<table>
<thead>
<tr>
<th>Title</th>
<th>Real time hand detection and gesture recognition system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Jin, Man Mau (金文茂)</td>
</tr>
<tr>
<td>Citation</td>
<td>Jin, M. M. (2011). Real time hand detection and gesture recognition system (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2011</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2031/6418">http://hdl.handle.net/2031/6418</a></td>
</tr>
<tr>
<td>Rights</td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
Project Code: 10CS086

Project Title:
Real Time Hand detection and gesture recognition system

(Volume 1 of 1)

Student Name : Jin Man Mau
Student No. : 
Programme Code : BSCCS

Supervisor : Dr Raymond Wong HS
1st Reader : Lee, Ka Chun Kenneth
2nd Reader : Kwong, TakWuSam
Date : 3-4-2011
Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Real Time Hand detection and gesture recognition system

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: Jin Man Mau  Signature: ____________

Student ID      : ___________________  Date   : 3rd, April, 2011
Acknowledgments

I would like to acknowledge and extend my heartfelt gratitude to my project supervisor Dr Raymond Wong for the long support and guidance during the project development. His comments and encouragement have been illuminated me in solving many difficulties during the project, thank you.

I must also express my deepest gratitude to all those who helps, tolerates or cares me to the completion of the project.
Abstract

Detection and recognition technique are very important in computer vision architectures. So far, it has much different object detection application on this area. These objects include face, fingerprint, human body, gesture and so on. The technique use in face detection are achieving excellent performance in real time, but those technique may not suitable for other object especially hand detection and gesture recognition. Existing hand detection approaches based on the Viola-Jones methods will be influenced by the background noise of training images and the rotation of the images. As hands are non-rigid objects, the training images always contain many other objects that decrease the performance of the training result. It then gives rise to the need of an improvement the detection method for increasing accuracy. Finger tip detection also contains lots of noise during the detection such as the shadow of the hand, the background object. It not only needs to determine the finger tip point correctly, but also need to consider how to eliminate the unnecessary point in the hand.

To cope with existing needs, this study looked into both feature and learning-based technique so as to for solution of the gesture detection. The features such as skin detector and the Viola-Jones method, and a learning based algorithm, Adaptive Boosting(AdaBoost), were used for the hand detection. And also we will find out optimum method how to recognize the finger dip accurately.

This project development will focus on the relationship between different factors in hand detection and gesture recognition. The different factors include the number of the training image set, stager number, and error rate and hit rate. To investigate those factor for perform better detection. On the finger dip detection side, we will focus on how to improve or replace the existing method or algorithm to detect the finger tip accurately.
# Table of Contents

STUDENT NAME ........................................................................................................... 0  
STUDENT FINAL YEAR PROJECT DECLARATION .......................................................... 1  
ACKNOWLEDGMENTS .................................................................................................... 2  
ABSTRACT ........................................................................................................................ 3  
TABLE OF CONTENTS .................................................................................................... 4  
1. INTRODUCTION ............................................................................................................ 7  
  1.1 PROJECT BACKGROUND ..................................................................................... 7  
  1.1.1 OVERVIEW OF HAND DETECTION IN COMPUTER VISION ......................... 8  
  1.1.2 OTHER TRENDS IN GESTURE RECOGNITION .............................................. 8  
  1.2 LOOPHOLES AND DIFFICULTIES ...................................................................... 9  
  1.2.1 INADEQUATE STRUCTURAL INFORMATION IN GESTURE RECOGNITION 9  
  1.2.2 THE INFLUENCE OF THE BACKGROUND NOISE ....................................... 9  
  1.2.3 THE SURPLUS POINT DURING FINGER TIP DETECTION ....................... 9  
  1.3 AIM ......................................................................................................................... 10  
  1.4 OVERVIEW ............................................................................................................ 10  
2. LITERATURE REVIEWS .............................................................................................. 11  
  2.1 FEATURE BASED HAND DETECTION .................................................................... 11  
  2.2 IMAGE SEGMENTATION ....................................................................................... 13  
  2.3 LEARNING BASED GESTURE RECOGNITION .................................................... 14  
  2.4 BACKGROUND SUBTRACTION FOR IDENTIFY HAND OBJECT ..................... 14  
  2.5 EXTRACTING THE FINGER TIP POINT BY DOT PRODUCT .............................. 15  
  2.6 INTEGRATION OF DIFFERENT ANALYSIS WORK ........................................... 15  
3. PROJECT DESIGN ...................................................................................................... 16  
  3.1 GESTURE RECOGNITION ...................................................................................... 16  
  3.1.1 SKIN CLASSIFIER ............................................................................................. 16  
  3.1.2 CONTOUR FINDING ......................................................................................... 16  
  3.1.3 HAAR-LIKE CLASSIFIER CASCADES ............................................................... 16  
  3.2 FINGER TIP DETECTION ....................................................................................... 18  
  3.2.1 SKIN CLASSIFIER ............................................................................................. 18  
  3.2.2 COMPUTER THE CONVEXITY POINT OF THE CONTOUR ....................... 18  
  3.2.3 CENTRALIZATION THE CONVEXITY POINT .............................................. 18  
  3.2.4 FINGER TIP CLUSTERING METHOD ............................................................... 18  
  3.2.5 PROXIMITY MEASURE OF CLUSTER ............................................................. 18  
  THE PREVIOUS ONE CLUSTERING METHOD WOULD MISS THE FINGER TIP POINTS AND IT IS NECESSARY TO USING PROXIMITY MEASURE TO CALCULATE THE ERROR. THE METHOD IS CALLED SUM OF SQUARE ERROR (SSE). .................. 18  
  3.3 OVERVIEW ............................................................................................................. 19  
4. METHODOLOGY .......................................................................................................... 21  
  4.1 MACHINE LEARNING ............................................................................................. 21
4.2 SKIN COLOR DETECTION ................................................................ 31
4.3 GAUSSIAN SMOOTH ................................................................ 32
4.4 CALCULATING THE DOT PRODUCT .............................................. 33
4.5 CENTRALIZATION POINT .............................................................. 34
4.6 FINGER TIP CLUSTERING METHOD .............................................. 35
4.7 PROXIMITY MEASURE OF CLUSTERS ........................................... 42

5. IMPLEMENTATION ........................................................................ 45
5.1 PREPARATION OF THE CASCADE CLASSIFIER .................................. 45
5.1.2.1 METHODS OF SEPARATING BACKGROUND .................................. 47
5.1.2.2 TRAINING IMAGE WITH SOLID BACKGROUND .................................. 47
5.1.2.3 TRAINING IMAGE WITH RANDOM BACKGROUND ............................. 49
5.1.3.1 PREPARING THE DESCRIPTION FILE ........................................ 50
5.1.3.1.1 DESCRIPTION FILE FOR POSITIVE IMAGE ............................... 51
5.1.3.1.2 DESCRIPTION FILE FOR NEGATIVE IMAGE ................................ 52
5.1.3.2 CREATING THE TRAINING SAMPLE .......................................... 52
5.2 SKIN COLOR DETECTION .............................................................. 58
5.3 CONTOUR FINDING ...................................................................... 60
5.4 INTEGRATION OF DIFFERENT ANALYSIS METHODS ................................ 60
5.5 THE FINGER TIP DETECTION .......................................................... 64
5.5.1 SKIN CLASSIFIER ...................................................................... 64
5.5.2 CONTOUR FINDING AND REGION OF INTEREST IN FINGER TIP DETECTION 65
5.5.3 COMPUTER THE CONVEXITY POINT OF THE HAND ......................... 65
5.5.3.1 USING OPENCV INTERNAL FUNCTION ..................................... 65
5.5.3.2 ALGEBRAIC OPERATION .......................................................... 66
5.5.4 CENTRALIZATION IN HAND REGION ........................................... 67
5.5.5 CLUSTERING THE FINGER TIP POINT .......................................... 68
5.5.6 PROXIMITY MEASURE IN FINGER TIP DETECTION ......................... 69

6. RESULTS ...................................................................................... 71
6.1 GESTURE RECOGNITION .............................................................. 71
6.1.1 DESCRIPTION .......................................................................... 71
6.1.2 RESULT AND FINDINGS ............................................................ 73
6.1.3 DISCUSSION AND ANALYSIS .................................................... 75
6.1.3.1 THE OPTIMUM CLASSIFIER ................................................. 82
6.1.3.2 THE EVALUATION OF TRAINING CLASSIFIERS ......................... 85
6.1.4 IMPROVEMENT OF THE STRONG CLASSIFIER ............................. 87
6.1.4.1 RESULT ................................................................................. 87
6.1.4.2 DISCUSSION AND ANALYSIS .................................................. 88
6.1.5 LIMITATIONS ........................................................................... 89
6.1.6 RECOMMENDATIONS .............................................................. 89
6.2 FINGER TIP DETECTION .............................................................. 90
1. Introduction

In this chapter, an introduction for hand detection in computer vision will be discussed. Afterwards, this report explains the significance on hand detection, important of this in human computer interaction and the technology we have for hand detection nowadays.

Then, the problem was brought to light that gesture recognition is more challenge than other detection such as face detection. And gesture recognition and hand detection also encounter some difficulties through the process.

Therefore, in the next section, this project proposes a generic idea of the hand detection and gesture recognition in the computer vision field.

1.1 Project Background

Keyboard and mouse is the early input device for a computer, computers have role in processing information passively. In order to enable the computers to become more actively, computer vision was generated. Computer vision development has had the great advancements in recent year. The famous computer vision book called “Learning OpenCV” mention that “Computer vision is the Transformation of data from a still or video camera into either a decision or a new representation. All such transformations are done for achieving some particular goal.” (Bradski, 2008). One of the techniques in computer vision is hand detection. In recent years, object detection application have become more general and found their way into different electronic devices such as digital cameras, capable of detecting faces as well as facial expressions. Apart from the facial detection side, computer vision can use in the field of human-robot interaction such as finger tip detection and gesture recognition.
1.1.1 Overview of hand detection in computer vision

Hand detection is an area of active current research in computer vision. It is applies of the image understanding. It divides into two parts: object detection and object recognition. Object detection is to try to find the position of a certain object in a sequence of image; Object recognition is to try to recognize and distinguish from object. (Nyan Bo Bo,2007). For instance, object detection detects the human hand; object recognition finds what gesture it is.

The gesture detection can be enhanced human-computer interaction. The most challenging issue is how to communicate with a robot instinctively and directly. Using hand gesture is a natural way for interaction between people and the robots because it is similar to the humans communicate that mainly by vision and sound ( Ko-Chih Wang,2007). Another advantage is the user not only can communicate from a distance, but also no need any physical contact with the computer. Unlike audio commands, a visual system would be preferable than sound because sound would make cause a disturbance in noisy environments but visual is not. Thus, hand detection and hand gesture recognition could be essential to human-robot interaction.

1.1.2 Other trends in gesture recognition

Apart from the man-machine interface, a gesture recognition system could be used in any of the following areas:

- **3D animation:** Convert the hand movement rapidly and simply into 3D computer space for the purposes of computer animation.
- **Visualization:** Just as objects can be visually examined by rotating them with the hand, so it would be advantageous if virtual 3D objects (displayed on the computer screen) could be manipulated by rotating the hand in space (Taehee Lee, Tobias H’ollerer, 2007)
- **Computer games:** Using the hand to interact with computer games would be more natural for many applications.
- **Control of mechanical systems (such as robotics):** Using the hand to remotely control a manipulator.
1.2 Loopholes and Difficulties

Over the last 15 years, the problem of hand tracking has become an attractive area for research (Nyan Bo Bo, 2007). Hand detection and gesture recognition is more difficult and challenge than other object detection such as face detection. It needs to overcome the environment disturbing and structural difficulties issue.

1.2.1 Inadequate Structural information in gesture recognition

Compare with the face detection, hand tracking is more difficult. Since, the face contains more structural information such as eyes, mouth and so on. On the contrast, the structural information such as gaps between fingers that provided in hand is less than face (Nyan Bo Bo, 2007). Therefore, the gaps between fingers will become the background noise of the image, and we need to remove that to prevent disturbing during the recognitions process.

1.2.2 The influence of the background noise

In addition, as background noise of training images degrades the training speed and detection accuracy of gesture recognition. One of the possible solutions is collected more training images and reduce the background noise influence. And data collection approach suggests that capture the training images randomly in different light condition and various backgrounds. (Anton-Canalis and Sanchez-Nielsen, 2006) However, the negative training image is representing the several of background, so it may not capture the positive image on the different backgrounds. It will be discussed more lately.

1.2.3 The surplus point during finger tip detection

Ankit and Kumar Ashis (2009) suggest using the mathematic method to find out the finger tip location, but many unnecessary points need to be treated during the detection. About this issue, OpenCV library provider some function to extract the finger tip point roughly. Although this method also contains unnecessary points, the amount of the points is less than the previous methods. On the other hand, they also suggest using the background subtraction method to detection the hand. However, this method not
only can detect the hand object, but also the moving object. In addition, the shadow of the hand is the main issue to interrupt the detection process. Therefore, in this project, we would investigate other method such as skin detection instead of background subtraction.

### 1.3 Aim
The aim of this study is to study a learning-based gesture recognition and finger tip detection scheme with feature based methods, including Viola and Jones Detector, HSV color model for image segmentation, clustering method for separate the finger tip points and the Adaptive Boosting Learning Algorithm (AdaBoost) was used as learning based algorithm. And investigate the relationship between different factors in hand detection and gesture recognition.

### 1.4 Overview
In this project, the first part is the introduction of hand detection and gesture recognition. The second part covers related knowledge includes feature and learning base algorithm, the evaluation of detection system. The third part describes the overall system design of gesture recognition. The fourth part is indicated the methodology and it includes AdaBoost algorithms, Viola and Jones Detector, clustering and so on. The fifth part presents the implementation procedure. The performance of our detection and recognition system are shown in the sixth part. At last is the conclusion and future work.
2. Literature Reviews

In this chapter, the literature review on the propose solution would be further discussed. We would like to review different potential technique used in gesture recognition and finger tip detection or some improvement the performance of the detection. First, we will discuss feature-based gesture recognition and then review the machine learning techniques. Second, the finger tip detection technique would be discussed and also some improvement would bring up.

2.1 Feature based hand detection

2.1.1 Viola and Jones Detector

Viola and Jones have proposed famous face detector architecture in 2004. The Viola-Jones face detector achieve 90% face detection rate and less than 50 false detection in 18,901,947 sub-windows scanned. (Viola and Jones, 2004) Due to its high accuracy on face detection, this approach suitable to detect some other objects. Despite this success, it may not have same performance in gesture detection. Mathias K"olsch and Matthew Turk (2004) tried to use Viola and Jones Detector for in-plan rotations of hand appearances. The result is that only about 15 degree of rotations can be efficiently detected with the detector. The result is different from the method’s performance on face detection. The reason is that if we compare the training image between hand and face which shown in Figure 2.1.1a and Figure 2.1.1b, the hand training images contain much more background than face images. Because of the characteristic of face can bounded by a rectangle box perfectly and no any background noise exists in the box. If the training image set contain much more noise, it is difficult to maintain high performance during the detection. Therefore, separate the background and the hand is required to achieve more accurate gesture detection.
Figure 2.1.1a the hand training images which contain more background noise than the face training sample. (Ko-Chih Wang, 2007).

Figure 2.1.1b some of sample used by Viola and Jones methods (Viola and Jones, 2004)
2.1.2 Lowe’s SIFT Feature

The Scale Invariant Feature Transform (SIFT) feature is introduced by Lowe. It is “a point consists of a histogram representation of gradient orientation and magnitude information within a small image patch surrounding this point” (Lowe, 2004). SIFT can detect the object and it is not affected by rotation and scale feature. Also it can work in some variation of light, viewpoint and noise. Although SIFT matching algorithm can achieve a good performance to match an object in a cluttered image even the images’ scale and rotation is different, the objective of the most object recognition is not only to recognize the special objects, but also to able to recognize a category of the objects (Ko-Chih Wang, 2007). However, if the object is move faster, the tracking would be lost, so if the ability of tracking in strong classifier is good, it is no need to implement SIFT.

2.2 Image Segmentation

2.2.1 Using HSV Color Space

Dae Hyun Kim and Myoung-Jun Kim (2006) proposed using HSV color model (shown in Figure 2.2.1a) rather than using RGB color space to determine the color of human skin. Also, K. Sandeep and A.N. Rajagopalan (2003) made use of color histogram in HSV color space for detecting human faces in color images and its performance is found to be quite satisfactory. However, this HSV color algorithm could not work in certain case. For example, if there are colors in the image which is similar to the skin but are not skin pixels, it could cause two skin regions to be recognized. Potentially, the HSV color model can use to separate the background noise of the training images for the machine leaning. It will discuss more lately.

Figure 2.2.1a conical representation of HSV color model
2.3 Learning based gesture recognition

2.3.1 The Adaptive Boosting Learning Algorithm

The Adaptive Boosting (Adaboost) learning algorithms can integrate the information of a category of objects. It can combine the weak classifiers which cannot provide satisfactory result to become a strong classifier to get the better result. The Adaboost learning algorithm chooses the best weak classifier from a set of positive and negative images. After choosing the best weak classifier, the weights of the training images are adjusted by Adaboost algorithms. In this round the weights of classified training images are decreased and the unclassified images are increased. In the next round, the unclassified images will be more focused by the Adaboost and try to correctly classify the misclassified images. The whole procedures are finished until a predefined performance is satisfied. S. M. S. Islam, M. Bennamoun and R. Davies(2008) applied this algorithm to detect the ear and the result is fast and accurate. However, Ko-Chih Wang (2007) reported that the result of using AdaBoost with Viola Jones detector in hand detection is worse than face detection accuracy due to the structural problem of hand. And he proposed using AdaBoost with SIFT Feature is more accurately. Therefore, it is necessary to apply AdaBoost learning algorithm with suitable detector for increasing the detection accuracy.

2.4 Background subtraction for identify hand object

The purpose of using the background subtraction technique is to find out the moving hand object in Ankit and Kumar Ashis’s project. The idea of the background subtraction is capture the model of the background first and then compared against the current image. After comparison, the known background parts are subtracted away. In their project, they use the background subtraction to extract the moving hand object. Although it can be detected the hand moving, it contains several disadvantage. First, it not only can be detected the hand, it can detected any object which haven’t occur in the first captured screen. Second, if the brightness of the environment is change, it would be interference the detection, because the whole image is varied with the background, then the changed part would treat as moving
object. Third, shadow of the hand will influence the extraction process. If the
detection situation can produce shadow of the hand, the shadow would be treating
as the moving object that will affect the detection result. Because of these reason, so
we decide using the skin classifier instead of the background subtraction in the
finger tip detection process.

2.5 Extracting the finger tip point by dot product
Ankit and Kumar Ashis’s project purposed that using the dot product to calculate
the finger point. After background subtraction, they used contour to probable
hand regions. And then using the dot product to find out the threshold index
between the vectors that in the contour. Then can find out the corresponding
finger tip point. However, it would also find out the peaks and valleys in the hand
and other surplus points. It is necessary to make lots of effort to eliminate these
points. Thus, we can use the OpenCV internal function to find out all the
convexity points in the hand region instead of the using the dot product method.
Because of the points that found out by OpenCV internal function is less than
using the dot product one.

2.6 Integration of different analysis work
It is not enough if one detection technique is used alone because different kinds of
method can deal with different problem during recognition or detection. For example,
HSV model method can remove the background noise in training image that can
increase the accuracy during the Adaboost training. The integration of different
methods is necessary to improve the quality of gesture recognition performance and
finger tip detection.
3. **Project Design**

The real time gesture recognition and finger tip detection system's basic operation describes as in the following step (The flow chart of the gesture recognition and finger tip detection operation is shown in figure 3a, and figure 3b):

3.1 **Gesture recognition**

The gesture recognition is divided into three major parts: first, skin classifier for determine whether the object is hand or not. Second part is the contour finding which is for extracting out the hand region. The last one is using the Haar-like classifier cascades to identify the open palm gesture. All the steps are introduced briefly in the following.

3.1.1 **Skin classifier**

Before using the Haar cascade to detect the hand gesture, it is necessary to find out the human hand position. Using the HSV threshold to determine the hand region, and then focus on these regions for the gesture recognition.

3.1.2 **Contour Finding**

After extract the region that may be contain the object, we need to use the contour finding methods to find out the boundary of that area. If the area is too small, then it will be disregard during the hand detection. And using the region of interest method that proposed by OpenCV to focus these area for the recognition and detection.

3.1.3 **Haar-like Classifier Cascades**

For the gesture recognition, we will use the AdaBoost algorithm to training the strong classifier. The following is the step for the training the classifier cascades:

1. Training data preparation
2. Image segmentation
3. Creates sample
4. Training sample and use the strong cascade.
3.1.3.1  **Data preparation**

First, in the data preparation, it is necessary to prepare vast original images used for Adaboost training. A large amount of original image is to improve the detection accuracy. It separated into two types, the positive and the negative images. The positive image is the detecting object image, for example, the hand detection needs many hand pictures as the positive image. Another is the negative images which haven’t contained any object picture. In this project, I will focus on palm hand detection. We need to take positive images via a webcam and the background images from internet or a webcam.

3.1.3.2  **Image segmentation**

Second, in the image segmentation stage, it is important for reduce the background noise of the original image. Therefore, the pure background color is required during the image capture progress. Moreover, the lighting condition and background noise also will influence the quality of the training data such as a hand and the background color is not separated clearly due to the lighting condition. To tackle the problem, we use HSV model’s threshold value to separate the background noise in the training images.

3.1.3.3  **Creates sample**

Third, in create sample stage; we need to create a description file for those positive images. After generated the description file such as sampel.txt, we use this description file to create a file name sample.vec to store the sample file. This sample.vec files which stores the entire sample is for the palm gesture recognition. Therefore, we can able to produce each classifier for palm gesture.

3.1.3.4  **Training sample and use the strong cascade.**

Fourth, it is time to create the classifier by using the Harrtraining in OpenCV. When using the Haartraining command, we have to set up a lot of negative images, more background image, and the better result. After the process is finish, the Haartraining create an .xml file. This classifier is cascaded stronger classifier. At last, the stronger
classifier is used to detection the hand gesture; it is the major issue for gesture recognition.

3.2 Finger Tip Detection
The following section is described the finger tip detection process. First, we also using the skin classifier to extract the hand region. Second part is using the OpenCV internal function to find out all the convexity points within the hand region. Third part is using the some clustering methods to reduce the unnecessary points and enhance the accuracy of the detection.

3.2.1 Skin classifier
The technique is same as in gesture recognition that is mentioned before.

3.2.2 Computer the convexity point of the contour
There are two methods to find out the convexity points; first one is using OpenCV internal function, second one is using the algebraic operation that is to calculate the angle between the contour points for the detection. It would be used two different methods to find out the convexity point and see which calculation are the most suitable methods.

3.2.3 Centralization the convexity point
Centralization point is use to find the central point of a set point. The aim of this technique is use to reduce a set point to become one point. It can remove a large amount of noise or unnecessary point during the computer vision process.

3.2.4 Finger tip clustering method
During the finger tip detection, it is necessary that to find out the noise such as false positive points in the hand region. To solve with this problem, we would using the clustering method to separate the noise and the critical point in the hand region.

3.2.5 Proximity measure of cluster
The previous one clustering method would miss the finger tip points and it is necessary to using proximity measure to calculate the error. The method is called sum of square error (SSE).
3.3 Overview

We will compare the performance of gesture detection by different training image set and the training time. For instance, using a small training set: 1000 positive and 2000 negative images. The number of different training will be discussed later.

Moreover, the performance of the gesture recognition system not only just depends on the sample set, but also the stage number. It is an important factor in the training stage. In this project, to evaluate the recognition system is depend on the error rate and hit rate. Thus, we will find out the relationship between stage number, error rate and the hit rate.

Apart from investigate the machine learning, it also required to eliminate the noise during the detection. Thus, we will use the contour finding and skin detector to find out the reasonable area for the gesture recognition.

Therefore, this project is designed to investigate the relationship between different factor such as the number of the training image set, the training time, stager number, error rate and hit rate in the hand detection and gesture recognition. Also, try to use different methods to increase the accuracy of the hand detection and gesture recognition.
Figure 3a. The overflow of open palm gesture recognition

Figure 3b. The overflow of finger tip detection
4. Methodology

The core technology of this research project is the Viola-Jones detector to achieve hand gesture recognition. However, the Viola-Jones detector exists problems on hand gesture recognition. The detection performance is influenced by background noise of training samples. In this chapter, we try to address the problem of the Viola-Jones detector and proposed a better approach on hand gesture recognition. Before explain the Viola classifier theory, let us describe some concept on Machine learning.

4.1 Machine learning

The aim of machine learning is to turn the data into information. After collect a lot of data, we expect a machine can answer questions about the data such as “Is there a hand in the image?” Thus, machine learning to be able turns data into information by extracting patterns or rules from that data. And the data usually preprocessed into features. For example, 5,000 hand images, run an edge detector on the faces, and then collect feature such as edge direction, strength, and offset from each hand. We obtain a large amount of feature vector and then using this collected data to construct some kind of model such as apply clustering algorithm to identify the hand gesture. Therefore, machine learning algorithms can analyze our collected feature and adjust the parameters such as weights, thresholds to optimize performance according to different goals (Gary & Adrian, 2008). And this parameter adjustment process is called learning. There are contain many machine learning algorithm included in OpenCV and in this paper will focus on the Adaptive Boosting algorithm.
4.1.1 Adaptive Boosting Algorithm

Adaptive Boosting Algorithm also called AdaBoost Algorithm. It is a machine learning algorithm, proposed by Yoav Freund and Robert Schapire in 1999. The original purpose is for the face detection and it is the big step in the face detection field. And it is developed from the Boosting algorithm which contains two learners; one is weak learner and another strong learner. A weak learner is a classifier only slightly correlated with the object. A strong learner is a well correlated with the object. That mean if there is a condition that can estimate slightly to improve the accuracy, this classifier is a weak classifier; if there is another condition that can estimate greatly to improve the accuracy, and it is called strong classifier. The Boost Algorithm can upgrade the weak learner to the strong learner and it is useful for classification, creation of models, image segmentation and data mining. And AdaBoost algorithm is an improvement algorithm from Boosting algorithm. In AdaBoost algorithm it is needs to know the error rate’s lower limit but Boosting algorithms doesn’t need. In addition, AdaBoost algorithm can adjusts the lower error rate according to the weak learning's feedback. Thus, after the AdaBoost algorithm was proposed, it has been applied in many areas because it can enhance the training and learning accuracy and very easily apply it in related areas in the real world. (Teo, 2009)

In the following section, we will describe the AdaBoost training algorithm first and then explain how to use it in Viola and Jones theory.

4.1.1.1 AdaBoost training Algorithm

The main concept of the AdaBoost Algorithm is from a set of weak classifiers and possible to learn from them to create a strong classifier: It trains T weak classifiers \( h_t \), \( t \in \{1,\ldots,T\} \). These classifiers are the decision tree with only one split or a few levels of splits. It also assigned a weight \( \alpha_t \) in each of classifiers at the final decision making process. And the algorithm described as follows (Viola and Jones, 2004)
1. Require: Given example image \((x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots (x_n, y_n)\) where \(y_i = 0\) for negative sample, \(y_i = 1\) for positive sample, \(n\) is number of total training example. \(x_i\) is a feature vectors of the labeled data.

2. Initialize a data point weighting distribution \(D_t(i)\) that tells the algorithm how much misclassifying a data point will cost:

   Initialize weights \(= w_1\),

   For positive: \(D_t(i) = \frac{1}{2l}\) for where \(l\) is number of positive samples.

   For negative: \(D_t(i) = \frac{1}{2m}\) where \(m\) is number of negative samples.

For \(t = 1, ..., T\):
   a. Find the classifier \(h_t\) that minimizes the \(D_t(i)\) weighted error
      \(\epsilon_t = \frac{1}{2} \log \left[ \frac{1 - e_t}{e_t} \right]\), where \(e_t\) is the arg min error from step 2b.

   b. \(h_t = \arg \min_{h \in H} \epsilon_j\), where \(\epsilon_j = \sum_{i=1}^{m} D_t(i) \ (for \ y_i \neq h_t(x_i))\) as long as \(\epsilon_j < 0.5\); else quit

   c. Set h_t voting weight \(\alpha_t = \frac{1}{2} \log \left[ \frac{1 - \epsilon_t}{\epsilon_t} \right]\), where \(\epsilon_t\) is the arg min error from step 2b.

   d. Update the data point weights : \(D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\), where \(Z_t\) normalizes the equation over all data points \(i\).

In step 3b, if we can’t find a classifier with less than a 50% error rate then we quit. This means it is necessary need more features. After the training algorithm, the final strong classifier takes create a new vector \(x\) and classifies it by a weighted sum over the learned weak classifiers \(h_t\):

\[ H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]

Note that the sign function converts anything positive into 1 and anything negative into -1 and zero remains 0.
4.1.1.2 Adaptive Boosting training flow chart

Begin

Input a set of training data

Initialize weights:
For i = 1 to n
If positive then $D_t(i) = \frac{1}{2l}$
If negative then $D_t(i) = \frac{1}{2m}$
END For

For t = 1, ..., T;

Find the classifier $h_t$ that minimizes the $D_t(i)$ weighted error

For each feature $f$ train a classifier

Choose the classifier $h_t$ with the lowest error $\varepsilon$

Update the weights:

Get final strong classifier

End
4.1.1.3 **Viola and Jones Detector**

In 2004, Viola and Jones proposed a famous face detector architecture called Haar-like feature to describe the face. The Haar-like feature is a very simple feature because it compares the intensity between black and white areas.

The Haar-like features used by the classifier are shown in Figure 4.1.1.3.a It has four kinds of features: 1. Edge features; 2. Line features; 3. Center surround features. 4.Special diagonal line feature.

![Figure 4.1.1.3.a Basic four Haar-like features. The image intensity difference between black and white areas is the feature value.](image)

Adding features can increase the accuracy of the AdaBoost machine. Each of the shapes seen in Figure 4.1.1.3.a is scaled and translated anywhere in training images and create the scaling factor. These features have the separating ability between positive and negative sets. For example, the Haar-like feature can represent the face feature very well (Figure 4.1.1.3.b.) The Haar-like feature is very nice to represent the intensity difference between eyes and eyebrows or the intensity difference between eyes and the bridge of nose. Thus, Viola and Jones use these feature to learn their facer detector.

However, it may not provide the satisfying result for the hand feature. In Figure 4.1.1.3.c, the Haar-like feature in the training image generates a small training error. In the image set, left part of the Haar-like feature is darker than the right part of the Haar-like feature. Adaboost algorithm can easy to find the threshold from the positive images
and the weak classifier can show satisfying training error, but the result biased during the real test. If the background is different kind of color, the training result will cause a very high detection error.

Figure 4.1.1.1.3.b The Haar-like features are used on face detection. This figure is cropped from (Viola and Jones, 2004). The first feature measures the difference in intensity between the region of eyes and eyebrows. The second feature measure the difference in intensity between the region of eyes and the bridge of nose.

Figure 4.1.1.1.3.c The Haar-like feature in hand detection. The weak classifier in this image will perform very good result and low error rate, but it will generate a biased result in real testing.
4.1.1.4 Weak Classifier

To choose the best set of the Haar-like feature in image is very difficult. But AdaBoost Algorithm supports an approach to solve this problem. According to AdaBoost Algorithm’s flow chart, it shows that each feature needs to train one classifier. This classifier is called weak classifier. And a Haar-like feature serves as a weak classifier. The aim of the weak classifier is to find out the optimal threshold level. The weak classifier can be trained for every possible feature that obtained in the training image, and the classifier was chosen by the lowest error rate for each round.

The function of the weak classifiers is \( (x, f, p, \theta) \); the parameter of the feature is vector \( x \), \( f \) is the use of feature in that positive training image that contain the gesture we want to recognize and using the Haar-like feature when detect the hand gesture; \( p \) is the sign ( + or - ) which is use to indicating the direction of the inequality:

\[
h(x, f, p, \theta) = \begin{cases} 
1 & \text{if } p f(x) < p\theta \\
0 & \text{otherwise}
\end{cases}
\]

The last parameter \( \theta \) is the threshold; it uses to determine whether a given image can pass a classifier test. (Miloš, 2005)

When feature \( f \) is evaluated on image, it is compared to threshold for categorizing the image by the given feature. For each feature \( f \), it were calculated the entire training sample’s Haar-like features, and then sort the list. The best threshold value can be found by searching the list. For instance, in the figure 4.1.1.3a, the numbers are considered to be the positive set, and the letters are considered to be the negative set. The threshold is set to be the vertical line after the ‘2’. Because after tried all location, the classification error rate at this location is minimal. And the error rate must be smaller than 50%, otherwise it is not accurate enough.

Figure 4.1.1.3.a
After the best threshold is found, then the trained weak classifier is created. The obtained weak classifier can separate and divides the training sample into hands or non-hands such that from the above inequality, we can see that if \( p \frac{f(x)}{p} < p \theta \) is true, function \( h(x, f, p, \theta) = 1 \) and 0 otherwise. If the function is equal 1, it is expected to correspond to positives examples that mean the image contain hand and 0 is the negatives one.

A single weak classifier is not accurate enough to classify the hand or not. Therefore, to combine the weak classifier to produce a strong classifier is necessary. (Miloš, 2005)

4.1.1.5 Strong Classifier

All weak classifiers \( h_t \) are assembled to form a final strong classifier \( H \) by using the AdaBoost algorithm. The strong classifier is linear combination weak classifiers. And it is also called Binary Class Classifiers. Using the strong classifier to detect the image is similar with to ask all of the weak classifier to vote, and calculate the sum of the weak classifier error rate's weight, and then comparing this sum with the average voting result to determine whether it is pass to be the strong classifier. The strong classifier is described as following (Teo, 2009):

Assume it have \( T \) weak classifiers in a strong classifier, labeled \( h_1, h_2, h_3, \ldots, h_T \), and each of classifier is labeled weight labeled \( \alpha_1, \alpha_2, \ldots, \alpha_T \), and \( h_i(x) = h_i(x, f_i, s_i, i) \). Tested image \( x \) is passed through the weak classifiers \( h_1(x), h_2(x), h_3(x), \ldots, h_T(x) \), then each weak classifier assesses if the image passed its test. The assessments are discrete values: \( h_i(x) = 1 \) for a pass and \( h_i(x) = 0 \) for a fail. \( \alpha_i(x) \) are in the range \([0, +\infty]\). To decide these classifies as being positive or negative is made by the following inequality:

\[
\alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_T h_T(x) > \frac{\alpha}{2}, \text{ where } \alpha = \sum_{i=1}^{T} \alpha_i
\]

From this equation, we can see that the result of each weak classifier is multiplied by its confidence rating \( \alpha \). It can represent as below:
And it concluded that if the images $x$ pass a weighted average of half of the weak classifier tests are catalogued as positive, otherwise a negative.

### 4.1.1.6 Cascade of Classifiers

Viola and Jones proposed a training algorithm for building a cascaded detector which can speed up the detection by using a small set of number of the feature to decline most of the negative features. Viola and Jones proposed the cascades of classifiers concept and they have pointed out that it can improve the performance of AdaBoost algorithm.

The basic concept of the cascades of classifiers is set up the cascades layer or stage according to the user’s requirement. Each classifier includes one or more features and for each stage, it may eliminate certain set of false and true of positive training set of sub windows. The incoming particular image is tested on the first classifier and it will not test further if it declared as non-similar. Otherwise, it will test on the next classifier. For example, if the user set up 10 layers, each layer can eliminate greater than 50% of false positives image and lesser than 2% true of positive image. Therefore, the false alarm rate and hit rate can reach at:

False alarm = \(0.5^{10}\)

\[\begin{align*}
&= 0.0009765625 \\
&= 0.098\%
\end{align*}\]
Hit rate = $0.998^{10}$

= 98%

The whole process of the cascade classifier is shown in figure 4.1.1.6.a. When we use this method in training, the computation time is so long and expensive. But it is necessary because the classifier achieved via this method will reach the real-time detection.

Figure 4.1.1.6.a
4.2 Skin color detection

The RGB and HSV color model also can be applied in skin detection, but HSV is more suitable for the skin detection. In RGB format, it can use a combination of Red, Green and Blue components to represent any standard color of brightness. Usually, it stored as a 24-bit number using 8-bit for each color component (0 - 255). For example, black is made of 0 Red, 0 Green, and 0 Blue. However, when RGB model is using in computer vision, RGB values will vary lots of depend on strong or dim lighting conditions or shadows. And HSV model can handle the lighting condition is much better than the RGB. Also, HSV color space is more related to human color perception. (Albiol, Torres and Delp, 2001)

HSV means Hue Saturation Value, where the Hue is the color. Hue is represented as a circular angle value which is within 0.0 to 1.0 and store in floats. For example, a Hue of 0.5 is blue, 1.0 is red which is same as 0.0 and 0.25 is green. Saturation is the grayness, if the saturation value near 0 means it is grey looking. At last, Value is the brightness of the pixel, so 1.0 is white and 0.1 is black, but in OpenCV, 1.0 is not mean the white, it is represented the bright color. In skin detection part, the threshold of H,S and V components is set which is depend on the color, brightness or darkness of the hand object and the background environment.
4.3 Gaussian smooth

Gaussian smooth also called Gaussian blur is used to blur an image by Gaussian function that is already embedded in OpenCV language’s library. This smoothing technique can reduced the image noise and the image’s detail. The visual effect of the Gaussian smooth is a smooth blur resembling that of viewing the image through a translucent screen which shown in figure 4.3.a and figure 4.3.b. Gaussian smooth can be used before any computer vision algorithms like a pre processing stage. If the computer vision algorithm get too detail of the image, then it would induced some noise. For instance, during the finger tip detection, we would like to roughly get the shape of the hand rather than more detail. If applying the Gaussian smooth method before get the shape of the hand, the contour will become more smoothly and roughly. Thus, it can enhance the image structures and avoid over fitting problem. It will discuss more in later in implementation session.

Figure 4.3.a original image

Figure 4.3.a After Gaussian smooth

(Ref: http://blog.csdn.net/hhygcy/archive/2009/07/07/4329056.aspx)
4.4 Calculating the dot product

Ankit and Kumar (2009) suggest calculating the dot product to determine the peaks and valleys in the hand. Let the contour to the hand be $C$, and they suggest that finding the dot product between two vectors $[C_i, C_{i-k}]$ and $[C_i, C_{i+k}]$. And they take a dot product value as 20 and $k$ is 5.

In this project, we will use part of this technique to determine the peaks. That is calculated the size of angle between two vectors. i.e.

$$\theta = \arccos\left(\frac{a \cdot b}{|a||b|}\right)$$

And we would set the range of threshold of angle is 150 to 160 degree. That is different with only calculate the dot product because Ankit and Kumar suggest only focus on the dot product value is 20 which is close to 90 degree. However, both of the peaks and valleys will be find out and it is necessary to determine them in later.

In contrast, if we set the angle to be 150 to 160 degree, only point A and point B would be detected which shown in figure 4.5.a. Thus, it is not necessary to determine which point is in peak or valley.

![Figure 4.5.a. Using the dot product to calculate the angle in the contour for determine the peak](image-url)
4.5 Centralization point

Centralization point is use to find the central point of a set point. The aim of this technique is use to reduce a set point to become one point. It can remove a large amount of noise or unnecessary point during the computer vision process. For example, in figure 4.4.a, the blue points are the original point which is a set of point. Then we take the average value of all x-axis value and the y-axis value. The resulting point is the central point which is in yellow color. The following is the calculation:

The average value of x-axis: \( \frac{1 + 2 + 4 + 6 + 5 + 7}{5} \)
\[ = 4 \]

The average value of y-axis: \( \frac{1 + 4 + 6 + 5 + 2}{5} \)
\[ = 4 \]

The centroid: (4,4)

![Figure 4.4.a The yellow point is the resulting point after Centralization](image)

The following is the general equation of Centralization

\[
\text{Centroid} \left( \frac{\sum_{k=1}^{n} x_k}{n}, \frac{\sum_{k=1}^{n} y_k}{n} \right)
\]

where \( n \) is the number of data points.

- \( x_k \) is the x–coordinate of k-th data points
- \( y_k \) is the y–coordinate of k-th data points
4.6 Finger tip Clustering method

During the finger tip detection, it is important that to find out the noise such as unnecessary point in the hand region. To cope with this problem, we would using the clustering method to separate the noise and the critical point in the hand region. The following is the step of the clustering:

**Step 1:** Assume using the OpenCV internal function to find out all the possible finger tip point that is shown in figure 4.5.a.

![Figure 4.5.a](image)

Figure 4.5.a. The yellow points are the resulting point after using the OpenCV internal function

**Step 2:** line up all the point according the shape of the hand. (Figure 4.5.b)

![Figure 4.5.b](image)
Step 3: Find the shortest distance of all line. And find the centroid of these points that is connecting to the shortest line. (Figure 4.5.c)

Step 4: Grouping these points to become a new group that is connect the shortest line and re-connect the centroid point to become new circuit like figure 4.5.d.
Repeat step 3 and step 4 to find out other group.

Figure 4.5.e It divides into three clusters which are shown in different colors: black, blue and yellow.

Figure 4.5.f. It divides into four clusters which are shown in different colors: black, blue, yellow and light green.
**Step 5:** If the shortest line is connect to one of the central point and it is necessary to recalculate the new central point of these black points. It shown in the Figure 4.5.f and 4.5.g.

Figure 4.5.f. The group of black point is increased due to the shortest line near by the point. And it has to recalculate the new centroid of these three points.

Figure 4.5.g. The new central point is created and the new circuit is generated.
Step 6: If the shortest line is connecting to the two different clusters, then it needs to combine two groups into one cluster. It is also necessary to recalculate the new central point again.

Figure 4.5.h. The shortest line is within two clusters.

Figure 4.5.i. The new cluster is generated which is in blue color. Also, the new centroid is created by four blue points. And then continue to find new shortest line which is in blue color.
Repeat step 3 to step 6 to find out other group.

Figure 4.5.j. It has been divided into three clusters which are in blue, yellow and black color.

Figure 4.5.k. The shortest line is selected and prepare to group the yellow point into black points cluster.
Step 7: If all the points are grouped into two clusters, that means the clustering is finished. All the finger tip points are separated from the surplus points.

Figure 4.5.1. All points are divided into two clusters and the clustering process stops.
4.7 Proximity measure of clusters

The previous one clustering method is not perfect and it is necessary to using proximity measure to calculate the error. In figure 4.6.a, the central point of the blue cluster is more near to yellow cluster than black one. Thus, the shortest line connects within yellow cluster and blue cluster are selected. And then the yellow point combines with the blue cluster (Figure 4.6.b.).

![Figure 4.6.a.](image1)
![Figure 4.6.b. one of the finger tip is miss selected.](image2)

However, the thumb finger tip point is not being selected. Actually, it is belong to the black cluster. Thus, it is need to use sum of square error (SSE) to find out the optimum cluster combination. The following is the detail step of SSE calculation.

**Step 1**: Calculate the Euclidean distance of each point to its closest centroid. (Figure 4.6.c)

**Step 2**: Sum up all Euclidean distance that measure before. And it called the Sum of the square error.
The general formula of SSE is:

\[ SSE = \sum_{i=1}^{K} \sum_{x \in C_i} d(x, c_i)^2 \]

where \( x \) is the point.
- \( C_i \) is the i-th cluster
- \( c_i \) is the central point of cluster \( C_i \)
- \( d \) is the Euclidean between cluster \( C_i \) and the data point \( x \).

**Step 3:** After calculate the original one cluster’s SSE, and then try to put another cluster’s point to opposite cluster to test the SSE. For instance, take one of the point in blue cluster into black cluster and calculate the SSE.

![Image](image.png)

Figure 4.6.c. Take one point in blue cluster to the black cluster. Then add up all the Euclidean distance of each point to its closest centroid to calculate the SSE.

**Step 4:** Compared the SSE to previous one, i.e. the original set of cluster. If the SSE is smaller, then take as this set of cluster to become the optimum one.

**Step 5:** Continues to calculate the SSE and repeat Step 3 and step 4 until try all the point from the another cluster. And then take the smallest SSE of the set of cluster to
become the optimum one. For example, in figure 4.6.d, assume the SSE is the smallest one, and then it will replace the original set of cluster.

Figure 4.6.d. the point in thumb become a black cluster from the blue cluster due to this set cluster’s SSE is smallest in all combination of set of clusters.
5. Implementation

In this session, it would be divided into two parts. The first part is described the detail implementation of the gesture recognition which is within session 5.1 to 5.4. And the second part is the finger tip detection which is within after session 5.4.

5.1 Preparation of the cascade classifier

The hand detection and gesture recognition’s basic operation’s basic operation describes as follows,

1. Training image preparation
2. Training Image Segmentation
3. Create training sample
4. Create strong cascade classifiers

5.1.1 Training image preparation

In the data preparation, we need to prepare a large amount of images which contain hand gesture for training. A large amount of image is the pre-requisite to ensure the accuracy of hand detection. For the hand detection, it is necessary to collect a large amount of the hand gesture image. In this report, it use palm gesture as gesture recognition. For the positive image that is the image contain palm gesture image and all of them are capture by the web camera (Figure 5.1.1.a). For the negative image, it represent the negative image, it also can be collected by the web cam and through the internet. (Figure 5.1.1.b)

Figure 5.1.1.a the example of the positive image captured by the webcam
Figure 5.1.1.b The example of negative image. And the size of the negative should be larger than the size of the positive image.

It is necessary to taking the positive image very effective and fast to save the time due to the amount of the positive image is very large. So, we write a program using Open CV to capture open palm hand gesture image via the digital web camera. And all the background is used a pure color such as black or green color for preparing the segmentation stage. After capture the image from the web camera, it is necessary to us a program to find out the boundary of the hand and cut out the unused part that can greatly reduced the training error. The example is shown in Figure 5.1.1.c

Figure 5.1.1.c All the unused part is cut out.
5.1.2 Training Image Segmentation

In this project, we would try to use two set of positive image in training process. The first one is the positive sample with solid background and the second one is the positive sample with random background. The aim of separate two set of positive image is to test what kind of background in the image can obtain the best performance of the training result.

5.1.2.1 Methods of separating background

To separate the background and the object, it would be used color model theory. In the color model based methods, we consider using RGB color models rather than HSV model. The reason is using OpenCV to obtain HSV image, it will be loosed some of the color resolution due to it is only storing the Hue as a 7 bit number instead of an 8 bit number.

5.1.2.2 Training image with solid background

In this set of training image, all positive images should be with the pure solid image. However, during capturing the positive image sample, the object and background cannot the separate rigidly due to the lighting condition (Figure 5.1.2.1.a). Therefore, it is necessary to apply a threshold operation to segment the object and the background in the training image and then the change the background to pure black color. It is used darker color as the background color better than other color such as yellow because we have compared the result of the after the segmentation of the original image, using more darker is more easy to separate the hand gesture and the background color. After the segmentation, the hand and the background restrict partitioned (Figure 5.1.2.1.b).
Figure 5.1.2.1.a. Some of the training image contain the shadow that will influence the performance of the classifier.

Figure 5.1.2.1.b. The left side is the original training image; After segmentation, the hand and the background restrict partitioned.
5.1.2.3  Training image with random background

Another set of the training image, all image have to capture with random background. If we capture the hand gesture with different background for each image, it is time consuming. Thus, a program for changing the background of the positive image is created. In that program, it can apply many different images that have not contains the hand to become a background of the positive image. For instance, according the size of the hand image, we need choose a background image (Figure 5.1.2.3.a) which is greater than the hand image. And then find the non-hand color pixels in the image and replace those respondent pixels. At last, a hand image with the random background is completed (Figure 5.1.2.3.b). All background images cannot be used as a negative training sample because if the positive training images include the negative training images’ object, the false positive rate would be increased that would extremely affect the training result. Be more seriously, the training progress will be terminated.

![Image of penguins](image.png)

Figure 5.1.2.3.a The background image used in this section cannot be used as a negative training image.
5.1.3 Create training sample

After finished the segmented image, it is needed to use the OpenCV functions to create samples to prepare training the classifiers. Before create the sample, we need to some preparation jobs.

5.1.3.1 Preparing the Description file

Before starting to train the classifiers, we need to gather a data set consisting of the training samples to describe the path of the samples, the position of the object in the image and the size of the image. It is called the description file. Both negative and
positive training image need to create description file, but it is slightly different between the positive and negative image. It will discuss more in the following section.

5.1.3.1.1 Description file for positive image

For the description file of the positive image, all positive training images may be stored in one or more directories that should be indexed by a text file in the following format (Gary & Adrian, 2008):

\[
\text{<path>/image_name count_object x y width height sample_ width sample_ height}
\]

The first parameter in the description file is the path of the positive image file, the second parameter is indicate the number of the object that contain in the image, the third and fourth parameter is describe the coordinator of the object that in the image. And the fifth and sixth parameter is the image size. For example:

<table>
<thead>
<tr>
<th></th>
<th>open_palmImage01.jpg 1 120 100 150 150</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>open_palmImage02.jpg 1 0 0 20 20</td>
</tr>
<tr>
<td>2</td>
<td>open_palmImage02.jpg 1 0 0 20 20</td>
</tr>
</tbody>
</table>

In the first example, file name is \textit{open_palmImage01.jpg}, the coordinate of the object is 120 100, and the image size is 150*150.

In the second example, the file name is \textit{open_palmImage02.jpg}, the beginning coordinator is 0 0, that mean the whole image is the object. And the width and height of the image also is 20.

The description file can be named sample.txt or sample.dat, both of them is the same. Viola and Jones pointed out that the optimal size of the sample image is 20*20. However, if the size of our detected object (open palm gesture) proportion to square i.e. 20*20, it will contain a lot of the background noise or the original size will be sink (Figure 5.1.3.1.1.a) Therefore, the sample size is necessary directly proportional to reduce from the original size of the image. For instance, if the image size is 300*400, then the sample size can be reduced to 20*26 (Figure 5.1.3.1.1.b). Although the sample size is not a square, it can reduce the background noise and maintain the proportion of the hand gesture.
5.1.3.1.1a If the training image are set to the optimal size 20*20, then the hand object would be deformation

5.1.3.1.1.b The proportional size of the positive training sample is fit to the original size of the image.

5.1.3.1.2 Description file for negative image

For negative image file, it is similar to create the description file for the positive ones. It only contains one parameter which is the path of the negative image file, for example:

```
negativeImage.jpg
```

Actually, the negative image description file is only for the training classifiers part, but for the convenience, we suggest create it in this stage. In addition, user has to pay attention for the unused file name in the description file for negative or the positive one. Because when using this command i.e. “dir /b /s > negative.txt” to create description file, the name of the description file “negative.txt” also include into the text file. Thus, user has to remember to delete it due to description file is only include the negative or positive training image.

5.1.3.2 Creating the training sample

After finishing the description file, it is time to use this file to create sample for training. There are some functions OpenCV provided for creating sample, the relative command is introduced as follows: createsamples : (Gary& Adrian, 2008)

Usage: ./createsamples

1. **-info** <description_file_name>
   The directory of the description file that is listed all positive training images.

2. **-img** <image_file_name>
   The source of the positive image, usually using –info is enough.
3. **-vec** `<vec_file_name>`
   The output file name of the sample image which used for training.

4. **-bg** `<background_file_name>`
   The background description file (the negative sample set). It contains a list of images into which randomly distorted versions of the object are pasted for positive sample generation.

5. **-num** `<number_of_samples>`
   The number of positive training sample. The default is 1000.

6. **-bgcolor** `<background_color>`
   The background color (currently grayscale images are assumed);
   The background color denotes the transparent color.
   The default is 0.

7. **-bgthresh** `<background_color_threshold>`
   The amount of color tolerance and the background color threshold.

8. **-inv**
   Invert the colors.

9. **-randinv**
   Invert the colors randomly.

10. **-maxdev** `<max_intensity_deviation>`
    The desired maximum intensity deviation of foreground samples pixels. The default is 40.

11. **-maxxangle** `<max_x_rotation_angle>`
    The maximum rotation angle in x-direction in radians. The default is 1.1.

12. **-maxyangle** `<max_y_rotation_angle>`
    The maximum rotation angle in y-direction in radians. The default is 1.1.

13. **-maxzangle** `<max_z_rotation_angle>`
    The maximum rotation angle in z-direction in radians. The default is 0.5.

14. **-show** `<scale_factor>`
    Show each created sample during the creation process. Optionally a scaling factor can be defined. The default is 4.0.
If <ESC> is pressed, the creation process will continue without showing the samples. This can be useful for debugging purposes.

15. -h <sample_height>
The height of the training samples. The default is 24.

16. -w <sample_width>
The width of the training samples. The default is 24.

For instance:

```
createsamples.exe -info positivesImage/positives.txt -vec data/positives.vec -num 1500
-w 20 -h 26 -show
```

The above command is use “positive.txt” as the positive sample description file which under “positivesImage” directory, then creates the sample called positives.vec and store into “data” directory, the size of the sample is 20*26. The total number of positive image is 1500.

To complete the open palm gesture recognition, it has to create a sample .vec file, which stores all positive samples.

5.1.4 Create strong cascade classifiers

After creating the training sample, it is time to train the cascade classifiers. OpenCV provided haartraining function to train the cascade classifiers. And the OpenCV haartraining supports the following options (Gary& Adrian, 2008):

Usage: ./haartraining

1. -data <dir_name>
The trained classifier is stored in which directory.

2. -vec <vec_file_name>
The file name of the positive samples file which is created by the opencv-createsamples.

3. -bg <background_file_name>
The description file of the negative sample set. It contains a list of negative images which is randomly extracted during the cascade training.

4. -npos <number_of_positive_samples>
   The number of positive samples used in training of each classifier stage.
   The default value is 2000.

5. -nneg <number_of_negative_samples>
   The number of negative samples used in training of each classifier stage.
   The default value is 2000.

6. -nstages <number_of_stage>
   The number of classifier stages to be trained. The default is 14.

7. -nsplits <number_of_splits>
   Determine the weak classifier used in stage classifiers. The default is 1.

8. -mem <memory_in_MB>
   Available memory for pre-calculation during training process. More memory you have been allocated, the training process will faster. The default is 200.

9. -sym, [-nonsym]
   Specify whether the object under training has vertical symmetry feature or not. For example, frontal faces show off vertical symmetry feature. Vertical symmetry speeds up training process and reduces memory usage.
   The default is -sym.

10. -minhitrate <min_hit_rate>
    The minimum desired hit rate for each training classifier stage. Overall hit rate may be estimated as min_hit_rate^number_of_stages. The default is 0.950000.

11. -maxfalsealarm <max_false_alarm_rate>
    The maximum desired false alarm rate for each training classifier stage. Overall false alarm rate may be estimated as max_false_alarm_rate^number_of_stages.
    The default is 0.500000.

12. -weighttrimming <weight_trimming>
    Specifies whether and how much weight trimming should be used.
    The default is 0.950000. A decent choice is 0.900000.
13. -eqw  Specify if initial weights of all samples will be equal.
14. -mode [BASIC | CORE | ALL]
   Select the type of haar features set used in training. BASIC uses only upright
   features, while CORE uses the full upright feature set and ALL uses the full
   set of upright and 45 degree rotated feature set. The default is BASIC.
15. -bt [DAB | RAB | LB | GAB ]
   The type of the applied boosting algorithm. You can choose between
   Discrete AdaBoost (DAB), Real AdaBoost (RAB), LogitBoost (LB) and Gentle
   AdaBoost (GAB). The default is GAB.
16. -err [misclass | gini | entropy]
   The type of used error if Discrete AdaBoost (-bt DAB) algorithm is applied. The
   default is misclass.
17. -maxtreesplits <max_number_of_splits_in_tree_cascade>
   The maximum number of splits in a tree cascade. The default is 0.
18. -minpos <min_number_of_positive_samples_per_cluster>
   The minimum number of positive samples per cluster. The default is 500.
19. -h <sample_height>
   The training sample height (The value must same as the value of the training
   sample during creation). The default is 24.
20. -w <sample_width>
   The training sample width (The value must same as the value of the training
   sample during creation). The default is 24.

In this project, we would use different command to find out what are parameters can
perform better performance of the classifier. One of the training the strong classifier’s
command is shown as following:

```
haartraining.exe -data data/data -vec data/positives.vec -bg negatives/negatives.txt -npos 500 -nneg 500 -nstages 20 -mem 3000 -mode ALL -w 20 -h 32 -minhitrate 0.9995 -nonsym
```
From the above command, it set the final classifiers is generated under the “data” directory, and the positive sample is called “positives.vec” which is under the “data” directory. And the description file of the negative training image is called “negatives.txt” which is under “negatives” directory.

And we set the number of positive samples used in training of each classifier stage is 500 that is not equal to the total amount of the positives training sample. Also, the number of negative samples used in training of each classifier stage is defined as 500 that are not equal to the total amount of the negative training sample.

The training classifier stage is set to be 20. In order to increase the training speed, 3000MB pre-calculation memory is allocated during the training process. And then the type of Haar-features is the full set of upright and 45 degree rotated feature set. The number of training sample height and width is same as the size of the training sample that created before. The minimum hit rate is set to 0.9995 instead of default value 0.9500. At last, because of the hand is not symmetry, so we use parameter “nonsym” to specify the object has not contain vertical symmetry feature.

After finished Haartraining command, the Haartraining generates a cascaded stronger classifier and the data type is “.xml”. In Figure 5.1.4.a, we can see that the result of the open palm gesture recognition by using the strong classifier. The performance can be improved, it would discuss more about in later section.

We not only try to use different number positives and negative samples used in training of each classifier stage, but also the number of stage and Haar-features mode to find out how to create the best performance of the hand recognition’s classifier.

![Image](image.png)

Figure 5.1.4.a The recognition performance can be improved further more
5.2 Skin color detection

After using the strong classifier for the gesture recognition, we found that some of the non-hand objects were detected by the strong classifier. Therefore, we need using the some methods to reduce the noise. Using the skin color detection find out the extracted recognition for gesture recognition.

In order to using the HSV model, we need to convert the image which is captured by the webcam from BGR to HSV format first. And then separate the HSV color image into Hue(H), Saturation(S) and Brightness(V) channels. (Figure 5.2.a) After that, using the OpenCV function “cvThreshold” to find out the pixels that are in the correct range in H, S and V. For example, we can set the range of the H channel from 0 to 30 that are closed to the human skin. About the S and V channel, adjust the value depends on the environment. If Saturation is below 30, it means the environment is too dull. To determine the threshold of the skin color, we need to use specific graphics software for the OpenCV to roughly get the Hue value which is similar to the human skin. (Figure 5.2.b) In the figure, 5.2.c, it has been shown the hand which is detected by skin detection and fill it into the white color. It assume that the skin color has a Hue value between 0 to 18 (out of 180), and Saturation value above 50, and Brightness value above 80.

Figure 5.2.b. The HSV graphic software for OpenCV which produced by Shervin Emami.
Figure 5.2.a The hand image which is split into HSV channel.

Figure 5.2.c The result of the skin color detection. Assume that the skin color has a Hue value between 0 to 18 (out of 180), and Saturation value > 50, and Brightness value > 80.
5.3 **Contour finding**

After finding the skin color pixel, it is time to summarize the skin color pixel into a region by contour method.

Contour is a list of points that can find the edge of the pixel, “Contours are represented in the OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve” (Gary and Adrian, 2008). OpenCV has a method to summarize the contour pixel information with a bounding box, circle or ellipse. (Figure 5.3.a)

![Figure 5.3.a](image)

Figure 5.3.a After using the contour finding within the skin colour pixel and the boundary rectangle is drawn.

5.4 **Integration of different analysis methods**

After drawing the boundary within the skin color pixel, it also need to combine the OpenCV function region of interest(ROI) to allow the strong classifier only work within these area. If only using the strong classifier for the gesture recognition in real time, these are contains several unnecessary detection which shown in Figure 5.4.a.

On the other hand, if integrating different analysis methods, the result will be better (Figure 5.4.b). On the left hand side windows in figure 5.4.b, the skin color pixels are bounded by the drawing rectangle. In addition, if the boundary area is too small, it would be eliminate before the recognition. On the right hand side windows, we can see
that the strong classifier can identify the open hand gesture correctly because the classifier only work in the rectangle box area that is shown on the left hand side. (The overall design is shown in figure 5.4.d.)

Therefore, the calculation time and memory and the error rate is strictly decreased due to the detection area is reduced. Also, accuracy of the gesture recognition is increased. The counter example is shown in 5.4.c. If the gesture is not open palm, the hand would not detected by the classifier.

Figure 5.4.a The screen shot of the real time gesture recognition only using to strong classifier. There are contain several mislead detection.

Figure 5.4.b The left hand side windows is the result of integrate the contour finding and skin detection methods ; The right side windows is shown the result of the strong classifier which is only work on the drawing boundary box in the left hand side window.
Figure 5.4.c The counter example of the real time open palm gesture recognition
Figure 5.4.d The overall flow chart of the gesture recognition (include classifier training part)
5.5 The Finger Tip detection

The finger tip detection is achieved by integration of several methods. The following is implementation of the finger detection. And the overall flow of finger tip recognition is shown in Figure 5.5.a. which is at the end of this session.

5.5.1 Skin classifier

Using the skin detection method obtain the hand region which is mentioned before in the gesture recognition session. It is shown in Figure 5.5.1.a.

![Figure 5.5.1.a. The yellow part is one of the examples of the spination feature that necessary to reduce.](image)

In order to reduce the spination of the hand, it is needed to apply the Gaussians smoothing. It is shown in Figure 5.5.1.b. The shape of the hand is become smoother than before.

![Figure 5.5.1.b. The effect of applying Gaussians smoothing](image)
5.5.2 Contour finding and region of interest in finger tip detection

After apply the Gaussians smoothing to blur the skin color pixel, it is time to group them into a rectangle region by the contour method that is same as the contour finding part in gesture recognition session.

In the figure 5.5.2.a, the contour is enclosed by the red rectangle which created by the OpenCV function. And we set the region of interest to this rectangle, that mean we only will focus in this region for any image processing. The advantage of this technique is that can ignore any noise outside of the region. That can reduce the false positive rate or false alarm rate.

![Figure 5.5.2.a](image)

Figure 5.5.2.a. It would only focus inside the red rectangle region for any image processing in the future. It can reduce a lot of wrong detection.

5.5.3 Compute the convexity point of the hand

5.5.3.1 Using OpenCV internal function

When the region of interest is set up, it is time to find the convexity point of the contour. The OpenCV internal function is called “cvConvexHull2()” which use to find out the contour convexity in shape of an object. (Gary and Adrian, 2008) In Figure 5.5.3.b, 5.5.3.c and 5.5.3.d, all the yellow points are generated by the “cvconvexHull2” function. We can see that all the data points are not very accurate point to the finger tip. They
contain so many noise and wrong detection. Therefore, it is need to apply the centralization method and clustering method to eliminate the unnecessary data points.

5.5.3.2 Algebraic Operation

The alternative method to extract the peak points are calculated the angle by dot product between these points. The result is shown in figure 5.5.3.e. The yellow points represent the peak in the contour of the hand. Moreover, we can see that these data points are not very accurate point to the finger tip. They contain so many noise and false alarm detection. Therefore, it is also need to apply the centralization method and clustering method to eliminate the unnecessary data points.

We will compare the performance of using this method and using OpenCV internal function to see which method is better in finger tip detection. The detail will be discussed in session 6.2.3.
5.5.4 Centralization in hand region

In order to reduce the false detection points after finding out the convexity points, we would apply the centralization method to centralize the data points. It takes the average value of the x-coordinate and y-coordinate to generate the centroid of a cluster. In figure 5.5.3.d, it contains so many data points in each finger tip, so it has to centralize all the points in each finger tip for increasing the accuracy of the detection. In the figure 5.5.4.a and 5.5.4.b, they are shown the result of the fingertip detection after doing the centralization progress. However, there are still contain the unnecessary data points, it haven’t been eliminated totally. Thus, we would separate the fingertip data points and the noise in the next session.

Figure 5.5.4.a two finger detection after doing the centralization progress.

Figure 5.5.4.b five finger detection after doing the centralization progress.
5.5.5 Clustering the finger tip point

In this session, we will use the finger tip clustering method to separate the actual finger tip data points and the unnecessary data points. After using the clustering method that mention in chapter 4.5 (Methodology), the results are shown in the following figure, the black points are indicated the finger tip points and the white data points are indicated the noise.

Figure 5.5.5.a one finger detection after doing the finger tip clustering method.

Figure 5.5.5.b two finger detection after doing the finger tip clustering method.

Figure 5.5.5.c three finger detection after doing the finger tip clustering method.

Figure 5.5.5.d four finger detection after doing the finger tip clustering method.
5.5.6 Proximity measure in finger tip detection

Although most of the unnecessary data points are eliminated, the clustering method sometime does not treat the thumb as the set of finger tip. For instance, in figure 5.5.6.a, the black points are the finger tip points, and the white points treat as the noise. We can see that the thumb has not been group into finger tip set. Therefore, using the Sum of the square error can found out the optimum solution of the clustering. The result is shown in figure 5.5.6.b; the color of data point in thumb is changed to the black color. That mean it treat as the finger tip set, so the five finger tip detection is successes.
Figure 5.5.a The overall flow chart of the gesture recognition (include classifier training part)
6. Results

6.1 Gesture recognition

6.1.1 Description

The purpose of gesture recognition not only recognizes the gesture, but also investigates the relationship between different factors during the training process. The factors include the number of the positive image and negative image, the hit rate, error rate, number of stage, number of weak classifier and time. In addition, the data collection approach suggests that capture the training image randomly in different light condition and various backgrounds (Anton-Canalis and Sanchez-Nielsen, 2006). About this statement, we also would use the positive image with pure background and with random background to train the classifier respectively and find out which one is better.

In order to evaluate and test the performance of the classifier that we generated, we will use the classifier performance utility that include in OpenCV. The target of the testing image is exactly is the positive image that use to generated the classifier. Thus it can test the performance of the positive image data set. The core command of performance utility is introduced as follows: (Gary& Adrian, 2008)

Usage: ./ performance

1. -data <classifier_file_name>
   The directory of the classifier file that is stored

2. -info <collection_file_name>
   The description file the testing image.

3. -w < sample_width >
4. -h < sample_height >
   The size of the testing samples that must exactly the same values as the used during training

5. –ni
   It is use to create the resulted image files during testing the performance. The detected object will be marked if the classifier hit the target.

Let us take an example of using the performance utility.
The resulting classifier name is called “open1300.xml”, the description file under positive2\ positives2 and called positive.txt. And the resulted image will save under positives2 directory.

After executed the performance utility, it will output the performance result. It includes the number of hit objects, number of missed objects, false alarms, the number of stages and weak classifiers. A sample of the output of the performance utility is in figure 6.1.1.a

![Performance utility output example](image)

Figure 6.1.1.a The number of correct detection is shown in “Hits”, The number of missed object is shown in “Missed”. The “False” column shows the number of false alarms

After find out the optimum strong classifier, we would use this classifier to test another testing data and compare the result that integrating with the contour finding and skin detection method that we mention before. To evaluate how these methods can improve the strong classifier.
6.1.2 Result and findings

The result provided by performance utility would divide into two main parts. The first part is test the classifiers that generated by the positive image with random background. The second part is test the classifiers that created by the positive image with solid background.

In table 6.1.2.a show the result of the classifier performance with trained by random color background image. And In table 6.1.2.b also show the result of the performance but the classifier is trained by solid color background image.

<table>
<thead>
<tr>
<th>No. of positive image</th>
<th>No. of negative image</th>
<th>Hits</th>
<th>Missed</th>
<th>False alarm</th>
<th>No. of Stage</th>
<th>Time (hr)</th>
<th>No. of weak classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>6000</td>
<td>282</td>
<td>2718</td>
<td>19</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2000</td>
<td>4000</td>
<td>144</td>
<td>1856</td>
<td>32</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1900</td>
<td>3800</td>
<td>360</td>
<td>1594</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>1800</td>
<td>3600</td>
<td>314</td>
<td>1486</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1700</td>
<td>3400</td>
<td>168</td>
<td>1532</td>
<td>1523</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1600</td>
<td>3200</td>
<td>330</td>
<td>1270</td>
<td>621</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>1500</td>
<td>3000</td>
<td>316</td>
<td>1184</td>
<td>28</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1400</td>
<td>2800</td>
<td>74</td>
<td>1324</td>
<td>126</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1300</td>
<td>2600</td>
<td>35</td>
<td>1265</td>
<td>41</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1200</td>
<td>2400</td>
<td>0</td>
<td>1200</td>
<td>158</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1100</td>
<td>2200</td>
<td>6</td>
<td>1094</td>
<td>132</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>30</td>
<td>970</td>
<td>827</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 6.1.2.a the result of classifiers which are trained by random color background images

<table>
<thead>
<tr>
<th>No. of positive image</th>
<th>No. of negative image</th>
<th>Hits</th>
<th>Missed</th>
<th>False alarm</th>
<th>No. of Stage</th>
<th>Time (hr)</th>
<th>No. of weak classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>6000</td>
<td>1575</td>
<td>1425</td>
<td>913</td>
<td>6</td>
<td>4</td>
<td>27</td>
</tr>
<tr>
<td>2000</td>
<td>4000</td>
<td>1272</td>
<td>728</td>
<td>637</td>
<td>8</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>1900</td>
<td>3800</td>
<td>1168</td>
<td>732</td>
<td>527</td>
<td>7</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>1800</td>
<td>3600</td>
<td>1108</td>
<td>792</td>
<td>1133</td>
<td>5</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>1700</td>
<td>3400</td>
<td>1156</td>
<td>544</td>
<td>248</td>
<td>6</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>1600</td>
<td>3200</td>
<td>1083</td>
<td>517</td>
<td>91</td>
<td>7</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>1500</td>
<td>3000</td>
<td>857</td>
<td>643</td>
<td>544</td>
<td>5</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>1400</td>
<td>2800</td>
<td>938</td>
<td>462</td>
<td>172</td>
<td>9</td>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>1300</td>
<td>2600</td>
<td>1020</td>
<td>280</td>
<td>337</td>
<td>6</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>1200</td>
<td>2400</td>
<td>670</td>
<td>530</td>
<td>22</td>
<td>9</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>1100</td>
<td>2200</td>
<td>619</td>
<td>481</td>
<td>254</td>
<td>8</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>664</td>
<td>336</td>
<td>166</td>
<td>8</td>
<td>2</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 6.1.2.b the result of classifiers which are trained by solid color background images
The accuracy (Hit rate) of the classifier:

\[
\text{Hit rate} = \frac{\text{The number of correct detected}}{\text{Total number of testing image}}
\]

The false alarm rate of the classifier:

\[
\text{False alarm rate} = \frac{\text{The number of false positive image}}{\text{Total number of testing image}}
\]

The following tables are shown the accuracy and false alarm rate of the classifiers which trained by random color or solid color background image:

<table>
<thead>
<tr>
<th>Classifier ID</th>
<th>No. of positive image</th>
<th>Hit rate (%)</th>
<th>False alarm rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR10</td>
<td>1000</td>
<td>3</td>
<td>82.7</td>
</tr>
<tr>
<td>CR11</td>
<td>1100</td>
<td>0.5454545454</td>
<td>12</td>
</tr>
<tr>
<td>CR12</td>
<td>1200</td>
<td>0</td>
<td>13.16666667</td>
</tr>
<tr>
<td>CR13</td>
<td>1300</td>
<td>2.692307692</td>
<td>3.153846154</td>
</tr>
<tr>
<td>CR14</td>
<td>1400</td>
<td>5.285714286</td>
<td>9</td>
</tr>
<tr>
<td>CR15</td>
<td>1500</td>
<td>21.06666667</td>
<td>1.866666667</td>
</tr>
<tr>
<td>CR16</td>
<td>1600</td>
<td>20.625</td>
<td>38.8125</td>
</tr>
<tr>
<td>CR17</td>
<td>1700</td>
<td>9.882352941</td>
<td>89.58823529</td>
</tr>
<tr>
<td>CR18</td>
<td>1800</td>
<td>17.44444444</td>
<td>0.222222222</td>
</tr>
<tr>
<td>CR19</td>
<td>1900</td>
<td>16.10526316</td>
<td>0.263157895</td>
</tr>
<tr>
<td>CR20</td>
<td>2000</td>
<td>7.2</td>
<td>1.6</td>
</tr>
<tr>
<td>CR30</td>
<td>3000</td>
<td>9.4</td>
<td>0.633333333</td>
</tr>
</tbody>
</table>

Table 6.1.2.b The Hits rate of the classifiers trained by random background positive images.

<table>
<thead>
<tr>
<th>Classifier ID</th>
<th>No. of positive image</th>
<th>Hit rate (%)</th>
<th>False alarm rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS10</td>
<td>1000</td>
<td>66.4</td>
<td>16.6</td>
</tr>
<tr>
<td>CS11</td>
<td>1100</td>
<td>56.27272727</td>
<td>23.09090909</td>
</tr>
<tr>
<td>CS12</td>
<td>1200</td>
<td>55.83333333</td>
<td>1.833333333</td>
</tr>
<tr>
<td>CS13</td>
<td>1300</td>
<td>78.46153846</td>
<td>25.92307692</td>
</tr>
<tr>
<td>CS14</td>
<td>1400</td>
<td>67</td>
<td>12.28571429</td>
</tr>
<tr>
<td>CS15</td>
<td>1500</td>
<td>57.13333333</td>
<td>36.26666667</td>
</tr>
<tr>
<td>CS16</td>
<td>1600</td>
<td>67.6875</td>
<td>5.6875</td>
</tr>
<tr>
<td>CS17</td>
<td>1700</td>
<td>68</td>
<td>14.58823529</td>
</tr>
<tr>
<td>CS18</td>
<td>1800</td>
<td>61.55555556</td>
<td>62.94444444</td>
</tr>
<tr>
<td>CS19</td>
<td>1900</td>
<td>61.47368421</td>
<td>27.73684211</td>
</tr>
<tr>
<td>CS20</td>
<td>2000</td>
<td>63.6</td>
<td>31.85</td>
</tr>
<tr>
<td>CS30</td>
<td>3000</td>
<td>52.5</td>
<td>30.43333333</td>
</tr>
</tbody>
</table>

Table 6.1.2.c The Hits rate of the classifiers trained by solid background positive images.
6.1.3 Discussion and Analysis

In this session, we would discuss the relationship of different factors of the classifiers such as the relationship of hit rate and the number of positive training images. And then find out the most effective classifier by some mathematic methods. The details are shown as following:

![The ratio of the number of positive and negative image](chart)

**Chart 6.1.3.a**

The line graph 6.1.3.a show us that the relationship between the amount of positive and the negative image. The number of the negative image is exactly twice to the positive image. According to the training algorithm, the number of negative images should be as more as possible, but the training time will become increase. Rhonda suggests that the proportion of the positive image and negative images should be 1:2. Therefore we would train the classifier with positive and negative images in ratio 1:2. For instance, if the amount of negative image is at 6000, the number of positive image is at 3000.
The line graph 6.1.3.b compares the correct detection images between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image. The hit rate (RB) increased gradually to 21% by 1500 positive images and then dropped to 9.4% at 3000 positive images.

In contrast, the hit rate (SB) at 1000 positive image is 66.4% that is massive greater than hit rate (RB) at same level. In addition, the hit rate (SB) is strictly increasing to the 78% at 1300 positive images. And then the hit rate (SB) decreased slowly to 52% at 3000 positive images.

In conclusion we can see that the overall hit rates (SB) are extremely greater than hit rate (RB). And the maximum hit rate (SB) is 78% and the particular classifier is trained by 1300 positive images. And it seems to be no strong relationship between the number of positive image and the hit rate due to the hit rate does not contain proportional to the amount of positive image.
The line graph 6.1.3.c compares the error rate between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image. The error rate (RB) decreased gradually to minimum point 78% by 1500 positive images and then climb to 91% at 3000 positive images.

In contrast, the error rate (SB) at 1000 positive image is 33.6% that is massive smaller than error rate (RB) at same level. In addition, the error rate (SB) is strictly decreasing to the 22% at 1300 positive images. And then the hit rate (SB) increased gradually to 48% at 3000 positive images.

In conclusion we can see that the overall error rates (SB) are extremely smaller than error rate (RB). And the minimum error rate (SB) is 22% and the particular classifier is trained by 1300 positive images. And it seems to be no strong relationship between the number of positive image and the error rate due to the error rate does not contain proportional to the amount of positive image.
The line graph 6.1.3.d compares the false alarm rate between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image.

The false alarm rate (RB) decreased rapidly to 12% by 1100 positive images and then grew up dramatically to 90% at 1700 positive images and drop down quickly below 1% at 1800 positive image and hold steady to the 3000 positive image.

On the other hand, the false alarm rate (SB) is dramatically up and down by 1900 positive images. In generally, the trend line illustrate the false alarm rate (SB) is increased gradually by 3000 positive images.

To sum up, the trend line show that the false alarm rate (RB) is indirectly proportional to the number of positive image. But the false alarm (SB) is directly proportional to the number of positive image.
The line graph 6.1.3.e compares the training stage between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image. The number of training stage (RB) is keep steady at 4 by 1700 positive images and then it increase gradually to 7 at 3000 positive images. In contrast, the trend line of number of training stage (SB) show that it decreased gradually by 3000 positive image.

In conclusion, the trend line shows that the training stage (SB) is indirectly proportional to the number of positive image. But the training stage (RB) is directly proportional to the number of positive image.
The line graph 6.1.3.f compares the training time between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image.

The training time (SB) keeps steady in 2 hour to the 1700 positive images. And it increases to about 4 hours from 1800 to 3000 positive images. On the other hand, the training time (RB) is strictly increased to 8 hours by 1500 positive images and slightly decline to 6 hours.

In generally, the trend lines show that both training time is slightly increased by 3000 positive images. But the training time (RB) is longer than training time (SB) if this pattern is continues.
The line graph 6.1.3.g compares the number of weak classifier between two type classifiers which are trained by random background color (RB) and solid background color (SB). The number of positive image represent different classifiers that trained by corresponding amount of positive image.

The number of weak classifiers (RB) is reminded steady by 3000 positive image. In addition, the number of weak classifiers (SB) is dramatically in to 50 approximately from 1300 to 1400 positive images and then it return to 20 at 1500 positive images.

Overall, the number of weak classifiers (SB) is larger than another one. The trend line show that the both of the amount of the weak classifiers keep stable even the number of the positive image is increased.
6.1.3.1 The optimum classifier

According to line charts we mention before, the performance of the classifiers (RB) that trained by random background positive images are not satisfied compare with the classifier (SB) that trained by solid background positive images. Therefore, we will focus more on the performance of classifiers (SB).

In Chart 6.1.3.1.a, the maximum hit rate is 78% but the corresponding false alarm rate is a little bit high. Therefore, it is necessary find out the optimum hit rate and false alarm rate. The following is the procedure of finding out the optimum classifier.
1. Calculated the mean of the hit rate and false alarm rate.

The mean of hit rate

\[
\frac{\sum_{i=1}^{n} x_i}{n} \quad \text{where } n \text{ is the total number of hit rate, } x \text{ is the value of hit rate}
\]

\[= 62.99\%\]

The mean of false alarm rate

\[
\frac{\sum_{i=1}^{n} y_i}{n} \quad \text{where } n \text{ is the total number of false alarm rate, } x \text{ is the value of false alarm rate}
\]

\[= 24.10\%\]

2. Filter out the classifiers which the value of hit rate is below the mean or the value of false alarm rate is above the mean. The following table is show the filter process of the classifiers.

<table>
<thead>
<tr>
<th>Classifier ID</th>
<th>No. of positive image</th>
<th>Hit rate(%)</th>
<th>IsAboveMean</th>
<th>False alarm rate(%)</th>
<th>IsBelowMean</th>
<th>IsSelected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS10</td>
<td>1000</td>
<td>66.4</td>
<td>T</td>
<td>16.6</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>CS11</td>
<td>1100</td>
<td>56.27272727</td>
<td>F</td>
<td>23.0909091</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>CS12</td>
<td>1200</td>
<td>55.83333333</td>
<td>F</td>
<td>1.83333333</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>CS13</td>
<td>1300</td>
<td>78.46153846</td>
<td>T</td>
<td>25.9230769</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>CS14</td>
<td>1400</td>
<td>67</td>
<td>T</td>
<td>12.2857143</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>CS15</td>
<td>1500</td>
<td>57.13333333</td>
<td>F</td>
<td>36.2666667</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>CS16</td>
<td>1600</td>
<td>67.6875</td>
<td>T</td>
<td>5.6875</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>CS17</td>
<td>1700</td>
<td>68</td>
<td>T</td>
<td>14.5882353</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>CS18</td>
<td>1800</td>
<td>61.55555556</td>
<td>F</td>
<td>62.9444444</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>CS19</td>
<td>1900</td>
<td>61.47368421</td>
<td>F</td>
<td>27.7368421</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>CS20</td>
<td>2000</td>
<td>63.6</td>
<td>T</td>
<td>31.85</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>CS30</td>
<td>3000</td>
<td>52.5</td>
<td>F</td>
<td>30.4333333</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 6.1.3.1.b

The above table illustrate that the possible optimum classifiers are CS10, CS14, CS16, CS17 that the value of the hit rate and false alarm rate upper and below the mean respectively.
3. Finding out the optimum classifier which is filtered by the mean of the selected possible optimum classifiers. After extract all the possible satisfied classifiers, it is time to calculate the new mean of this subset.

The mean of hit rate

\[
= \frac{66.4 + 67.6875 + 68}{4}
\]

= 67.27%

The mean of false alarm rate

\[
= \frac{16.6 + 12.2857 + 5.6875 + 14.5882}{4}
\]

= 12.29%

The classifier which contains the value of hit rate above this mean and the false alarm rate below this mean is CS16. Therefore, the optimum open palm classifier is CS16 which is trained by solid background color image.

The features of this classifier are:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of positive image</td>
<td>1600</td>
</tr>
<tr>
<td>No. of negative image</td>
<td>3200</td>
</tr>
<tr>
<td>Hits rate (%)</td>
<td>67.69</td>
</tr>
<tr>
<td>Error rate (%)</td>
<td>32.31</td>
</tr>
<tr>
<td>False alarm rate (%)</td>
<td>5.69</td>
</tr>
<tr>
<td>No. of Stage</td>
<td>7</td>
</tr>
<tr>
<td>Training Time (hr)</td>
<td>2</td>
</tr>
</tbody>
</table>
6.1.3.2 The evaluation of training classifiers

After find out the best classifier, it is time to estimate and evaluate how to train the classifier efficiently. We extract the possible satisfied classifiers which are mention in Table 6.1.3.i and then estimate these factors that will provide satisfied classifiers.

The following table is pointed out all features of possible satisfied classifiers.

<table>
<thead>
<tr>
<th>Classifier ID</th>
<th>No. of positive image</th>
<th>Hits rate (%)</th>
<th>Error rate (%)</th>
<th>False alarm rate (%)</th>
<th>No. of Stage</th>
<th>Time (hr)</th>
<th>No. of weak classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS17</td>
<td>1700</td>
<td>68</td>
<td>32</td>
<td>14.5882</td>
<td>6</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>CS16</td>
<td>1600</td>
<td>67.6875</td>
<td>32.3125</td>
<td>5.6875</td>
<td>7</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>CS14</td>
<td>1400</td>
<td>67</td>
<td>33</td>
<td>12.2857</td>
<td>9</td>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>CS10</td>
<td>1000</td>
<td>66.4</td>
<td>33.6</td>
<td>16.6</td>
<td>8</td>
<td>2</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 6.1.3.2.a

First, the background of the positive image should be in pure color and the range of the number of positive image from 1000 to 1700. However, it is not illustrate that if we train the classifier with this range of the positive image, it will produce the high performance classifier. It is show that the more number of the positive images in training does not represent it will generate an efficiency classifier.

Second, the more of number of the positive images during the training, the resulting number of training stage does not surely increased. However, if using random color background image for training, the training stage must be small than using solid background image for training. And we set the minimum training stage to produce the satisfied classifiers is 6.

Third, the training time of using random color background image for training must be longer than using solid background image for training. Because of the resources is put more on deal with the random color background during the training. In addition, the period of the training is also depend on your hardware performance such as if user can provide more memory to training, the time will be shorter. In this report, the hardware that we used to training is shown as Table 6.1.3.2.b and we use 4 GB memory for training.
Fourth, the more of number of the positive images during the training, the resulting number of weak classifiers do not surely increased. However, if using random color background image for training, the number of weak classifiers must be smaller than using solid background image for training. And we set the minimum number of weak classifiers to produce the satisfied classifiers is 19.

The follow table 6.1.3.2.c is the summary of the estimation and evaluation for training the satisfied classifiers.

<table>
<thead>
<tr>
<th>System Hardware</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor:</td>
<td>Intel® Core™ 2 Duo CPU E8500 @ 3.16 GHz 3.17 GHz</td>
</tr>
<tr>
<td>Installed memory (RAM)</td>
<td>4.00 GB</td>
</tr>
<tr>
<td>System type:</td>
<td>64-bit Operating System</td>
</tr>
</tbody>
</table>

Table 6.1.3.2.b The performance of the PC

<table>
<thead>
<tr>
<th>Factor</th>
<th>The evaluation of training satisfied classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of positive image</td>
<td>The background of the positive image should be in solid background color.</td>
</tr>
<tr>
<td></td>
<td>The range of the positive image is from 1000 to 1700</td>
</tr>
<tr>
<td>No. of negative image</td>
<td>2000~3400</td>
</tr>
<tr>
<td>Hits rate (%)</td>
<td>&gt;66%</td>
</tr>
<tr>
<td>Error rate(%)</td>
<td>&lt;33%</td>
</tr>
<tr>
<td>False alarm rate (%)</td>
<td>&lt;16.6%</td>
</tr>
<tr>
<td>No. of Stage</td>
<td>&gt;6</td>
</tr>
<tr>
<td>Training Time (hr)</td>
<td>about 2 hour</td>
</tr>
<tr>
<td></td>
<td>if the hardware criteria is same as or greater than table 6.1.3.2.b</td>
</tr>
<tr>
<td>No. of weak classifiers</td>
<td>&gt;19</td>
</tr>
</tbody>
</table>

Table 6.1.3.2.c The estimation and evaluation for training the satisfied classifiers
6.1.4 Improvement of the strong classifier

After find out the optimum classifier, we would use it to test another testing data which is not using to train that classifier. In addition, we would integrate different methods such as skin detection that are mentioned before to improve the classifier. One of the testing data sample is shown in figure 6.1.4.a.

![Sample of the testing data](image)

Figure 6.1.4.a sample of the testing data

6.1.4.1 Result

When the classifier hit can detect the gesture, it will shows in figure 6.1.4.a. However, if the image is hit but it is a false positive image, then it not only as a hit image, but also the false positive image that is shown in figure 6.1.4.b. The result is shown in table 6.1.4.c.

![Hit image](image)

Figure 6.1.4.a The “Hit” image

![False alarm image](image)

Figure 6.1.4.b The “False alarm” image


<table>
<thead>
<tr>
<th></th>
<th>Before the improvement</th>
<th>After the improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of testing image</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>Hits</td>
<td>176</td>
<td>176</td>
</tr>
<tr>
<td>False positive image</td>
<td>162</td>
<td>73</td>
</tr>
<tr>
<td>Missed</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>Hit rate (%)</td>
<td>70.4</td>
<td>70.4</td>
</tr>
<tr>
<td>False alarm rate (%)</td>
<td>64.8</td>
<td>29.2</td>
</tr>
<tr>
<td>Error rate (%)</td>
<td>29.6</td>
<td>29.6</td>
</tr>
</tbody>
</table>

Table 6.1.4.b the result of the testing data

6.1.4.2 Discussion and Analysis

In table 6.1.4.b, it shows that the main different before and after improvement is the false alarm rate. And the false alarm rate is big different from the result of using OpenCV’s performance utility to test. The reason is if the testing data. The area of background in this testing data is larger than previous one. Therefore, the chance of the false positive image is increased. And the false alarm rate is 64.8% before improvement.

In contrast, the false alarm rate is reduced to 29.2% after using different methods to eliminate the noise. Therefore, these methods such as contour and skin detection, they not only can reduce the noise from the image, but also they can let the classifier to more focus on the possible area where contains the hand.
6.1.5 Limitations

The variation of the data set is not enough. Because the data set only contains a few people’s open palm gesture. The different of each image is the angle, size, brightness and contrast but with the same object. It could be effect the result of the training process. Also, the training time is consuming. During the training process, the most of the memory will spend on this task, so it is difficult to do other job during the process. In addition, the result of the OpenCV test performance utility is different from my internal testing because of the testing data set is different. Therefore, it is hard to predict the performance of the classifier after training and difficult to test the performance in real time.

6.1.6 Recommendations

In order to increase the precision of the performance of the result, we can increase the variation of the detected object. i.e. the open palm gesture. The training data set should be contains so many different people’s open palm gesture to increase the variation of the training data set. In addition, it is necessary to training anther classifier for recognize different gesture to prove the testing prediction is suitable for another training object. Moreover, it is suggested that provide as more as memory for the training process to increase the training speed.
6.2 Finger Tip Detection

6.2.1 Description

After discuss with the performance open palm gesture recognition, it is time to discuss the result of the finger tip detection. In this session, we would using 250 testing image for testing and for each finger, it will be used 50 testing image to test the detector.

In addition, we would test two finger tip detectors; the first detector is using “Dot product” method to extract the finger tip points, another is using “cvConvexHull2” function. And then both detectors will apply centralization, clustering and proximity measure to eliminate the noise. And we would use the hit rate, false alarm rate to evaluate the performance. The following is the evaluation of the finger tip detection.

6.2.2 Result and findings

In table 6.2.2.a, it shows the testing result of finger tip detector which using “Dot product” method (DP) to recognize the finger tips.

<table>
<thead>
<tr>
<th>No. of Finger</th>
<th>No. of testing Image</th>
<th>Hits (DP)</th>
<th>False positive image (DP)</th>
<th>Missed (DP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>50</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>50</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>27</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>8</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>9</td>
<td>0</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 6.2.2 a testing result of finger tip detector created by algebraic operation

In table 6.2.2.b, it shows the testing result of finger tip detector which using “cvConvexHull2” function (CV) to recognize the finger tips.

<table>
<thead>
<tr>
<th>No. of Finger</th>
<th>No. of testing Image</th>
<th>Hits (CV)</th>
<th>False positive image (CV)</th>
<th>Missed (CV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>47</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>50</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>38</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>26</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 6.2.2 b testing result of finger tip detector created by OpenCV internal function
The accuracy (Hit rate) of the finger tip detector:
\[
\text{Hit rate} = \frac{\text{The number of correct detected}}{\text{Total number of testing image}}
\]

The false alarm rate of the finger tip detector:
\[
\text{False alarm rate} = \frac{\text{The number of false positive image}}{\text{Total number of testing image}}
\]

The error rate of the finger tip detector:
\[
\text{Error rate} = \frac{\text{The number of incorrect detected}}{\text{Total number of testing image}}
\]

The following tables are shown the hit rate, error rate and false alarm rate of the finger tip detector which trained by “Dot product” method (DP) and “cvConvexHull2” function (CV):

<table>
<thead>
<tr>
<th>No. of Finger</th>
<th>Hit rate (DP) (%)</th>
<th>Error rate (DP) (%)</th>
<th>False alarm rate (DP) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>84</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>82</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.2.2 c

<table>
<thead>
<tr>
<th>No. of Finger</th>
<th>Hit rate (CV) (%)</th>
<th>Error rate (CV) (%)</th>
<th>False alarm rate (CV) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>94</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>36</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.2.2 d
6.2.3 Discussion and Analysis

In this session, we would compare some factors of the detectors such as hit rate, error rate, and false alarm rate. And then find out the satisfied finger tip detector. The details are shown as following:

![The comparison of Hit rate](image)

The graph 6.2.3.a shows the comparison of accuracy (hit rate) of two finger tip detectors. On one and two finger tip detection, both detectors perform really high hit rate which are almost 100%, so that the performance is really good.

However, on the three finger tip detection, the hit rate of finger tip detector (DP) is dropped down dramatically to 54 % and the hit rate of finger tip detector (CV) also decreases to 76%.

Unfortunately, the finger tip detector’s (DP) performance is continuing decrease dramatically to around 16% to 18% on four and five finger tip detection. In contrast, the finger tip detector’s (CV)’s hit rate falling down steadily on four finger detection and return to 64%.

In summary, both of the detectors perform really well during the one and two finger tip detection, and the trend is shown us the hit rate (DP) is rapidly dropped to 18% on five finger detection but the hit rate(CV) is not. So finger tip detector’s (CV)’s hit rate is better than finger tip detector’s (DP)’s one on three to five finger tip detection.
The graph 6.2.3.b shows the comparison of the error rate of two finger tip detectors. On first two finger tip detection, the error rate is kept almost 0 in both finger tip detectors. However, the error rate (DP) is increased dramatically from 0% to 80% around on four and five finger detection.

On the other hand, the error rate (CV) also increase to 48% on four finger tip detection and drop back to 36% on five finger tip detection.

To sum up, the trend is shown us the error rate (DP) rise dramatically but the error rate (CV) is not. Therefore, finger tip detector’s (CV)’s error rate is better than finger tip detector’s (DP)’s one on three to five finger tip detection.
The graph 6.2.3.b shows the comparison of false alarm rate of two finger tip detectors.

On one finger detection, the false alarm rate (CV) is 6% which is half of the false alarm rate (DP). But false alarm rate (CV) is increase to 10% in two finger tip detection and false alarm rate (DP) is decrease to 6%.

The false alarm rate (DP) is continue increase to 12% until four finger tip detection and then drop rapidly to zero in five finger tip detection. And the false alarm rate (CV) remains low value in three, four and five finger tip detection.

In conclude, the trend show us both finger tip detectors’ false alarm rate is decreased steadily from one finger tip detection to five finger tip detection. But finger tip detector’s (DP) false alarm rate is higher than finger tip detector’s (CV). Therefore, the finger tip detector (CV) performs well in false alarm rate.
6.2.3.1  The optimum finger tip detector

In order to find the optimum finger detector, it is necessary to compare two detector's false alarm rate and hit rate. Because of the error rate is depend on the hit rate and vice versa, so it is only compare two factors are enough.

On first two finger detection, both detectors' hit rate is high and the false alarm rate is low. However, on three, four and five finger detection, the hit rate (CV) is higher than hit rate (DP) and the false alarm rate (CV) is zero on three and four finger tip detection. Now, we will calculate the mean of the hit rate and false alarm rate to determine the optimum finger tip detector.

The mean of hit rate (DP)

\[
\frac{100 + 100 + 54 + 16 + 18}{5} = \frac{288}{5} = 57.6\%
\]

The mean of false alarm rate (DP)

\[
\frac{12 + 6 + 8 + 12 + 0}{5} = \frac{48}{5} = 7.6\%
\]
The mean of hit rate (CV)

\[
= \frac{94+100+76+52+64}{5}
\]

\[= 77.2\%
\]

The mean of false alarm rate (CV)

\[
= \frac{6+10+0+4+0}{5}
\]

\[= 4\%
\]

In summary, the mean of hit rate in finger tip detector (CV) is higher than finger tip detector’s (DP) one. In addition, the false alarm rate (CV) is lower than finger tip detector’s (DP) false alarm rate. Therefore, the optimum finger detector is used cvConvexHull2 function to detect the finger tip.

6.2.4 Limitations

The environment issue such as brightness will influence the accuracy of the detection. If it is not enough brightness, it cannot extract the hand region, and then the finger tip points will not be recognized. In addition, if the background contains the human skin color issue, it would be increase the false alarm rate and influence the performance of the detection. Also, it cannot be reversed the webcam in order to provide a stable and suitable platform to the clustering. Moreover, sometime the thumb does not be recognized because if the amount of unnecessary data point is large, then it will affect the result clustering finger tip data point. At last, the testing data is limited to test the precision of the detectors.

6.2.5 Recommendations

In order to maintain the stable performance of the finger tip detector, it is important to ensure the environment has enough brightness and it cannot contain skin color base’s object during the recognition. Also, it is need to enlarge the amount and the variation of the testing data to testing the performance.
6.3  **Real Time Finger Tip Detection Application Development**

In order to apply the finger tip detection techniques discussed in the above sections, a software product of finger tip detection is developed. This software product can control the Windows mouse cursor in real time and it is called “Finger Cursor”. The components and its performance are introduced and discussed in the first part of this session.

Apart from the mouse controller, another application has been developed in future that is called “Finger Draw”. It is a drawing panel that can provide a platform to the user to drawing directly by hand. For more detail, it will discuss in the second part of this session.

6.4  **The components of the “Finger Cursor”**

The “Finger Cursor” can imitate the Windows mouse’s function that include right click, left click, mouse drag and mouse wheel movement.

6.4.1  **Mouse cursor movement**

In this application, user can use index finger to imitate the Windows mouse movement, when the finger move to left, then the cursor also will move to left and so on. Figure 6.4.a shows the finger can move entire screen whatever at the centre, left corner or upper side.

![Figure 6.4.a Using index finger to control the mouse cursor](image)

For detail, please refer to the demo clip.
6.4.2 Left Click & Mouse drag

Two finger tips can represent the left click. When user use one finger tip to move the mouse cursor, they can extract the thumb to outside to imitate the mouse left down and return to original position to imitate the mouse left up which is shown in Figure 6.4.2.a.

In addition, if users want to imitate the mouse drag action, they can hold the thumb to outside and move the entire hand to desired position to drag the object.

![Figure 6.4.2.a The whole process of left click](image)

6.4.3 Right Click

Right Click can be imitated by the three finger tips. When user wants to left click, just extract the middle finger and ring finger to outside. The whole process is shown in Figure 6.4.3.a

![Figure 6.4.3.a The whole process of right click](image)
6.4.4 Mouse wheel movement

To imitate the mouse wheel movement, user can use the index finger and middle finger to represent the wheel and move downward or move upward. The whole process is shown in figure 6.4.4.a. For instance, this function can enlarge or reduce the size of the picture or as the zoom in or zoom out function in Google Earth. (Figure 6.4.4.b)

Figure 6.4.4.a The whole process of mouse wheel movement

Figure 6.4.4.b Using the index finger and middle finger can zoom in or zoom out the Google Earth.
6.4.5 Imitated the Keyboard right and left arrow key

In order to let the user to browse the pictures in different picture viewers, it imitated the right and left arrow key. In normal case, when using the picture viewers, user can press the right and left arrow key to browse the next image or previous image. In this application, user can swing their hand quickly to imitate the right and left arrow key. When the hand swings to the right hand side quickly, it is imitate the right arrow key in keyboard, and then the viewer would be gone to the next image. (In Figure 6.4.5.a) On the other hand, when the hand swings to the left hand side quickly, the viewer would be gone to the previous image.

Figure 6.4.5.a When the hand swings to the right hand side quickly, the viewer would be gone to the next image. For detail, please watch the FYP Demo.
6.5 Discussion and analysis

6.5.1 Performance

The real time Finger Cursor application applied the technique of finger tip recognition. In addition, it check the direction of the finger movement and number of finger in each direction, so it can estimate whether user is right click or left click during control the cursor. And the precision of the mouse function is quiet satisfied. First, the movement of the mouse cursor is smooth and easy to control. Second, the left click or right click is convenient to activated and easy to imitate double click. Third, the accuracy of wheel movement is satisfied which can refer to demo clip. At last, the additional function of right and left arrow key can let the user be more convenient to browser the image.

6.5.2 Limitations

The accuracy of the “Finger Cursor” is limited. When using the index finger to control the mouse cursor, the cursor sometime will be a little bit shake due to shaking of the finger tip point. Then it will lead to influence the accuracy of the left click. In addition, the control of the mouse cursor is not as fast as compare with using the mouse to move the cursor.

6.5.3 Recommendations

It is needs to improve the stability of the finger tip point’s recognition, and then will solve the shaking problem of the mouse cursor. It is also necessary to ensure the environment of the recognition where should not contains any skin color base object which can interrupt the performance.
6.6 The components of the “Finger Cursor” in future development

The second Finger tip recognition’s application is called “Finger Draw” which can let the user to draw anything on the drawing panel by their index finger or others. One of the main core functions is developed which is the drawing function. User can use one finger such as index finger to drawing whatever they want in the drawing panel. For example, a tree is drawn on the drawing panel in figure 6.6.a and figure 6.6.b

![Figure 6.6.a. Start to draw](image1.png)

![Figure 6.6.b Finish!](image2.png)

6.7 Discussion and analysis

6.7.1 Performance

The drawing panel is applied the finger tip detection technique. The fluency of the drawing function is quiet satisfied. However, if the user’s finger moves very fast, the drawing panel only would display a dot pixel rather than a line.

6.7.2 Limitations

The drawing panel has not been developed completely. It cannot edit the image; change the line color and so on.

6.7.3 Recommendations

There is lots of enhancement of “Finger draw”, the menu of the drawing panel should be updated to more users friendly and increase the loading, editing and saving the image function. Also, it should be let the user to change the color or the type of the brushes in drawing panel.
7. Conclusion

7.1 Achievements

Hand detection and gesture recognition are very famous in computer vision architectures, but it is difficult to maintain a precision result because it is hard to reduce noise. To solve the problem, this study looked into both feature and learning-based technique so as to for solution of the gesture detection.

First, it is success to use AdaBoost algorithm to train the strong classifier. The optimum classifier can detect about 60% open palm gesture and the false alarm rate is quite low in OpenCV internal testing. However, there are existed the bias of the result, the false alarm rate dramatically increased and the accuracy is almost unchanged. To cope with this problem, it is necessary to apply some improvement methods such as contour, skin detection theory. At last, the false alarm rate of the strong classifier has been dropped to 29%.

Second, compare the performance with different mathematic algorithm for finger tip detection, the algebraic operation is used for the detection. It basically can extract the finger tip points because there are contain lots of noise and unnecessary data points. However, I found that the algebraic operation provided less precision results if number of finger tip is increased. Therefore, using the OpenCV internal function for the basic finger tip detection is still a better option.

Third, in order to eliminate the unnecessary data points, the centralization theory, and finger tip clustering has been developed. It is also has been decided to using the contour, skin detection, Gaussian smooth, proximity measure to increase accuracy and decrease the false alarm rate during the recognition. At last, the most of the finger tip point can be detected.

Fourth, to apply the finger tip recognition and show the interaction with the computer, the software of Windows mouse cursor controller which is called “Finger Cursor” has been developed. The software are applies the finger tip detection technique for controlling the mouse cursor. It also provides some feature that does not occurs in Windows mouse.
7.2 Critical Review

Final Year Project can provide student to enhance their innovative, creative and problem solving skill. I have faced lots of challenges in the Final Year Project. First, the time of training strong classifiers is so long and the result of the training is unpredictable in sometimes. In addition, the reference of this type project is limited because it has been just developed in recent years. Also, another problem is the memory leak. Because of gesture recognition is run in real time, so the memory leak problem is the main issue to affect the speed of the program. Then, it is necessary to check the code line by line to tackle the memory leak. At last, the memory leak defect has been resolved.

Apart from the challenging, I have learned a lot from this project. I learn the knowledge of the machine learning algorithm and the training process. Also I have developed the algorithm only for cluster the finger tip points and the result is satisfied. In addition, communication with supervisor is a desired part in this year because he provides me lot of idea on the project.

In short, all of them brought me lots of experience and knowledge and it is impressive.

7.3 Future development

The gesture recognition can apply in many applications. There are some development and improvement in this project.

First of all, leaning base algorithm not only can train the open palm gesture recognition, but also training another object for the detection. For instance, we can train the classifier to recognize an object and then out put the information in the screen.

Second, we can develop a Sand printing panel by using the finger tip detection technique. Because setting the environment of the Sand Printing is difficult. And using the computer imitate the Sand printing which is so convenience and adaptable because it can provide more chance to practice for the beginner.
8. Reference:


<table>
<thead>
<tr>
<th>Month</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>October</td>
<td>1. <strong>Studies of the OpenCV</strong></td>
</tr>
<tr>
<td></td>
<td>- Capture the web cam</td>
</tr>
<tr>
<td></td>
<td>- Machine Learning AdaBoost training</td>
</tr>
<tr>
<td></td>
<td>2. <strong>Documentation:</strong></td>
</tr>
<tr>
<td></td>
<td>- the Project plan</td>
</tr>
<tr>
<td></td>
<td>- Project Management Plan</td>
</tr>
<tr>
<td></td>
<td>- Project Schedule Draft</td>
</tr>
<tr>
<td></td>
<td>3. <strong>Development:</strong></td>
</tr>
<tr>
<td></td>
<td>- An image capture programme is created. It is used to capture a</td>
</tr>
<tr>
<td></td>
<td>large amount of the image fast and quickly</td>
</tr>
<tr>
<td></td>
<td>- Image segmentation programme is created. It is used to cut out</td>
</tr>
<tr>
<td></td>
<td>the object that in the image fastly.</td>
</tr>
<tr>
<td></td>
<td>- Try to using opencv's AdaBoost training program to train the</td>
</tr>
<tr>
<td></td>
<td>xml file</td>
</tr>
<tr>
<td>November</td>
<td>1. <strong>Background Studies:</strong></td>
</tr>
<tr>
<td></td>
<td>- Learning OpenCV. Cambridge</td>
</tr>
<tr>
<td></td>
<td>- Literature Review on Adaboost machine learning and Harr-</td>
</tr>
<tr>
<td></td>
<td>feature materials.</td>
</tr>
<tr>
<td></td>
<td>- Literature Review on computing vision materials</td>
</tr>
<tr>
<td></td>
<td>2. <strong>Development:</strong></td>
</tr>
<tr>
<td></td>
<td>- using opencv's AdaBoost training program to train the open</td>
</tr>
<tr>
<td></td>
<td>palm gesture classifier.</td>
</tr>
<tr>
<td></td>
<td>- Using different combination of the training data set to train the</td>
</tr>
<tr>
<td></td>
<td>strong classifier and find out the optimum classifier</td>
</tr>
<tr>
<td>January</td>
<td>1. <strong>Development:</strong></td>
</tr>
<tr>
<td></td>
<td>- create a programme for changing the background image</td>
</tr>
<tr>
<td></td>
<td>- create a programme for changing the brightness of the training</td>
</tr>
<tr>
<td></td>
<td>sample.</td>
</tr>
<tr>
<td></td>
<td>- Using different combination of the training data set to train the</td>
</tr>
<tr>
<td></td>
<td>strong classifier and find the optimum classifier</td>
</tr>
<tr>
<td></td>
<td>- reduce the background noise during the hand detection by</td>
</tr>
</tbody>
</table>
using skin detector and ROI methods.
- solve the memory leak problem
- integrate all the method to increase the hit rate and reduce the error rate.

2. **Documentation:**
- Preparing the intern report.

<table>
<thead>
<tr>
<th>February</th>
<th>1. <strong>Background Studies:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Literature Review on finger tip detection technique</td>
</tr>
</tbody>
</table>

2. **Documentation:**
- Intern report.

3. **Development:**
- Detection finger tip:
  - Gaussian entire image
  - using background subtraction to separate the hand and the background
  - using contour method to find out the hand accurately.
  - using dot product and contour hull method to identify the finger tip
  - try to eliminate the non finger tip point
  - success to detect the finger tip

<table>
<thead>
<tr>
<th>March</th>
<th>1. <strong>Development:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- After testing the detection of finger tip, the background subtraction method is eliminated and instead of using the skin detector.</td>
</tr>
<tr>
<td></td>
<td>- Using dot product method is eliminated and instead of using opencv contour hull function.</td>
</tr>
<tr>
<td></td>
<td>- Using some clustering method to reduce the noise.</td>
</tr>
<tr>
<td></td>
<td>- Using SSE to find out the optimum cluster.</td>
</tr>
<tr>
<td></td>
<td>- Implement the application for showing the finger tip detection.</td>
</tr>
<tr>
<td></td>
<td>- Finishing the application: Finger tip mouse controller</td>
</tr>
</tbody>
</table>

2. **Documentation:**
- Final report