouse twinkling situation, we decided to update the position every time rather than every frame. We set a thread to update the position of the cursor, which thread looking for a target every time.

$$\text{Move point: } M_{n+1} = M_n + \frac{M_n + T_{n-1}}{\lambda}, \text{ where } \lambda \text{ is the speed to target } T$$

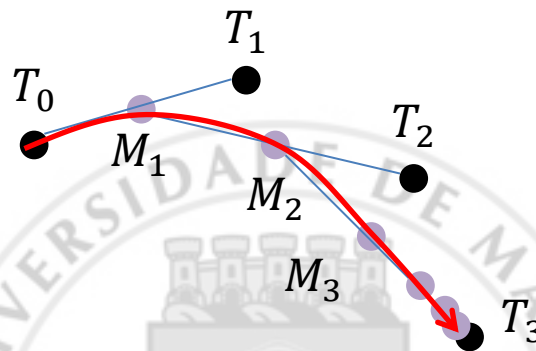


Figure 28: Mouse smoothing ( $\lambda=2$ )

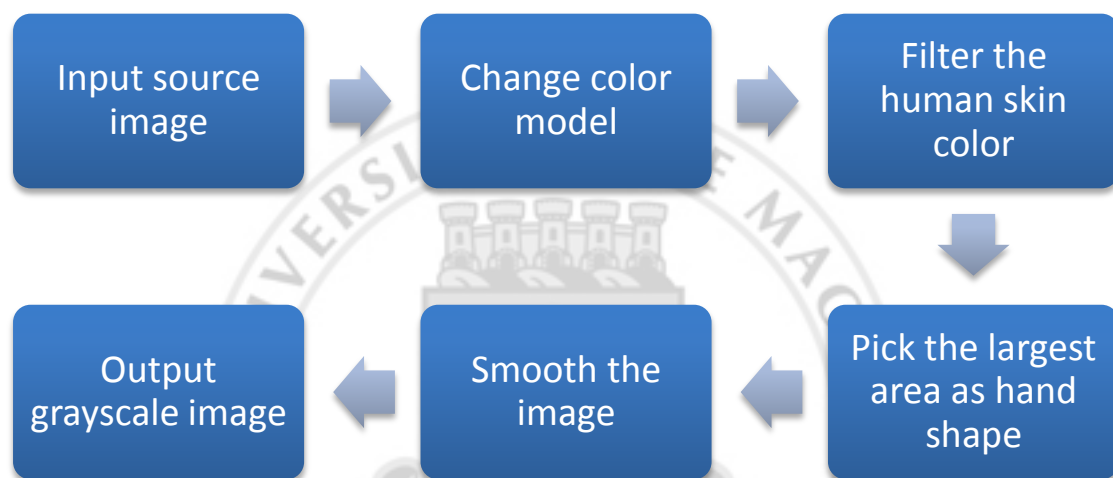
The cursor position is updated based on the above formula. The program can guess the middle point between current position and target even the target is changing.

## CHAPTER 4. IMPLEMENTATION

In our program, we are using OpenCV [18] to process the image. It is very useful tool which provide many image processing functions.

### 4.1 Color Filter and Hand Shape Extraction

Color filter means change the pixels to white color, otherwise change to black. The image with only white and black color, we call it grayscale image.



The first function is turning the image color from RGB to YCbCr color model. The second function is turn the range of color into white, otherwise turn into black.

```

cvCvtColor(src, image, CV_BGR2YCrCb); //change to YCbCr
cvInRangeS(image, cvScalar(y0, cb0, cr0), cvScalar(y1, cb1, cr1), grayscale);
// grayscale image
  
```

After the color filter, it will do the Opening operation in order to reduce the noise. It is easy to do because OpenCV provided the basic Erosion and Dilation operations.

```

cvDilate(grayscale, grayscale, element, 5); //do 5 times
cvErode(grayscale, grayscale, element, 3); //do 3 times
  
```

As you see, the fourth variable means doing that operation number of times. After these operations, the finger can become more and more thin and smooth. The element can be set as 3x3 circle by using:

```

IplConvKernel *element =
cvCreateStructuringElementEx(3, 3, 1, 1, CV_SHAPE_ELLIPSE, NULL);
  
```

## 4.2 Hand Features Extraction

### 4.2.1 Hand shape

We can find out the shape of hand from the grayscale image by the following function which also provided by OpenCV library:

```
cvFindContours(grayScale, storage, &contours, sizeof(CvContour),
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE); //get contours from the image
```



Using this function can find out all borders of the white area, and save into a sequence of point. The variable `CV_RETR_EXTERNAL` let the function select the outer contour only. Therefore, we don't need to consider the pixels inside the hand shape.

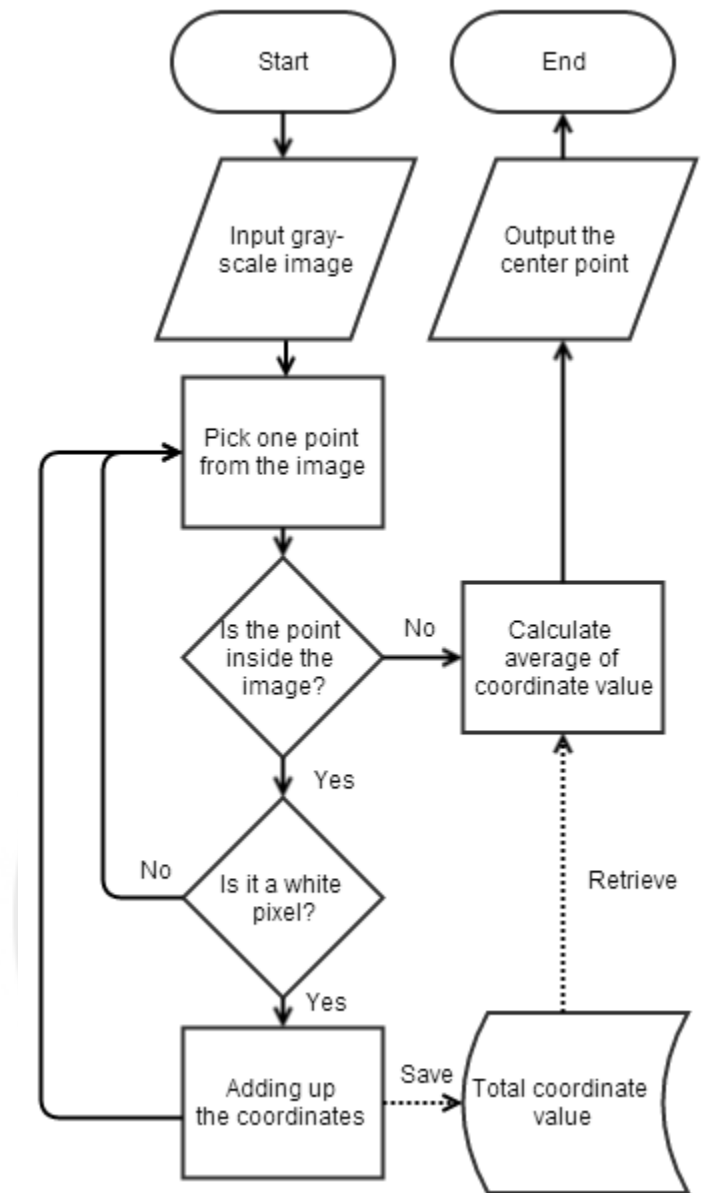
### 4.2.2 Extract Palm Center

The palm center has the similar characteristic to the gravimetric point of the image. The method we are designed to do is using the following equation.

$$\begin{cases} x_{center} = \frac{\sum x}{\text{number of point}}, \text{if pixel}(x, y) \text{ is white} \\ y_{center} = \frac{\sum y}{\text{number of point}}, \text{if pixel}(x, y) \text{ is white} \end{cases}$$

In our program, it is skipping every 10 pixels in the image, which can increase our processing about 0.002 seconds. The flowchart for the palm center extraction is the following:





**Figure 29: Extract palm center flowchart**

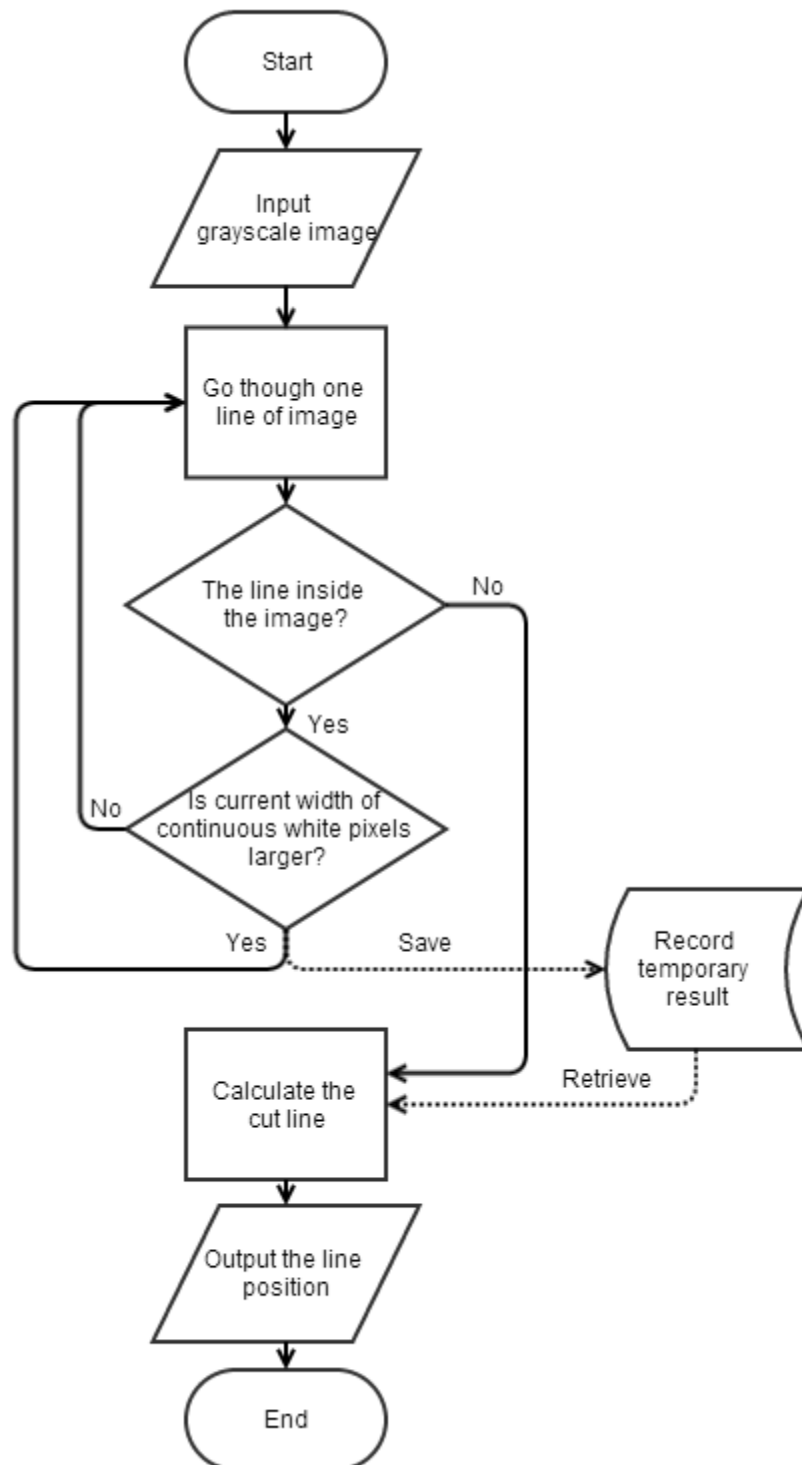
The source code for this method is:

```

for(int h = 0; h < bi_dist->height; h += 10) //height
    for(int w = 0; w < bi_dist->width; w += 10) //width
        //add the pixel when the image color is white
        if(255 == data[step * h + w]) {
            x += w;
            y += h;
            s++; //count the number of pixel
        }
//calculate the center point
if (s > 0) {
    x = x/s;
    y = y/s;
}
  
```

### 4.2.3 Cut hand

The detection of cut hand algorithm is easy to understand. First select the maximum continue width from the grayscale image, set the width as the diameter of circle. The lowest point of the circle will be the cut line of the hand.



*Figure 30: Cut hand flowchart*

```

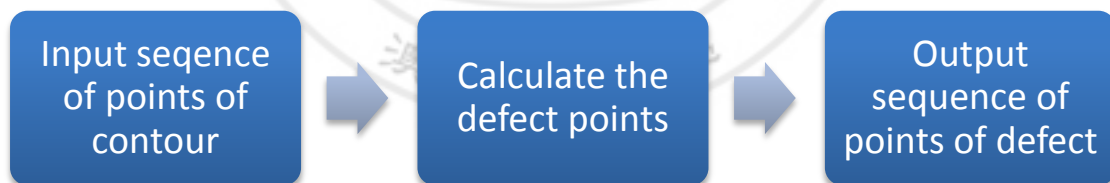
//get maximum width
bool flagFound = false;
for(int y = 0; y < img->height; y+=10){//increase speed (+10)
    for(int x = 0; x < img->width; x+=10){//increase speed (+10)
        int B = ((uchar*)(img->imageData + y*img->widthStep))[x*img->nChannels + 0];
        if (flagFound && B == 0) { //end point
            flagFound = false;
            end = x;
            if (count > maxCount){ //new record
                maxCount = count;
                line = y;
                maxStart = start;
                maxEnd = end;
            }
            count = 0;
        } else if (!flagFound && B == 255){ //start point
            start = x;
            flagFound = true;
            count++;
        }
        if (flagFound && B == 255){ //in middle
            count++;
        }
    }
}
//calculate cut line
midWidth = (maxStart+maxEnd) / 2;
cutLineTmp = line + ((maxEnd - maxStart) / 2);

```

#### 4.2.4 Hand defects

We can extract the hand defect by using OpenCV function directly:

```
cvConvexityDefects(contours, hull, defect_storage); //find defect points
```



The above function returns a list of points which is the defect of hand, and the distance of defect.

After extracting the defect points, we can filter some defect points according to the distance is deep or shallow.

```

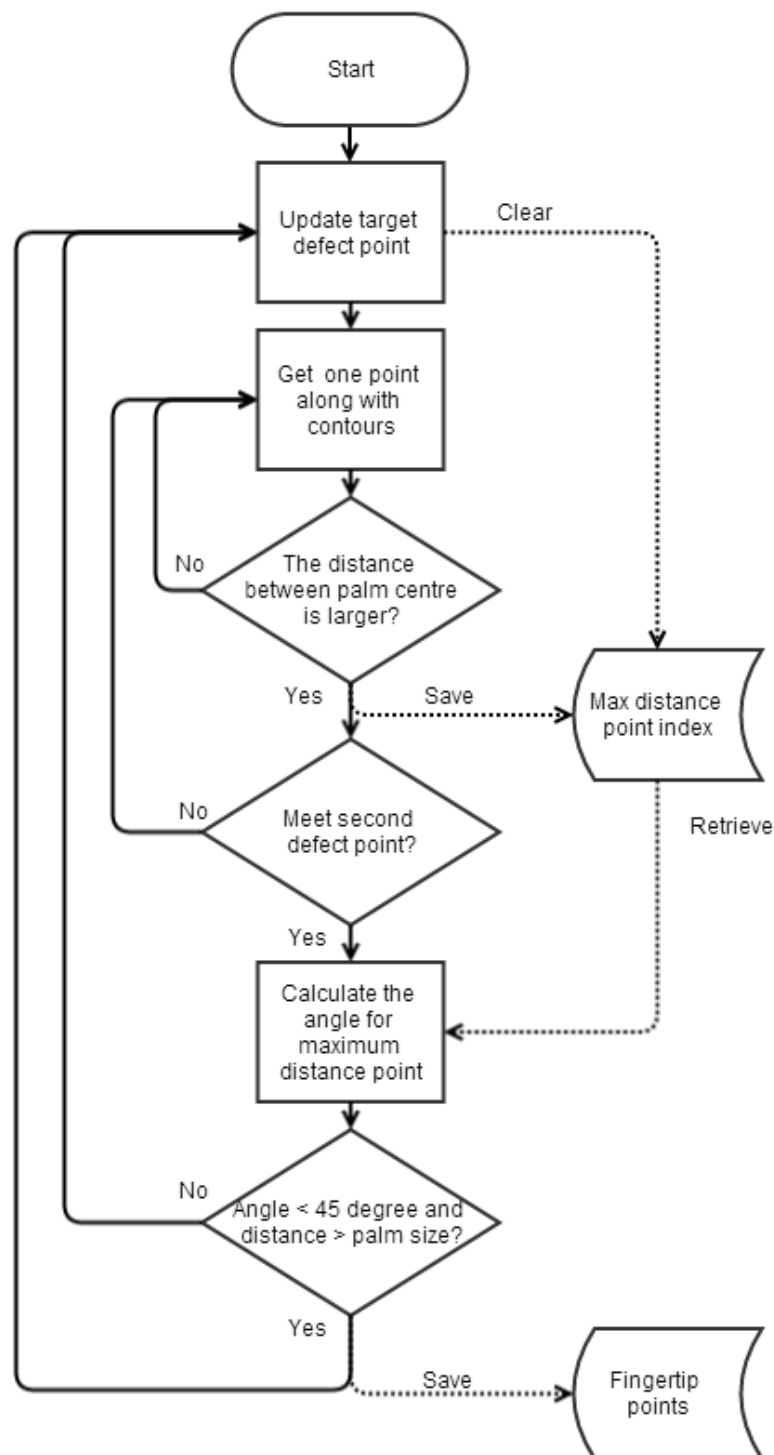
//save defect into palmArray iff distance > x
for(int i = 0; i < defect->total; i++) {
    CvConvexityDefect* d = (CvConvexityDefect*)cvGetSeqElem(defect, i);
    if(d->depth > 20) { //if bigger the number
        palmArray[count].x = d->depth_point->x;
        palmArray[count].y = d->depth_point->y;
        count++;
    }
}

```

### 4.2.5 Fingertips

The fingertip extraction is the mainly component of gesture analyzing. Let me reference the features in the Figure 27: Fingertip extraction in Chapter 3.

We are starting from the defect point of hand, and go through the contour point until back to the first defect point. During this process, we can calculate the position of fingertips with the features we had found before.



*Figure 31: Fingertip extraction flowchart*

In the above flowchart, the fingertip features will be extracted only when it's satisfying the requirements which are the length to the palm center and the angle between defects. The detail source code for the fingertip extraction is provided here:

```
//find the first palm(yellow) point
if (palm->total > 0){
    pyellow = (CvPoint*)cvGetSeqElem(palm,0);
    //j is the first palm(yellow) index
    for (j=0; j < contours->total-1; j++){
        pgreen = (CvPoint*)cvGetSeqElem(contours,j);
        if (dis(*pgreen, *pyellow) <= 6){
            break;
        }
    }
}

//find fingertip, save it into array
for (i = 1; i < palm->total+1; i++){//begin with start palm(yellow) point, end
with start point
    pyellowOld = pyellow;
    pyellow = (CvPoint*)cvGetSeqElem(palm, i);//get next yellow point
    dmax = 0;
    index = -1;

    //loop contours
    for (; j++){
        if (j >= contours->total) j = 0; //back to start point
        pgreen = (CvPoint*)cvGetSeqElem(contours,j);
        if (dis(*pgreen, *pyellow) <= palmDis){//meet next yellow point
            //take point by index, add to array
            if (index != -1) {
                point = (CvPoint*)cvGetSeqElem(contours,index);
                if (point->x < 630 && point->y < 470 ) { //remove border point
                    vector1.x = pyellowOld->x - point->x;
                    vector1.y = pyellowOld->y - point->y;
                    vector2.x = pyellow->x - point->x;
                    vector2.y = pyellow->y - point->y;

                    //cos angle
                    dotproduct = (vector1.x*vector2.x) + (vector1.y*vector2.y);
                    length1 = sqrtf((vector1.x*vector1.x)+(vector1.y*vector1.y));
                    length2 = sqrtf((vector2.x*vector2.x)+(vector2.y*vector2.y));
                    angle = dotproduct/(length1*length2);
                    if (angle > 0.7) { //angle less than 45
                        cvSeqPush(fingerseq, point);
                        fingertipOld[n-1][i] = *point;
                    }
                }
            }
            break;//change next yellow point as target
        } else {
            //calculate distance
            d = dis(*pgreen, center);
            if (d > dmax && dis(*pgreen, *pyellow) > d/3){//update dmax if bigger
                dmax = d;
                index = j;
            }
        }
    }
}
}
```

### 4.3 Optimization

I had implemented two optimization operations in this project, such as Anti-shaking and Smooth mouse moving.

### 4.3.1 Anti-shaking Algorithm

We implement the anti-shaking method for some features, such as: hand defect, palm center and cut line. The method for anti-shaking is simple but useful, which is:

$$points = \begin{cases} \text{new points, if } distance(\text{new}, \text{old}) > \text{threshold} \\ \text{old points, if } distance(\text{new}, \text{old}) < \text{threshold} \end{cases}$$

We add some variables to record the old position of the feature points. The position of the point will be updated by compared the distance between the new point and old point.

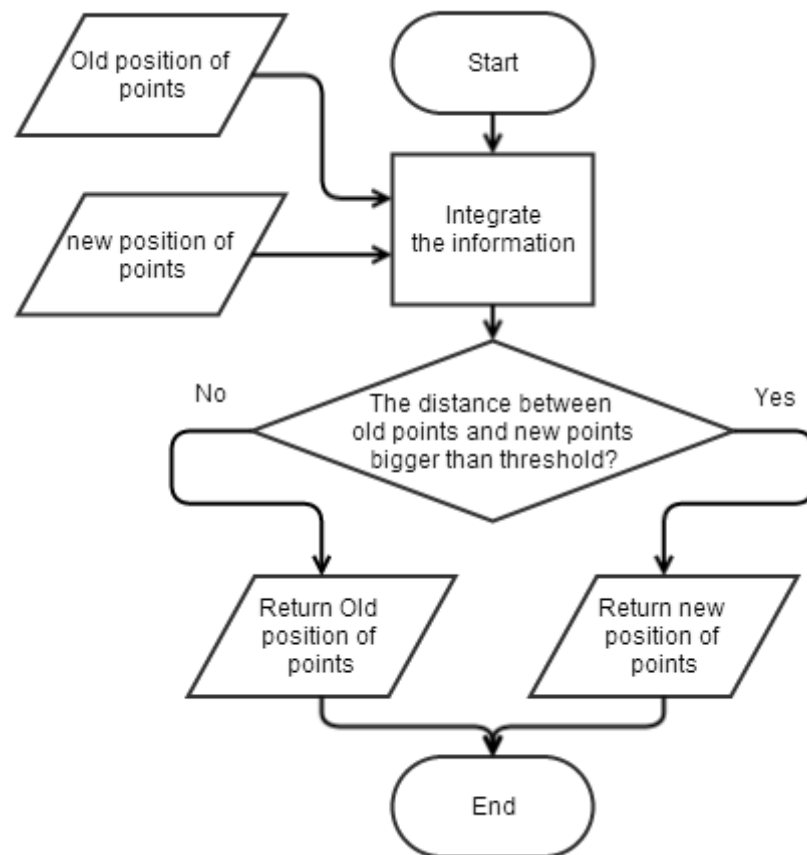


Figure 32: Anti-shaking flowchart

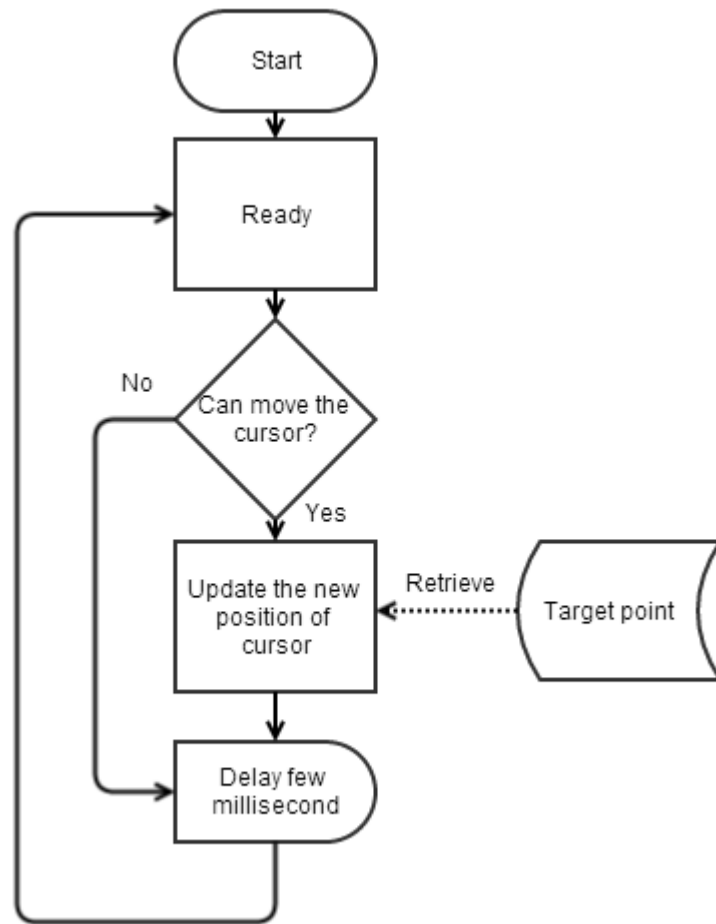
### 4.3.2 Smooth Mouse Moving Algorithm

This optimization is processed in a thread, so that the processing speed is not related to the image processing time any more. In order to slow down the processing time the thread, we added sleep operation to avoid using too much computational power.

Basically, the cursor update equation can simplify into this format:

$$CurrentPos = CurrentPos + \frac{moving\ direction}{moving\ speed}$$

The position of the cursor will move closer and closer to the target position which is updated by the pointing gesture which processed by Anika.



**Figure 33: Smoothing mouse moving**

```

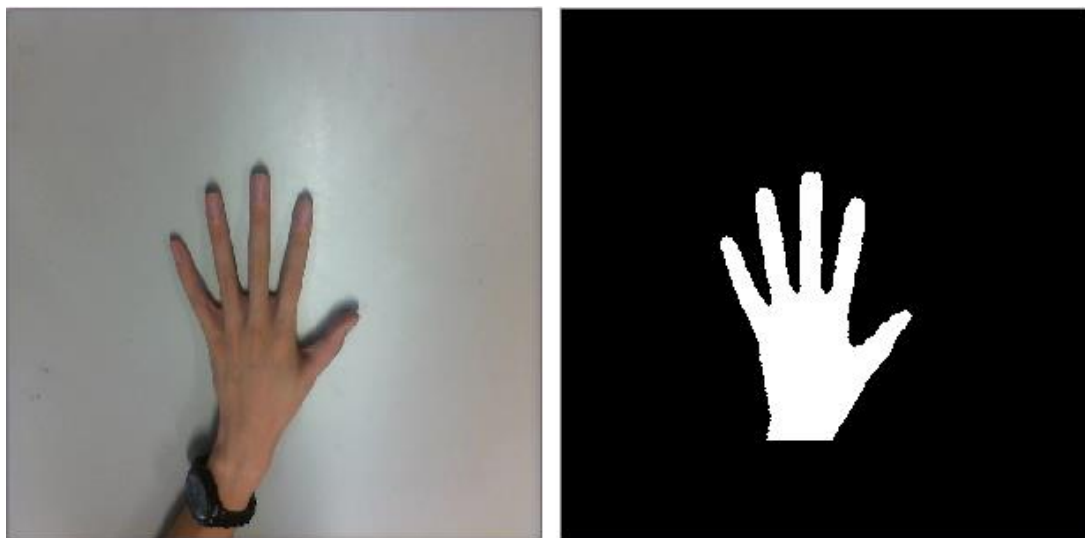
//optimize mouse smooth
DWORD WINAPI setNewCursorPos(LPVOID lpParamter){
    int smoothTime = 5; //one while of main program move how many time
    int smoothSpeed = smoothTime * 4; //the speed moving to the target
    while(1) { //forever loop in thread
        if (smoothCanMove){
            xCursor = xCursor + ((smoothPoint.x - moveAreaLeft) *
window.right / moveAreaWidth - xCursor) / smoothSpeed;
            yCursor = yCursor + ((smoothPoint.y - moveAreaUp) *
window.bottom / moveAreaHeight - yCursor) / smoothSpeed;
            SetCursorPos(xCursor, yCursor);
        }
        Sleep(whileTime / smoothTime);
    }
    return 0;
}
  
```

## 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 34: 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 35: 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.







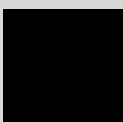





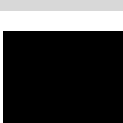

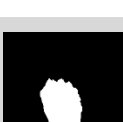

*Figure 36: Complex background*

### 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 reference the gestures table from my partner for this system:

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







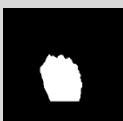


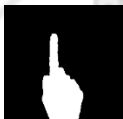




8			Left/Right	Previous tab or next tab
9			Left/Right	Previous page or next page
10			Up/Down	Open or Close the On-Screen keyboard
11			None	Open a new tab
12			Left/Right	Zoom in or out
13			None	Video play or stop
14			Left/Right	Increase or decrease the volume

Table 1: Gesture table

### 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 2: 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.

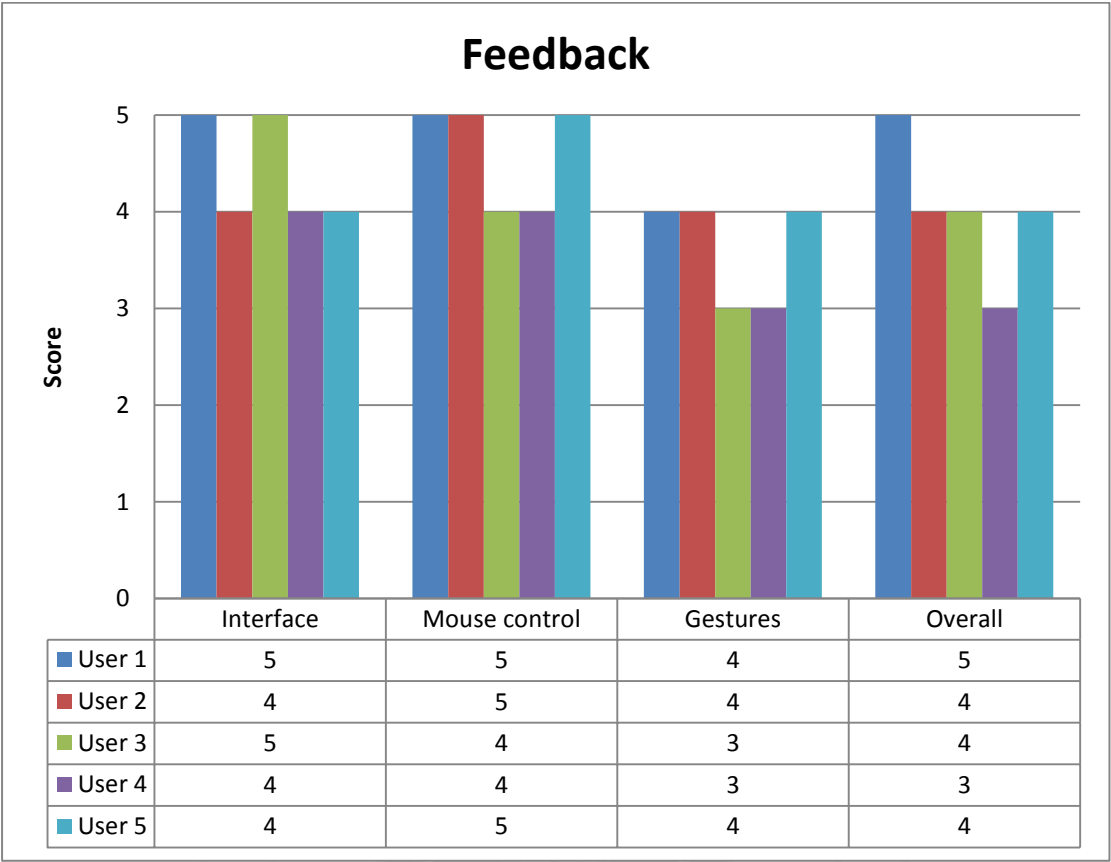


Table 3: Feedback

## 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 4: 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] A. Ford and A. Roberts, “Colour space conversions,” *Westminster University, London*, vol. 1998, pp. 1–31, 1998.
- [14] “HSV Color Model .” [Online]. Available:

[http://en.wikipedia.org/wiki/HSL\\_and\\_HSV](http://en.wikipedia.org/wiki/HSL_and_HSV).

[15] “YCbCr Description.” [Online]. Available: <http://en.wikipedia.org/wiki/YCbCr>.

[16] D. Ramanan, “CS 117 Project in Computer Vision, Lecture: Background subtraction,” University of California, 2013, p. 15.

[17] P. Premaratne and Q. Nguyen, “Consumer electronics control system based on hand gesture moment invariants,” *Computer Vision, IET*, vol. 1, no. 1, pp. 35–41, 2007.

[18] “OpenCV Introduction.” [Online]. Available: <http://docs.opencv.org/modules/core/doc/intro.html>.

