159-793

Hand Detection and Gesture Recognition using ASL Gestures

Supervisor: Andre L. C. Barczak

Student: Dakuan CUI
Abstract

Gesture recognition application is very important in computer vision architectures. So far, it has many different applications on this area. These include human body, face, fingerprint recognition, include vehicle, aircraft recognition etc. In this report, I will describe the algorithms and techniques related to computer vision, object recognition, machine learning and also associated with ASL (American Sign Language) gesture detection and recognition, along with their operation approach in the real world.

This work is distinguished by four key contributions. The first contribution is the description of gesture detection and recognition. The second contribution covers related knowledge include PAC Model, the evaluation of detection system and ASL. The third contribution is to describe methodologies that include AdaBoost algorithms, integral image and rectangle feature (Haar-like features). The fourth contribution presents the general operation approach, including how to use OpenCV functions provided in the implementation procedure.

Key Words: Gesture Recognition, Integral Image, Haar-Like, Moment Invariant, AdaBoost Algorithms, OpenCV.
# Table of Content

ABSTRACT ..............................................................................................................2

CONTENT ........................................................................................................... 3

LIST OF FIGURES ...............................................................................................5

LIST OF TABLE ....................................................................................................6

1. INTRODUCTION ..............................................................................................7

2. LITERATURE REVIEW ......................................................................................8

2.1 Uses of Gesture Recognition ........................................................................8

2.2 Detection method categories .......................................................................8

2.3 Feature Invariant Approaches ..................................................................9

2.4 Machine Learning ......................................................................................10

2.4.1 PAC Learning model ..............................................................................11

2.4.2 Boosting and AdaBoost ........................................................................12

2.5 Moment Invariant ......................................................................................12

2.6 American Sign Language (ASL) ..................................................................13

2.7 Performance Evaluation ............................................................................16

2.7.1 Object Database ....................................................................................16

2.7.2 Face Image Database ............................................................................16

2.7.3 Hand Gesture Database .........................................................................17

2.7.4 Performance Evaluation .......................................................................17

3. METHODOLOGY ............................................................................................18

3.1 AdaBoost Algorithm ..................................................................................18

3.1.1 AdaBoost Training Algorithm ................................................................18

3.1.2 AdaBoost Algorithm flow chart ............................................................19

3.1.3 Weak Classifier .....................................................................................21

3.1.4 Strong Classifier ....................................................................................22
3.2 Cascade of Classifiers ................................................................. 23
3.3 Rectangle Features and Integral Image ........................................ 24
   3.3.1 Rectangle Features ............................................................. 24
      3.3.1.1 Abstract ................................................................. 24
      3.3.1.2 Features Templates .................................................. 24
      3.3.1.3 Features calculate property ...................................... 26
   3.3.2 Integral Image ................................................................. 28
      3.3.2.1 Concept ................................................................. 28
      3.3.2.2 Use Integral Image compute Rectangle Features ............. 29
4. IMPLEMENTATION ..................................................................... 31
   4.1 Data Preparation ................................................................. 31
   4.2 Image Segmentation ............................................................. 32
   4.3 Create samples ................................................................. 35
   4.4 Training and Use Classifier ................................................. 39
5. RESULT ................................................................................... 41
   5.1 Samples set ........................................................................ 41
   5.2 Experiment Results ............................................................ 41
6. CONCLUSIONS ....................................................................... 49
7. ACKNOWLEDGMENTS ............................................................. 49
8. REFERENCE ........................................................................... 50
   APPENDIX 1 Take positive image use Webcam ........................... 52
   APPENDIX 2 Segmentation of positive image with HSV colour model 55
   APPENDIX 3 Segmentation of positive image with contour model .... 61
   APPENDIX 4 Replace background with random background ........... 69
   APPENDIX 5 Using Classifier .................................................... 75
LIST OF FIGURES

1. Figure 2.2 The template is composed of 16 regions (the gray boxes) and 23 relations (shown by arrows)................................................................. 9
2. Figure 2.5 Seven type of moment invariant................................................................. 13
3. Figure 2.6 ASL alphabet ................................................................. 15
4. Figure 3.1.2 AdaBoost Algorithm flow chart................................................................. 20
5. Figure 3.1.3 Training 2 dimensional feature space using AdaBoost................................. 22
6. Figure 3.2 A classifier organized in cascades................................................................. 24
7. Figure 3.3.1.2.1 Three kinds of Haar-Like Features................................................................. 25
8. Figure 3.3.1.2.2 Three kinds of rectangular features and four kinds of formals................. 25
9. Figure 3.3.1.3 Calculate possible rectangle in m*m sub window.................................. 26
10. Figure 3.3.2.1.1 Integrals Image........................................................................... 28
11. Figure 3.3.2.1.2 Integrals Image expressions........................................................................ 28
12. Figure 3.3.2.1.3 Matrix sum........................................................................... 29
13. Figure 3.3.2.2 Compute Haar features........................................................................... 30
14. Figure 4.1 Black and Green background image from webcam(gesture 3 in ASL).................. 32
15. Figure 4.2.1 Positive image in the normal lighting condition........................................ 32
16. Figure 4.2.2 Hand and background image........................................................................ 33
17. Figure 4.2.3 Apply the histogram equalization image................................................. 33
18. Figure 4.2.4 RGB color space........................................................................... 34
19. Figure 4.2.5 HSV color wheel(lift) and conical representation (right).............................. 34
20. Figure 4.2.6 Segment positive image using HSV colour model.................................... 35
21. Figure 4.2.7 Contour, mask and result using found contour method.............................. 35
22. Figure 4.3.1 Gesture 3 with random background.......................................................... 36
23. Figure 4.3.2 Ten positive samples with different angles and lighting condition............... 37
24. Figure 5.1 ASL 0-5 positive image sample.................................................................... 41
25. Figure 5.2.1 ROC Curve.................................................................................. 42
26. Figure 5.2.2 Gesture 3 result with 4 different stages...................................................... 43
27. Figure 5.2.3 Gesture 4 result with 4 different stages...................................................... 43
28. Figure 5.2.4 Gesture 5 result with 4 different stages

29. Figure 5.2.5 Gesture 0 Stage 8 result

30. Figure 5.2.6 Gesture 1 Stage 9 result

31. Figure 5.2.7 Gesture 2 Stage 12 result

32. Figure 5.2.8 Gesture 3 Stage 13 result

33. Figure 5.2.9 Gesture 4 Stage 14 result

34. Figure 5.2.10 Gesture 5 Stage 15 result

LIST OF TABLE

1. Table 2.7.2 Several popular face databases

2. Table 2.7.3 Massey’s hand gesture databases

3. Table 3.3.1.3.1 Different rectangle feature corresponded with different condition (s,t)

4. Table 3.3.1.3.2 Feature sum by 24*24

5. Table 5.2.1 Gesture 3 HR and ER with different stage

6. Table 5.2.2 Gesture 4 HR and ER with different stage

7. Table 5.2.3 Gesture 5 HR and ER with different stage
1. Introduction

The early input devices for a computer included a keyboard and mouse, the computers have played role in processing information passively. In order to enable the computers to gather information actively and expand the fields of application of the computer, the subject of computer vision was generated. There are many areas of study related to computer vision; Young et al. [1] suggest the following categories:

● Image Processing Image in → Image out (improvements filters);
● Image Analysis Image in → Measurements out (size, texture, positions, etc);
● Image Understanding Image in → High-level description out (what is there, what is the relationship with the environment etc)

Gesture recognition is a typical applies of image understanding. It includes two phases: 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 a certain pattern that differentiates the object from the world.[2]. For example: object detection detects any human hand, object recognition finds which gesture it is. (ASL).

In recent years, computer vision development has had great advancements. Whether these advancements were in the algorithms or in new ways of thinking, they both have essentially changed the gesture recognition area. The research of gesture detection can traced back to 1970s. Now this research is base on Learning Classification Functions. For example, AdaBoost structure is based on Viola-Jones method, which is a method that is based on Integral Image, Cascade of Classifiers and AdaBoost Algorithms. The “Hand Detection and Gesture Recognition using ASL Gestures” project main objective is to detect and recognize an ASL hand gesture via a web camera. To fulfill the objective, a list of popular methods have been implemented. These include AdaBoost algorithm, Cascade of Classifiers, Rectangle Feature and Integral Image. In the project report, the first part is the introduction of gesture detection and recognition. The second part covers related knowledge includes PAC Model, the evaluation of detection system and ASL. The third part describes the methodology and it includes AdaBoost algorithms, Integral Image and Rectangle Feature. The fourth part presents the general operation approach, including how to use OpenCV functions, provided in the implementation procedure.
Hand Detection and Gesture Recognition using ASL Gestures

2. Literature Review

During the development of this project, some useful technical definitions have been reviewed. Those are closely connected with the hand detection and gesture recognition using ASL Gestures project. Therefore, those concepts are essential to review.

2.1 Uses of Gesture Recognition

It is our dream to communicate with computer without any boundaries, even reaching the way the communication between people happens one day. The development of Gesture recognition is the witness of the human being effort aiming to this concept.

There are several typical applications [3]:

A. Sign language recognition
B. For socially assistive robotics
C. Directional indication through pointing
D. Control through facial gestures.
E. Alternative computer interfaces.
F. Immersive game technology
G. Virtual controllers
H. Affective computing
I. Remote control.

2.2 Detection method categories

According to Ming-Hsuan Yang etc (2002), [4] the face detection methods were classified into four categories. These four categories could be used with other object detection as well. Certainly some of recognition algorithms do not apply to only a category, and possibly combination of more than one category are used so that they will be more efficient and accurate.

The following are the four categories:

A. Knowledge-based methods

This method makes a rule using the object detection knowledge, the computer detects any object in
the detection area using this rule. For example, in the face detection transfers the human face knowledge to the rule. It includes the contour knowledge, organ structure’s relationship knowledge.

B. Feature invariant approaches
When choosing the object to train, we often encounter a different angle, or a different lighting condition. In this method, an invariant characteristic can be found, even when the angle and lighting conditions change.

C. Template matching methods
First, assuming a standard object model, compute and input the relation values. The method will detect if there is any object according to the relation values. For instance, if we use face detection, assumes a standard face model, and then input the values about image and the standard model. The figure 2.2 listed below includes 16 regions, 23 relations values. Finally, we can detect the detecting object according to those values.

![Figure 2.2](image)

Figure 2.2
The template is composed of 16 regions (the gray boxes) and 23 relations (shown by arrows)[5].

D. Appearance-based methods
In contrast to template matching, the models (or templates) are learned from a set of training images, which should capture the representative variability of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection.

2.3 Feature Invariant Approaches
In 2.2 of chapter two, four kinds of different detection methods are introduced, here only focuses on
feature invariants approaches. When the environment changes, the image character will change along
with the environment, it is harder to detect the image. Especially, when posture or shooting angle
changes, it will affect image detection seriously. Therefore, the algorithms need to search for particular
characters which are the fixed structure feature especially when posture, visual angle or shooting angle
varies, and use those features to locate the human hand. This method is known as feature invariant
approaches.

To implement this approach, the academician starts to observe the image feature, which does not
change along with the environment. If the Academician found that skin color is a feature that will not
change along with the environment after large amount of experiments and research, such a feature can
then be used. Certainly different lighting condition may interfere with the skin color, reducing the
accuracy of the detection process. However, if the studied image sample is big enough, we can locate
the skin color inside a smaller color space and will be able to make the image detection simple.

The AdaBoost algorithm in this report is based on Haar-like feature, which are invariant to scaling. In
the following chapter, will continue to introduce the process using AdaBoost algorithm.

2.4 Machine Learning

Machine learning is a sub domain of artificial intelligence After Probably Approximately Correct
(PAC) Learning has proposed, it enhanced the concepts of stronger learning and weak learning which
make the training machine study more convenient.

VALLANT proposed Probably Approximately Correct (PAC) Learning in 1984 [6], it is based on
sample training, is a framework for machine learning's mathematical analysis. In this framework, the
learner receives samples and must select a generalization function (called the hypothesis) from a
certain class of possible functions. “The goal is that, with high probability (the "probably" part), the
selected function will have low generalization function (the "approximately correct" part). The learner
must be able to learn the concept given any arbitrary approximation ratio, probability of success, or
distribution of the samples.” [7]

VALLANT proposed the methodology to obtain learning from computer science's point of view, the
method includes:

1. choosing an appropriate information gathering mechanism
2. the learning protocol
3. exploring the class of concepts that can be learned in a polynomial number of steps.

The PAC model will be introduced as we shall see in the following section.

### 2.4.1 PAC learning model

**Definition 2.0.1** Let $C$ be a class of Boolean functions $f : \{0,1\} \rightarrow \{0,1\}$. We say that $C$ is (efficiently) PAC-learnable if there exists an algorithm $A$ such that:

- for every $f \in C$
- for every probability distribution $D$
- for every $0 < \varepsilon < 1/2$
- and for every $0 < \delta < 1$

algorithm $A$ on input $\varepsilon$ and $\delta$ and a source of random examples, distributed according to the distribution $D$, runs in time polynomial in $1/\varepsilon$, $1/\delta$ and in the relevant parameters of the class $C$ (in particular it gets at most polynomially-many examples), and with probability at least $1-\delta$ halts with a (polynomial-time computable) function $h$ such that $\text{error}(h,f) \leq \varepsilon$. [8]

The formal concept of PAC learning model is very abstract which made it is hard to understand.

- X is the instance space, in the project, it will include the sample prepared for training.
- C is the concept, it is a subset of $X$. One concept is the set of all the samples that prepared for the particular gesture in ASL.
- D is the probability distribution.
- EX(c,D) is a procedure, it use the probability distribution D, then gives the correct label $c(x)$.

Say that there is an algorithm $A$ that given access to EX(c,D) and inputs $\varepsilon$ and $\delta$ that, with probability of at least $1-\delta$, $A$ outputs a hypothesis $h \in C$ that has error less than or equal to $\varepsilon$ with examples drawn from $X$ with the distribution $D$. If there is such an algorithm for every concept $c \in C$, for every distribution $D$ over $X$, and for all $0 < \varepsilon < 1/2$ and $0 < \delta < 1/2$ then $C$ is PAC learnable.

We can also say that $A$ is a **PAC learning algorithm** for $C$. [9]
2.4.2 Boosting and AdaBoost

The concepts of weak learner and strong learner derived from PAC model, a weak learner is a classifier only slightly correlated with the labels. A strong learner is a well correlated with true labels.

Random guessing would be like tossing a coin to get a probability accuracy of 50%. If there is a condition that can help the estimate to slightly improve the accuracy, we say this classifier is a weak classifier. At the same time, if there is another condition that can improve the accuracy greatly, it is a strong classifier, the procedure that gets this condition is considered a strong learner. The boosting algorithm is a machine learning algorithm and it can upgrade the weak learner to the strong learner, it brings a big benefit for machine learning.

Boosting algorithm is useful for classification, creation of models, image segmentation and data mining. AdaBoost's full name is Adaptive Boosting, from the name, we will understand it is a improvement from Boosting algorithm. Freund and Schapire proposed the AdaBoost algorithm in 1996 [10]. The difference with Boosting algorithms is that boosting algorithms needs to know the error rate's lower limit. AdaBoost algorithm automatically adjusts the lower rate of error rate according to the weak learning's feedback. The AdaBoost algorithm does not need to know the details of weak learning and it will achieve the same efficiency as Boosting.

After the AdaBoost algorithm was proposed, it obtained lots of attention in machine learning field and it has been applied in many areas. No matter what kind of data is, AdaBoost algorithm is able to enhance the learning accuracy and very easily apply it in related areas in the real world.

2.5 Moment Invariant

Moment invariants are properties of connected regions in binary images that are invariant to translation, rotation and scaling. They are useful because they define a simply calculated set of region properties that used for shape classification and parts recognition.

These are two famous moments invariant:

Hu's Moment Invariant

Zernike’s Moments Invariant
I will only focus on Hu’s Moment Invariant, which was proposed in 1962. There are commonly used seven types of moment invariant in Hu’s Moment Invariant (Figure 2.5):

\[ I_1 = \eta_{20} + \eta_{02} \]
\[ I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \]
\[ I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \]
\[ I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \]
\[ I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 
   (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \]
\[ I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \]
\[ I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - 
   (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \]

Figure 2.5 seven type of moment invariant [11]

2.6 American Sign Language (ASL)

American Sign Language (ASL, also Ameslan) is the dominant sign language of the Deaf community in the United States, in the English-speaking parts of Canada, and in parts of Mexico. “It is a manual language or visual language, meaning that the information is expressed not with combinations of sounds but with combinations of handshakes, palm orientations, movements of the hands, arms and body, and facial expressions.”[12]

The Figure 2.6 list below is the American manual alphabet; it contains A-Z 26 letters and 0-9 ten numbers. For the deaf people, they can use those visual alphabet to figure spell some meaningful words in order to communicate with other people who know ASL. In most cases, the dominant hand will sign
those letters and palm will face the viewer.

The main project task is to produce a gesture recognition system, the system will be able to recognize the ASL gesture using web cameras. In the real world, it has practical usage, if the mute demonstrates the hand posture in front of the web-camera, the system will detect which letter it is in the ASL; If we know which letter it is, we will understand which word several gesture represents, it will help mute people to communicate with normal people more easily.
Figure 2.6 ASL alphabet [13]
2.7 Performance Evaluation

2.7.1 Object Database

Building the detecting object database is a good method that helps evaluate the performance. Current popular gesture recognition method needs many original images to train, at the same time, it also require a database, which includes a large amount of number of images to evaluate the accuracy. The FERET is a project sponsored by the Defense Advanced Research Products Agency (DARPA), “its primary mission was to develop automatic face recognition capabilities that could be employed to assist security, intelligence and law enforcement personnel in the performance of their duties.” [14] It involves one face database and an evaluation standard.

2.7.2 Face Image Database

That is several popular face databases (Table 2.7.2):

<table>
<thead>
<tr>
<th>Title</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FERET</td>
<td><a href="http://www.nist.gov/humanid/feret">www.nist.gov/humanid/feret</a></td>
<td>Different person, different Facial Expression</td>
</tr>
<tr>
<td>Yale</td>
<td><a href="http://cvc.yale.edu">http://cvc.yale.edu</a></td>
<td>Illumination condition</td>
</tr>
<tr>
<td>UMIST</td>
<td><a href="http://images.ee.umist.ac.uk/danny/database.html">http://images.ee.umist.ac.uk/danny/database.html</a></td>
<td>More than 500 images</td>
</tr>
<tr>
<td>AT&amp;T (Olivetti)</td>
<td><a href="http://www.uk.research.att.com">http://www.uk.research.att.com</a></td>
<td>More than 400 images</td>
</tr>
<tr>
<td>MIT CBCL</td>
<td><a href="http://cbcl.mit.edu/software-datasets/FaceData2.html">http://cbcl.mit.edu/software-datasets/FaceData2.html</a></td>
<td>2,429 frontal faces with few illumination variations and pose variations.</td>
</tr>
<tr>
<td>AR Face Database</td>
<td><a href="http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html">http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html</a></td>
<td>126 peoples over 4,000 colors images</td>
</tr>
<tr>
<td>CVL Database</td>
<td><a href="http://www.lrv.fri.uni-lj.si/facedb.html">http://www.lrv.fri.uni-lj.si/facedb.html</a></td>
<td>114 persons ,7 images for each person</td>
</tr>
<tr>
<td>Labeled Faces</td>
<td><a href="http://vis-www.cs.umass.edu/lfw/">http://vis-www.cs.umass.edu/lfw/</a></td>
<td>more than 13,000 images of faces</td>
</tr>
</tbody>
</table>

Table 2.7.2 several popular face databases
2.7.3 Hand Gesture Database

Here will introduce the Massey’s hand gesture database, *the Massey Hand Gesture Database is an image database containing a number of hand gesture and hand posture image.*[15] (Table 2.7.3)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Lighting Condition</th>
<th>Background</th>
<th>Size</th>
<th>Number of Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand gesture</td>
<td>Normal</td>
<td>Dark background</td>
<td>640*480</td>
</tr>
<tr>
<td>2</td>
<td>Hand gesture</td>
<td>Normal</td>
<td>RGB(0,0,0)</td>
<td>Varying/Clipped</td>
</tr>
<tr>
<td>3</td>
<td>Hand palm</td>
<td>Normal</td>
<td>Dark background</td>
<td>640*480</td>
</tr>
<tr>
<td>4</td>
<td>Hand palm</td>
<td>Normal</td>
<td>RGB(0,0,0)</td>
<td>Varying/Clipped</td>
</tr>
<tr>
<td>5</td>
<td>Hand palm</td>
<td>Artificial light/Dark room</td>
<td>Dark background</td>
<td>640*480</td>
</tr>
<tr>
<td>6</td>
<td>Hand palm</td>
<td>Artificial light/Dark room</td>
<td>Dark background</td>
<td>Varying/Clipped</td>
</tr>
</tbody>
</table>

Table 2.7.3 Massey’s hand gesture database [15]

2.7.4 Performance Evaluation

To evaluate a system is good or not, normally there are three standards to evaluate, time, accuracy and error rate. If the evaluated time is too long, it will lose the actual meaning to apply it. Testing the accuracy means testing the ratio between object number and actual number. The error rate is the ratio between error and object number. At present, only human brain is able to reach 100% accuracy and 0% error rate.
3. Methodology

Object detection methods were classified into four categories. Hand Detection and Gesture Recognition using ASL Gestures project uses a strong classifier which obtains from training lots of samples to detect object. The Haar-like features applies when training the samples, to improve the speed of computing the Haar-like feature, integral image method has been used.

3.1 AdaBoost Algorithm

The core algorithm is the AdaBoost algorithm, “AdaBoost algorithm, short for Adaptive Boosting, is a machine-learning algorithm, formulated by Yoav Freund and Robert Schapire” [16]. It was initially created for detecting face, and has had a meaningful milestone in the field of face detection, it is from a set of weak classifiers and linearly combines them, the result is a strong classifier built by boosting the weak classifiers.

3.1.1 AdaBoost Training Algorithm

The algorithm described as follows (Based on Viola and Jones, 2001):

Algorithm:

1. Given a set of training samples (x1,y1),(x2,y2)…..(xn,yn) where yi = 0 for negative sample, yi = 1 for positive sample. N is number of total training example.
2. Initialize weights W1,i = D(i), for negative D(i) = 1/(2m), where m is number of negative samples. For positive D(i) = 1/(2l), where l is number of positive samples. m + l = N.
3. For t = 1….T:
   A: Normalize the weights:
   \[
   q_{t,i} = \frac{W_{t,i}}{\sum_{j=1}^{n} W_{t,j}}
   \]
   B: For each feature, f, train a classifier h(x, f, p, θ) The error is evaluated with respect to qt :
   \[
   \varepsilon_f = \sum_{i} q_{t} |h(x_i, f, p, \theta) - y_i|\]
C: Choose the classifier, $h_t$, with the lowest error $\varepsilon$:

$$\varepsilon_t = \min_{f, p, \theta} \sum_i q_i |h(x_i, f, p, \theta) - y_i| = \sum_i q_i |h(x_i, f_t, p_t, \theta_t) - y_i|$$

$$h_t(x) = h(x, f_t, p_t, \theta_t)$$

D: Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\varepsilon_i}$$

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Where $\varepsilon_i = 0$ if sample $X_i$ is classified correctly, $\varepsilon_i = 1$ otherwise.

4 The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Where:

$$\alpha_t = \log \frac{1}{\beta_t}$$

3.1.2 AdaBoost Algorithm flow chart

As showing in Figure 3.1.2 flow chart:
BEGIN

Input a set of training samples

Initialize weights:
For i = 1 to N
  If positive then D(i) = 1/(2l)
  If negative then D(i) = 1/(2m)
END For

For t = 1, 2 .. T

Normalize the weights
For i = 1 to N
  \[ q_{t,i} = \frac{W_{t,i}}{\sum_{j=1}^{n} W_{t,j}} \]
End For

For each feature, f, train a classifier h(x, f, p, θ)

\[ \varepsilon_f = \sum_{i=1}^{n} q_i \left| h(x_i, f, p, \theta) - y_i \right| \]

Choose the classifier, h_t, with the lowest error \( \varepsilon \)

\[ \varepsilon = \min \left\{ \sum_{i=1}^{n} q_i [h(x_i, f, p, \theta) - y_i] \right\} \]

h_t = h(x, f, p, θ)

Update the weights:
\[ \beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t} \]
For i = 1 to N
  If Xi is classified correctly Then ei = 0
  If Xi isn’t classified correctly Then ei = 1
End For

END

Figure 3.1.2 AdaBoost algorithm flow chart (Based on Viola and Jones, 2001)
3.1.3 Weak Classifier

In AdaBoost Algorithm flow chart, we can see for each feature need to train one classifier. This classifier is the weak classifier \( h(x, f, p, \theta) \). There into, the most important parameter is feature \( f(x) \).

In this report, the feature value is a Haar-like feature when recognize the hand gesture. Because when doing the training, a set of chosen training sample size equals detection sub window's size, the detection sub window's size determines the quantity of Haar-like features. Image samples were collected for every gesture in the ASL table.

Training a weak classifier is to find the optimal threshold level \( \theta \) under the current weight, using this value. Weak classifiers are trained for every possible feature, and for a single round the chosen weak classifier will have the lowest error among them.

For each feature \( f \), calculate all of the training sample's Haar-like features, and sort this list. Searching the sorted list, we can find the best threshold value, thereby get a trained weak classifier. Concretely speaking, computer those four values for each element of the sorted list.

1) the sum of entire hand sample weight \( T^+ \)
2) the sum of non-hand sample weight \( T^- \)
3) the sum of hand sample weight before current element \( S^+ \)
4) the sum of non-hand sample weight before current element \( S^- \)

when select threshold level from current element's character \( F_k \) and its before character \( F_{k-1} \), the obtained weak classifier will separate the sample in the current element position, the corresponding weak classifier divides all the elements before current element into hands or non-hand, the elements after current element into non-hands or hands.

Then, search the sorted list to find the optimal threshold level to make the smallest classifier error, in other word, it means to select the optimal weak classifier.

Figure 3.1.3 shows an example for two dimensions. The color of the pixels represent the features. For
each weak classifier, a simple threshold has been implemented. Each weak classifiers are better than 50%, however, they are not accurate enough. “By combining them linearly it is possible to define regions of the space that contain either positive or negative examples of the object. Eventually the training can achieve 0% error rate. In practice, it is difficult to achieve such low training errors, specially if the sample sets are large.”[17]

Figure 3.1.3 Training 2 dimensional feature space using AdaBoost.[17]

3.1.4 Strong Classifier

According to the description above, after T times loops, we will get a strong classifier which is made from several weak classifiers.
Where:

\[ h(x) = \begin{cases} 
1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\
0 & \text{otherwise}
\end{cases} \]

\[ \alpha_t = \log \frac{1}{\beta_t} \]

To detect the image using the strong classifier is similar with to ask all of the weak classifier to vote, and calculate the sum of the weak classifier error rate's weight, then find the final result from comparing this sum with the average voting result.

### 3.2 Cascade of Classifiers

A strong classifier is made of lots of weak classifier, we can generate from the AdaBoost. To detect the object in real-time, another concept is introduced, Cascades of classifiers. Cascades of classifiers are the improvement of AdaBoost algorithm. Viola and Jones proposed the Cascades of classifiers concept; they point out they can improve the performance without any loss in accuracy if they split this monolithic classifier in several pieces (figure 3.2).

Cascades of classifier’s basic concept are to set up the cascades layer or stage according to the user’s requirement. Each layer may eliminate certain proportions false of positive and true of positive of sub windows. For instance, if we set up 10 layers, each layer can eliminate 50% of false positives and 0.02 true of positive, thus, in theory, we can reach the accuracy:

- **False alarm** = \(0.5^{10} = 0.0009765625 = 0.098\%\)
- **Hit rate** = \(0.998^{10} = 0.98 = 98\%\) [2]

It takes lots of time to compute, when use cascades of classifier method. This method is computationally more expensive; however, the classifier achieved via this method will reach the real-time detection.
3.3. Rectangle Features and Integral Image

Rectangle features and integral image is two important factors for improving the speed of AdaBoost algorithm. “Features are characterized by performance, accuracy and discrimination powers.” [17] So all detection and recognition are based on feature of extraction. In computer vision, we can use colors, edges, FFTs, etc as features of image. How to choose features and compute features is a very important aspect, as it can influence the AdaBoost Algorithm’s detection speed. The that features have been chosen are rectangle features (Haar-like features), and method of computing is Integral Image.

3.3.1 Rectangle Features

3.3.1.1 Abstract

Rectangle feature is a feature vector using rectangles. Based on those character's detection we can encode the state of the particular area, and it is faster than the detection base on appearance.

Haar-like Rectangle features are sensitive about simple graph structure, like edge, line, but it only depicts the particular direction, like horizontal, vertical and diagonal.

3.3.1.2 Feature Templates

Feature templates are made of two or more congruent rectangle which color is white or black, define
the feature value is the pixel of white color area minus pixel of black area. This value is known as a Haar-like feature.

Haar-Like Features used to express human face first time feature by Papageorgiou. Haar-Like has three kinds of features: a. Edge feature; b. Line features; c. Center-surround features. (Figure 3.3.1.2.1). These features are based on the idea of Haar functions used in wavelets. They found that these standards Haar wavelets functions are restrict to some applications. Therefore, in order to get much better space differentiability, they implement three kinds of rectangular features. (Figure 3.3.1.2.2).

For A, B, D in Figure 3.3.1.2.2

Haar features expressions: \( V = \text{Sum (white)} - \text{Sum (Black)} \)

For C in Figure 3.3.1.2.2

Haar features expressions: \( V = \text{Sum (white)} - 2\times \text{Sum (Black)} \)

The reason for image c need to minus 2 sum (Black) is to keep the accordance of the sum of pixels of...
rectangular areas.

3.3.1.3 Features calculate property

Feature template may randomly place in the sub window with random size, each shape is a character, find all the features is the foundation to perform weak classifier training.

Assume detection rectangle size is m*m, if A and B are confirmed, then can confirm a rectangle. If those two conditions below are satisfied (Figure 3.3.1.3):

1. X direction's edge can be divide by nature number s.
2. Y direction's edge can be divide by nature number t.

then, the minimum size of the rectangle is s x t (t x s), maximum size is (m/s) * s x (m/t) * t

To find a satisfied rectangle, the following step is performed:

1. make sure A(x1,y1): x1 ∈ {1, 2, ……, m-s, m-s+1}, y1 ∈ {1, 2, ……, m-t, m-t+1}
2. after A is confirmed, B can only be inside the shadow:

X = {x1+(s-1), x1+2(s-1), ……, x1+(p-1)(s-1), x1+p(s-1)}
y2 ∈ Y = {y1+(t-1), y1+2(t-1), ……, y1+(q-1)(t-1), x1+q(t-1)}

which

p = \left\lfloor \frac{m - x_1 + 1}{s} \right\rfloor, q = \left\lfloor \frac{m - y_1 + 1}{t} \right\rfloor

In m*m's detection windows, satisfied (s,t) condition's rectangle number is :[19]
Hand Detection and Gesture Recognition using ASL Gestures

$$\Omega^m_{(s,t)} = \sum_{x=1}^{m-s+1} \sum_{y=1}^{m-t+1} p \cdot q$$

$$= \sum_{x=1}^{m-s+1} \sum_{y=1}^{m-t+1} \left[ \frac{m-x+1}{s} \right] \left[ \frac{m-y+1}{t} \right]$$

$$= \sum_{x=1}^{m-s+1} \left[ \frac{m-x+1}{s} \right] \cdot \sum_{y=1}^{m-t+1} \left[ \frac{m-y+1}{t} \right]$$

$$= \left( \left[ \frac{m}{s} \right] + \left[ \frac{m-1}{s} \right] + \cdots + \left[ \frac{s+1}{s} \right] + 1 \right) \cdot \left( \left[ \frac{m}{t} \right] + \left[ \frac{m-1}{t} \right] + \cdots + \left[ \frac{t+1}{t} \right] + 1 \right)$$

different rectangle feature corresponded with different condition (s,t). (Table 3.3.1.3.1)

<table>
<thead>
<tr>
<th></th>
<th>1,</th>
<th>2,</th>
<th>3,</th>
<th>4,</th>
<th>5,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1, 2)</td>
<td>(2, 1)</td>
<td>(1, 3)</td>
<td>(3, 1)</td>
<td>(2, 2)</td>
</tr>
</tbody>
</table>

Table 3.3.1.3.1 different rectangle feature corresponded with different condition (s,t) [19]

In the m*m's sub window, the total number of feature template in 5 is $\Omega^m_m$, which is the sum of satisfied 5 (s,t) condition's rectangle feature.

$$\Omega^m = \Omega^m_{(1,2)} + \Omega^m_{(2,1)} + \Omega^m_{(1,3)} + \Omega^m_{(3,1)} + \Omega^m_{(2,2)}$$

using 24*24 as an example, the feature sum is

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 \times \Omega^m_{(1,2)}$</td>
<td>$2 \times \left( \left[ \frac{24}{1} \right] + \left[ \frac{23}{1} \right] + \cdots + \left[ \frac{2}{1} \right] + 1 \right)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$= 2 \times (24 + 23 + \cdots + 2 + 1)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$= 2 \times 300 \times 144 = 2 \times 43,200$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$= 86,400$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| $2 \times \Omega^m_{(1,3)}$ | $2 \times \left( \left[ \frac{24}{3} \right] + \left[ \frac{23}{3} \right] + \cdots + \left[ \frac{3}{3} \right] + 1 \right)$ |
|   | $= 2 \times (24 + 23 + \cdots + 3 + 1)$ |
|   | $= 2 \times 300 \times 92 = 2 \times 27,600$ |
|   | $= 55,200$ |

| $\Omega^m_{(2,2)}$ | $\left( \left[ \frac{24}{2} \right] + \left[ \frac{23}{2} \right] + \cdots + \left[ \frac{3}{2} \right] + 1 \right)$ |
|   | $= (12 + 11 + \cdots + 2 + 1 + 1)$ |
|   | $= 144 \times 144$ |
|   | $= 20,736$ |

Table 3.3.1.3.2 feature sum by 24*24 [19]

$$\Omega^{24} = 86,400 + 55,200 + 20,736 = 162,336$$
the sum of different size sub window's rectangle feature.[19]

<table>
<thead>
<tr>
<th>36×36</th>
<th>30×30</th>
<th>24×24</th>
<th>20×20</th>
<th>16×16</th>
</tr>
</thead>
<tbody>
<tr>
<td>816,264</td>
<td>394,725</td>
<td>162,336</td>
<td>78,460</td>
<td>32,384</td>
</tr>
</tbody>
</table>

### 3.3.2 Integral Image

#### 3.3.2.1 Concept

Because the number of training sample image usually is very large. In order to compute the features quickly, Viola and Jones used a data structure called integral image.

A simple calculus formula: to compute
\[ \int_{a}^{b} f(x) \, dx \], need computer \( F(x) = \int f(x) \, dx \) first

\[ \int_{a}^{b} f(x) \, dx = F(b) - F(a) \]

![Figure 3.3.2.1.1 Integrals Image](image)

Integrals Image expressions: (Figure 3.3.2.1.2)

\[
\ii(x, y) = \sum_{x' < x} \sum_{y' < y} I(x', y')
\]

![Figure 3.3.2.1.2 Integrals Image expressions](image)

Where: \( I(x', y') \) is value of pixel at \( (x', y') \)

It is easy to extend this formula to the one below:
According to the formula above, there are two kinds of computing:

1. Compute Integrals of pixels in any rectangular area

As showing in figure 3.3.2.1.3 matrix sum

![Matrix Sum Diagram](image)

Figure 3.3.2.1.3 matrix sum [20]

Point A:

\[ \text{ii1} = \text{Sum} (A) \]

Point B:

\[ \text{ii2} = \text{Sum} (A) + \text{Sum} (B) \]

Point C:

\[ \text{ii3} = \text{Sum} (A) + \text{Sum} (C) \]

Point D:

\[ \text{ii4} = \text{Sum} (A) + \text{Sum} (B) + \text{Sum} (C) + \text{Sum} (D) \]

Rectangular area D Integral:

\[ \text{Sum} (D) = \text{ii1} + \text{ii4} - \text{ii2} - \text{ii3} \]

3.3.2.2 Use Integral Image compute Rectangle Features

The Haar features of this rectangle are the value between two different sums of the pixel in two different rectangular areas. As showing in Figure 3.3.2.2 the Haar features in ii(46) is sum of pixel in rectangle A minus sum of pixel in rectangle B.
Figure 3.3.2.2 Compute Haar features [20]

\[ \text{Sum}(r) = \text{Sum}(A) - \text{Sum}(B) \]

\[ \text{Sum}(A) = ii_4 + ii_1 - (ii_2 + ii_3) \]

\[ \text{Sum}(B) = ii_6 + ii_3 - (ii_4 + ii_5) \]

\[ \text{Sum}(r) = ii(x+w,y+h) + ii(x-1,y-1) - ii(x+w,y-1) - ii(x-1,y+h) \]

Thus, calculate the rectangle feature is only relate with integral image, and not relate to image coordinate at all. So, no matter what the measure the rectangle feature is, the time spend on computing feature is constant, and always is simple plus or minus operation. Because of those, integral image enhances the detection speed rapidly.
4. IMPLEMENTATION

Based on the idea mentioned above, and combine with the OpenCV provided functions, gesture recognition’s basic operation describes as follows:

- Data Preparation
- Image Segmentation
- Crate sample
- Training sample
- Use strong cascade

In different stage, the requirement is different. In the 'create sample' and 'training sample' stages, we will apply the functions OpenCV provided to achieve the goal. In the early data preparation stage, we need to implement the auxiliary program to detect. In this project, C++ is the programming language.

4.1 Data Preparation

In the data preparation, we need prepare a large amount of original images used for training. Vast original image is the pre-requisite to improve the detection accuracy, need prepare the positive and negative image. Positive image is the detecting object image, for example, the face detection will need many face images as positive image, and the hand detection will need lots of hand gesture as positive image. In this Hand Detection and Gesture Recognition using ASL Gestures project, the data we are talking about is the ASL gesture image. We can take positive images via a camera or acquire correlative images from the internet; the image database mentioned in this report is a good image resource. Negative image represent the background image, also can be acquired from the internet or acquire some with a camera.

In this project, the positive images were acquired using a camera (a typical low-cost webcam), and hopefully those positive image will expand the hand gesture database in the Massey university.

Taking positive hand image can via digital camera or Webcam, and to facilitate the shooting task, we can write a program using OpenCV and let the computer do it via the Webcam. The code is in appendix 1. In order to segment the original images in the segmentation stage, the image used a simple pure
color background.

We experimented using two colors, green and black, as the shooting background. The reason for using two different colors for the background was to find a better segmentation result. In the report, we will compare the result using those two colors in the segmentation stage.

![Figure 4.1 Black and Green background image from webcam(gesture 3 in ASL )](image1)

**4.2 Image Segmentation**

When obtaining the positive image, sometimes we will fall across some issues, object and background are not separate rigidly. There is also some noise impact. We need segmented images with the intention to extrude the object and apply a threshold operation, and employ some histogram equalization operation.

This hand image (Figure 4.2.1) is the positive image took in the normal lighting condition. It is easy to see the hand and background is not separated rigidly due to the lighting condition.

![Figure 4.2.1](image2)

This image (Figure 4.2.2) is after the segmentation, so hand and background is restrict partitioned.
Also can apply the histogram equalization (Figure 4.2.3):

The purpose of segmentation is to get a hand gesture image using solid black color as background. There are several means to segment the images, no matter which methods used, the basic rule is to find a threshold value will be suitable for all kinds of hand gesture image. The author used a segmentation method based on color model and another method based on contours. The two methods are described as follows:

1. Method based on colour model

In the colour model based method, we can use two different colour models, RGB or HSV.

RGB:

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. [21]

The RGB space is represented in Figure4.2.4.

In the RGB space, each color has 256 states.
HSV:
The HSV space is composed by hue, saturation and value. HSV values are within the following ranges:

- H - between 0 and 360
- S - between $\geq 0$ and 1 (if S is 0 then H is undefined)
- V - between 0 and 1

Because the difference between RGB and HSV, using HSV model to find the threshold value is easier and faster. The code using HSV to segment is in appendix2.

Using HSV colour model to segment positive image result as show: (Figure 4.2.6)
2. Method base on contour

The basic concept is to find the contour of original hand image, and set the pixel inside the contour to white colour, then save this result image as a mask image. The next step is create a new image called result image which has the same size as the original image but the colour is all black, then compare-copy between original image and the mask image to the result image. The final result image is a hand gesture with black background image. The code is in appendix 3.

Using method of found contour to segment positive image result as show: (Figure 4.2.7)

Based on my experiments, segmentation based on HSV is faster and more accuracy than based on contour, the threshold value in HSV can be used for all sample image’s segmentation.

4.3 Create samples

After segmentation image stage is finished, need use OpenCV functions to create samples to prepare for the training. Before create the samples, there are some prepared task needs to do.

First, cut the image according the hand size in the image. After cut, replace the black background with random background.
The algorithm for preparation:

First, according to the size of the hand in the image, get a same size image as the hand called imag1.

Second, choose a random background image, produce a image has the same size as image1 called image2.

Third, find the black pixels in image 1, if found, replace those respondent pixels in image 2.

After the preparation is complete, we will get a new image with the random background and the hand in back colour. The code is in appendix 4.

When finished this step, will get positive image as show (Figure 4.3.1),Now it is the time to create sample.

![Figure 4.3.1 Gesture 3 with random background](image)

There are some functions OpenCV provided for gesture detection, the relative command is introduced as follows:

Create samples :

Usage: ./createsamples

[-info <description_file_name>]
[-img <image_file_name>]
[-vec <vec_file_name>]
[-bg <background_file_name>]
[-num <number_of_samples = 1000>]
[-bgcolor <background_color = 0>]
[-inv] [-randinv] [-bgthresh <background_color_threshold = 80>]
[-maxidev <max_intensity_deviation = 40>]
[-maxxangle <max_x_rotation_angle = 1.100000>]
[-maxyangle <max_y_rotation_angle = 1.100000>]
[-maxzangle <max_z_rotation_angle = 0.500000>]
[-show [<scale = 4.000000>]]
[-w <sample_width = 24>]
[-h <sample_height = 24>]

Using this command we can create positive image for each image according to different angle, vary lighting environment.

Those ten images in figure 4.3.2 is positive image for the same gesture but in ten different angles, ten vary lighting condition.

![Figure 4.3.2 ten positive samples with different angles and lighting condition](image)

In the real world, always need to create samples use lots of images at the same time. Batch processing is a good method to improve the speed, to batch process those samples, need create a description file for those images. For different operating system, the create method is different, of course, can write a program to create as well. The code is in Appendix 5.

The description file is despite as follows:

**Under windows:**

```
#cd <your working directory>
#dir /b > description file
```
Under Linux:

$ cd <your working directory>

$ find <dir> -name '*.<ext>' -exec identify -format '%i 1 0 0 %w %h\n' \; > <description file>

[filename] [# of objects] [[x y width height] [2nd object] ...]
[filename] [# of objects] [[x y width height] [2nd object] ...]
[filename] [# of objects] [[x y width height] [2nd object] ...]
...

For example:

img/img1.jpg 1 140 100 45 45
img/img2.jpg 2 100 200 50 50 50 30 25 25
img/img3.jpg 1 0 0 20 20

In the first line, filename is img1.jpg, create a sample in file img1.jpg, the beginning coordinator is 140 100, width is 45 and height is 45.

In the second line, filename is img2.jpg, create a sample in file img2.jpg, the first coordinator is 140 100, width is 50 and height is 50, second coordinator is 50 30, width is 25 and height is 25.

We can name this kind of description file samples.dat. After created the description file, need pay special attention to the size of produced sample, normally, the sample size is the corresponding reduced size according to the actual image size. However, due to the various size of each original image, we can live up to the same gesture has the same sample size. Taking all things into consideration, common sample size is about 20 * 20.

The next step is to use the description file and the sample size to batch process the create samples. The example listed as follows:

createsamples -info samples.dat -vec samples.vec -w 20 -h 20 --show --num 1200

that is the example about how to use description file to batch process to create samples:
createsamples -info samples.dat -vec samples.vec -w 20 -h 20 –show –num 1200
The sentence above is to use samples.dat as the positive sample description file, create a file to store the sample file named samples.vec, the sample size is 20 * 20. The total number of created sample is 1200, it displays the image sample when creating them.

To complete the ASL gesture recognition, we need to create a sample .vec file, which stores all the samples for each ASL gesture, thus, we are able to produce each classifier for each ASL gesture. In this project, 1200 samples were created for each ASL gesture.

4.4 Training and Use classifiers
The samples are already created in the create samples stage and it provides the foundation for training the sample, now, it is the time to perform training sample. To create training sample, the OpenCV provided function would be implied.

We can use other command to train our own classifier.
Usage: ./haartraining
-data <dir_name>
-vec <vec_file_name>
-bg <background_file_name>
[-npos <number_of_positive_samples = 2000>]
[-nneg <number_of_negative_samples = 2000>]
[-nstages <number_of_stages = 14>]
[-nsplits <number_of_splits = 1>]
[-mem <memory_in_MB = 200>]
[-sym (default)] [-nonsym]
[-minhitrate <min_hit_rate = 0.995000>]
[-maxfalsealarm <max_false_alarm_rate = 0.500000>]
[-weighttrimming <weight_trimming = 0.950000>]
[-eqw]
Hand Detection and Gesture Recognition using ASL Gestures

[-mode <BASIC (default) | CORE | ALL>]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
[-bt <DAB | RAB | LB | GAB (default)>]
[-err <misclass (default) | gini | entropy>]
[-maxtreesplits <max_number_of_splits_in_tree_cascade = 0>]
[-minpos <min_number_of_positive_samples_per_cluster = 500>]

After complete Haartraining command, we will get our own classifier. When use the Haartraining command, we need set up many negative images, more negative images applied, the result will be better. The method used the large amount of negative image is same as batch processing positive image, which means need to create a descriptive image. The way to create a descriptive image for negative image is similar to create the descriptive image for positive image. In this project, 6400 negative images have been used. By the way, when training the sample, also need to set up the size of the training sample, the size is same as the size when created sample.

The command for training the classifier:
opencv-haartraining -data gesture0_class -vec vecg0.vec -bg ./Negatives/negative.dat -npos 1200 -nneg 6400 -nstances 30 -mode BASIC -w 20 -h 25 -minhitrate 0.9995

This command is to train gesture 0. gesture0_class represents the training classifier. Vecg0.vec is a file which stores gesture 0 samples, negative.dat represents the negative image’s description file, npos represents the number of positive image, in this example, it is 1200. nneg represents the number of negative image, in this example, it is 6400. mode is BASIC, size is 20 * 25. min_hit_rate is 0.9995. stages is 30.

The Haartraining generates a .xml file when the process is completely finished. This classifier is cascaded stronger classifier. The training stage takes days or weeks to complete. For use the classifier, the details is in Appendix5.
5. Result

5.1 Samples sets
The final AdaBoost classifier is good enough or not, it depends on the selection of weak classifier. How
to choose weak classifier depends on the selection of sample. In this project, the positive image uses
ASL hand gesture, for every gesture, there are 40 images. Each image will produce 30 different
images use random background. Thus, there are 1200 samples for each gesture. The negative image is
the random background image without any object, the total number is 6400.

Positive image: 0-5 as example as in Figure 5.1

5.2 Experiment Results
The recognition system not only just depends on the sample set mention in Figure5.1, but also the stage
number. In the training stage, stage number is an important factor as well as sample set, however, it
does not mean the stage number bigger, the result is better. Two important factors to evaluate the recognition system is the error rate and hit rate. The relationship between stage number, error rate and hit rate is mentioned in Figure 5.2.1 ROC Curve [19] which derives from Viola and Jones.

From Figure 5.2.1, this curve is to describe the face detection, we can see that the hit rate and error rate is increased along with the increase of stage number. However, when the stage number is reached at a certain number, before this number, hit rate raise rapidly, error rate rise slowly. After this number, it is opposite, hit rate raise slowly, error rate rise rapidly. Thus, it provides us a reasonable stage number via experiment.

Next I will discuss the relationship between hit rate, error rate and stage number using the classifier obtained from actual training. Use the classifier for gesture 3, gesture 4, gesture 5, gain 4 various stages (5, 8, 11, 14) with corresponding classifier, the actual experiment result is showing as Figure 5.2.2 Figure 5.2.3 and Figure 5.2.4:
Figure 5.2.2 Gesture 3 result with 4 different stages

Figure 5.2.3 Gesture 4 result with 4 different stages
Figure 5.2.4 Gesture 5 result with 4 different stages

From this, we can get various hit rate and error rate for gesture 3, gesture 4, gesture 5 under different stage showing in Table 5.2.1, Table 5.2.2 and Table 5.2.3:

<table>
<thead>
<tr>
<th>Gesture 3</th>
<th>Stage 5</th>
<th>Stage 8</th>
<th>Stage 11</th>
<th>Stage 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/T</td>
<td>43/50</td>
<td>43/50</td>
<td>36/50</td>
<td>30/50</td>
</tr>
<tr>
<td>E/T</td>
<td>43/86</td>
<td>18/61</td>
<td>1/37</td>
<td>0/30</td>
</tr>
<tr>
<td>HR</td>
<td>0.86</td>
<td>0.86</td>
<td>0.72</td>
<td>0.60</td>
</tr>
<tr>
<td>ER</td>
<td>0.50</td>
<td>0.30</td>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2.1 Gesture 3 HR and ER with different stage

<table>
<thead>
<tr>
<th>Gesture 4</th>
<th>Stage 5</th>
<th>Stage 8</th>
<th>Stage 11</th>
<th>Stage 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/T</td>
<td>50/50</td>
<td>49/50</td>
<td>40/50</td>
<td>39/50</td>
</tr>
<tr>
<td>E/T</td>
<td>126/176</td>
<td>39/88</td>
<td>8/48</td>
<td>4/43</td>
</tr>
<tr>
<td>HR</td>
<td>1.00</td>
<td>0.98</td>
<td>0.80</td>
<td>0.78</td>
</tr>
<tr>
<td>ER</td>
<td>0.72</td>
<td>0.44</td>
<td>0.17</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 5.2.2 Gesture 4 HR and ER with different stage
Hand Detection and Gesture Recognition using ASL Gestures

<table>
<thead>
<tr>
<th>Gesture 5</th>
<th>Stage5</th>
<th>Stage8</th>
<th>Stage11</th>
<th>Stage14</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/T</td>
<td>49/50</td>
<td>35/50</td>
<td>31/50</td>
<td>11/50</td>
</tr>
<tr>
<td>E/T</td>
<td>126/175</td>
<td>46/81</td>
<td>9/40</td>
<td>1/12</td>
</tr>
<tr>
<td>HR</td>
<td>0.98</td>
<td>0.70</td>
<td>0.62</td>
<td>0.22</td>
</tr>
<tr>
<td>ER</td>
<td>0.72</td>
<td>0.57</td>
<td>0.23</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Table 5.2.3 Gesture5 HR and ER with different stage

The experiment results list below is different gesture classifier under various stage number:

Figure 5.2.5 Gesture 0 Stage 8 result
Figure 5.2.6 Gesture 1 Stage 9 result

Figure 5.2.7 Gesture 2 Stage 12 result
Hand Detection and Gesture Recognition using ASL Gestures

Figure 5.2.8 Gesture 3 Stage 13 result

Figure 5.2.9 Gesture 4 Stage 14 result
In the recognition system, the recognition is fine or not depends on the quality of classifier. Based on the introduction of this report, under the fixed sample numbers, the training stage number is a crucial factor, unfortunately, the stage number is bigger, the training time is longer.

The result above uses the classifier obtained from mixtures of 6 different gestures under various stage numbers. Due to the stage number is not big enough, it leads to test result arise some bias. If we use classifier based on bigger stage number to recognize the image, the result will improve greatly.

Use current classifier is able to recognize the image via webcam, however, the recognition time is long and error rate is a little bit higher. In like manner, the real time recognition result is improved if we use classifier produced from big stage number.
6. Conclusions

Gesture recognition is a very challenging and interesting task in computer vision, and the technology about gesture recognition is getting mature in the real world application.

In the hand detection and gesture recognition using ASL Gestures project, the user use the famous AdaBoost algorithms which is proposed by Viola-Jones and create a Haar-like feature classifier. The final cascade of classifiers are robust to rotation and lighting variations and is able to dealing with all ASL standard gestures. During the project development, some of the key point has been experienced.

If the weak classifier is big enough, the strong classifier produced will have extreme low error rate.

Increase the amount of positive sample size will improve the detection accuracy.

Using green background color to segment the image is easy to find the threshold value compared with other color background.

The sample size for create sample is same as training sample.

The Haar-training takes weeks, if it is interrupt, it is able to restart and carry forward the training.

7. Acknowledgments

First of all my warm thanks to Dr Andre L. C. Barczak as he has given me such a chance to let me get to know object recognition and Artificially Intelligent during the course of project, which made me realize how amazing and profound it is. I broaden my knowledge in the computer vision and expand my vision from this unforgettable studying experience.

This report could not have been accomplished without great support and constant help from Dr Andre L. C. Barczak. His solid professional knowledge and enriched experience are my main supporter for the fantastic year I have spent here. I have learned a lot from him. Dr. Andre is such a generous and inspired person and he guide me the right way to complete my report. Thank Dr. Andre. for his encouragement and inspiration.

Special thanks to the IT support from IIMS lab for providing a safe and efficient lab environment.
8. Reference


Hand Detection and Gesture Recognition using ASL Gestures


[19] Z.NAN *Face Detection Based on AdaBoost* Beijing University 2005


APPENDIX 1 Take positive image use Webcam

#include <cv.h>
#include <highgui.h>
#include <cvcam.h>
#include <cvaux.h>
#include <cxcore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>

#define pixelB(image,x,y)\((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels]\n
#define pixelG(image,x,y)\((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels+1]\n
#define pixelR(image,x,y)\((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels+2]\n
char * combinchar(char * achar1,char * achar2);
char * conver_suffix(int anumber);
char * getsaveimagename(int anum);

int number = 0;   //save image count

char * combinchar(char * achar1,char * achar2){
char * TotalChar;

TotalChar = (char *)malloc((strlen(achar1)+strlen(achar2))*sizeof(char));
strcpy(TotalChar,achar1);
strcat(TotalChar,achar2);
return TotalChar;

char * getSaveImagename(int _num){
    int num = _num + 1;
    char * name = itoa(num,name,10);
    char * suffix = ".jpg";
    char * imagename = combinchar(name, suffix);
    number = num;
    return imagename;
}

int main ( int argc, char** argv ) {
    CvCapture* capture = 0;
    IplImage *image = 0;
    capture = cvCaptureFromCAM(0);
    cvNamedWindow( "cam0", 1 );
    if( capture ) {
        printf("This is TakePhoto from WebCam Tool !!\n");
        for(;;) {
            if( !cvGrabFrame( capture ) )
                {if( !cvGrabFrame( capture )}
break;

image = cvRetrieveFrame( capture );

if (!image )
    break;
int key;
key=cvWaitKey(100);
printf("key %d\n",key);

if (key==113) // Q for byebye
{
    printf("Bye Bye~~~!!");
    exit(0);
}

if (key==115 ) // S for save
{
    char * saveimagename = getsaveimagename(number);
    cvSaveImage(saveimagename,image);
    printf("Save Image!!");
}

cvShowImage( "cam0", image );
}

cvReleaseImage( &image);

cvReleaseCapture( &capture );
}

cvDestroyWindow("cam0");
return 0;
APPENDIX 2 Segment of positive image with HSV colour model

```c
#include <cv.h>
#include <highgui.h>
#include <cxcore.h>
#include <stdio.h>
#include <stdlib.h>

//Macros for colour pixels
#define pixelB(image,x,y)  
    ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels]
#define pixelG(image,x,y)  
    ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+1]
#define pixelR(image,x,y)  
    ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+2]

char* window_name="<SPACE>save and load next <Q>exit";
char* window_seg = "Result Window";

char* filename;
int numOfRec=0;
IplImage *image = 0, *segimage = 0, *final =0;

void on_mouse(int event,int x,int y,int flag,void*bla) {
    if (event == 4) {  //left for 000
        printf("X : %d",x);
        printf("Y : %d\n",y);
        pixelB(final,x,y) = 0;
    }
```
if (event == 5) { //right for undo
    printf("X : %d",x);
    printf("Y : %d",y);

    pixelB(final,x-2,y) = pixelB(image,x-2,y);
pixelG(final,x-2,y) = pixelG(image,x-2,y);
pixelR(final,x-2,y) = pixelR(image,x-2,y);

    pixelB(final,x-1,y) = pixelB(image,x-1,y);
pixelG(final,x-1,y) = pixelG(image,x-1,y);
pixelR(final,x-1,y) = pixelR(image,x-1,y);

    pixelB(final,x,y) = pixelB(image,x,y);
pixelG(final,x,y) = pixelG(image,x,y);
pixelR(final,x,y) = pixelR(image,x,y);

    pixelB(final,x+2,y) = pixelB(image,x+2,y);
pixelG(final,x+2,y) = pixelG(image,x+2,y);
pixelR(final,x+2,y) = pixelR(image,x+2,y);

    pixelB(final,x+1,y) = pixelB(image,x+1,y);
pixelG(final,x+1,y) = pixelG(image,x+1,y);
pixelR(final,x+1,y) = pixelR(image,x+1,y);

    pixelB(final,x,y+2) = pixelB(image,x,y+2);
pixelG(final,x,y+2) = pixelG(image,x,y+2);
Hand Detection and Gesture Recognition using ASL Gestures

```c
int main( int argc, char** argv ){
    int inicio = 1;
    char filename[255];
    int iKey = 0;
    if (argc != 3) {
        printf("Usage: dir image_number_to_start\n");
        exit(0);
    }
    intR(final,x,y+2) = pixelR(image,x,y+2);
    intB(final,x,y+1) = pixelB(image,x,y+1);
    intG(final,x,y+1) = pixelG(image,x,y+1);
    intR(final,x,y+1) = pixelR(image,x,y+1);
    intB(final,x,y-2) = pixelB(image,x,y-2);
    intG(final,x,y-2) = pixelG(image,x,y-2);
    intR(final,x,y-2) = pixelR(image,x,y-2);
    intB(final,x,y-1) = pixelB(image,x,y-1);
    intG(final,x,y-1) = pixelG(image,x,y-1);
    intR(final,x,y-1) = pixelR(image,x,y-1);
    }
    cvShowImage(window_seg,final);
    
}
```
cvAddSearchPath(argv[1]);
cvNamedWindow(window_name,1);
cvNamedWindow(window_seg,1);
printf("search path \"%s\n",argv[1]);

cvSetMouseCallback(window_seg,on_mouse);

inicio = atoi(argv[2]);

char strPrefix[8000];
char strPostfix[8000];

for (int i = inicio; image = 0, sprintf(filename,"%s/%d.jpg",argv[1],i), printf("trying to open \\
%\s\n",filename),image = cvLoadImage(filename,1),image != 0; i++) {
    printf("picture : %s \n",filename);
    strcpy(strPrefix,filename);
    numOfRec = 0;
    //work on current image begin
    segimage = cvCloneImage(image);
    final = cvCloneImage(image);
    cvCvtColor(image,segimage, CV_RGB2HSV);
    for (int x=0;x<image->width;x++){
        for (int y=0;y<image->height;y++){
            if( (pixelB(segimage,x,y)>110) && (pixelB(segimage,x,y) < 160) ){
                pixelB(final,x,y)=pixelB(image,x,y);
                pixelG(final,x,y)=pixelG(image,x,y);
                pixelR(final,x,y)=pixelR(image,x,y);
            } else {
                pixelB(final,x,y)=0;
                pixelG(final,x,y)=0;
            }
        }
    }
}
Hand Detection and Gesture Recognition using ASL Gestures

pixelR(final,x,y)=0;

//work on current image end

do {
    cvShowImage(window_name,image);
    cvShowImage(window_seg,final);
    iKey = cvWaitKey(0);
    printf("%d\n",iKey);
    switch(iKey){
    case 113://q key
        cvReleaseImage(&image);
        cvReleaseImage(&segimage);
        cvReleaseImage(&final);
        cvDestroyWindow(window_name);
        cvDestroyWindow(window_seg);
        printf("QUIT\n");
        return(0);
        break;
    case 99:// c key to do something
        numOfRec++;
        printf("%d\n", numOfRec);
        break;
    }
}

while(iKey!=32);//space key for save image and load next image

char * name = itoa(i,name,10);
char * suffix = ".jpg";
char * imagename = strcat(name, suffix);

cvSaveImage(imagename, final);
cvReleaseImage(&image);
cvReleaseImage(&segimage);
cvReleaseImage(&final);

}

cvDestroyWindow(window_name);
cvDestroyWindow(window_seg);

return 0;
}
APPENDIX 3 Segment of positive image with contour model

```c
#include <cv.h>
#include <highgui.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

#define pixelB(image,x,y)\
((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels]
#define pixelG(image,x,y)\
((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels+1]
#define pixelR(image,x,y)\
((uchar *)(image->imageData + y*image->widthStep))[x*image->nChannels+2]

char * filename;
char * outputname;
char * outputtxt;

IplImage * src = 0;
IplImage * image1 = 0;
IplImage * imageinit = 0;
IplImage * imagesmooth = 0;
IplImage * imagecanny = 0;
IplImage * imagecontour = 0;
IplImage * outimage = 0;
```
int edge_thresh = 1;
int edge_thresholding = 1;

void draw(IplImage *p, int x, int y);
void showcenter(int ax, int ay);
void thresholding(IplImage *p, int _t);
void increase(IplImage *p);

int findPin(IplImage *p);
uchar** findNeighbour(IplImage *image, int pos_x, int pos_y);

void on_trackbar_thresholding(int h){
    outimage = cvCloneImage(imagecontour);
    thresholding(outimage, edge_thresholding);
    cvShowImage("thresholding", outimage);
}

void on_trackbar(int h){
    cvSmooth(imageinit, imagesmooth, CV_BLUR, 3,3,0,0);
    cvCanny(imagesmooth, imagecanny, (float)edge_thresh, (float)edge_thresh*3,3);
    cvShowImage("gray image1", image1);
    cvShowImage("imagesmooth", imagesmooth);
    cvShowImage("imagecanny", imagecanny);
}

bool check(IplImage *p, int x, int y){
    if ((pixelR(p,x+1,y)===0) && (pixelG(p,x+1,y)===0) && (pixelB(p,x+1,y)===0) ) {
        if ((pixelR(p,x-1,y)===0) && (pixelG(p,x-1,y)===0) && (pixelB(p,x-1,y)===0) ) {

```
if ((pixelR(p,x,y-1)== 0) && (pixelG(p,x,y-1)==0) && (pixelB(p,x,y-1))==0) {
    if ((pixelR(p,x,y+1)== 0) && (pixelG(p,x,y+1)==0) && (pixelB(p,x,y+1))==0) {
        return false;
    } else {
        return true;
    }
} else {
    return true;
}

void usemask(char * fn){
    IplImage *pImg=cvLoadImage(fn,1);
    IplImage *mask=cvLoadImage("mask.jpg",0);
    IplImage *dstImg;
    dstImg = cvCreateImage(cvSize(pImg ->width,pImg->height),IPL_DEPTH_8U,3);
    cvSetZero( dstImg );
    cvCopy(pImg,dstImg,mask);
    for (int i = 0; i < dstImg ->width; i++) {
        for (int j = 0; j < dstImg ->height; j++) {
            
                
        }
    }
}
if ((pixelB(dstImg,i,j) !=0) || (pixelR(dstImg,i,j) !=0) || (pixelG(dstImg,i,j) !=0)) {
if (((i <= 20) && (j <= 20)) || ((i >= 300) && (j <= 20)) || ((i <= 20) && (j >= 220))
|| ((i >= 300) && (j >= 220))){
        pixelB(dstImg,i,j) = 0;
        pixelR(dstImg,i,j) = 0;
        pixelG(dstImg,i,j) = 0;
    }
}

cvNamedWindow("seg image",7);
cvShowImage("seg image",dstImg);
char *path = "./0/";
cvSaveImage(strcat(path,filename),dstImg);

int main(int argc,char **argv){

    if(argc==2){
        filename = argv[1]; //input image;
    }
    else {
        printf("Parameters Error!!\n");
        exit(0);
    }

    int iKey = 0;
if((src=cvLoadImage(filename,1))==0){printf("Load file name Error"); return -1;}

image1=cvCreateImage(cvSize(src->width,src->height),IPL_DEPTH_8U,1);
imageinit=cvCreateImage(cvSize(image1->width,image1->height),IPL_DEPTH_8U,1);
imagesmooth=cvCreateImage(cvSize(image1->width,image1->height),IPL_DEPTH_8U,1);
imagecanny=cvCreateImage(cvSize(image1->width,image1->height),IPL_DEPTH_8U,1);
imagecontour = cvCreateImage(cvSize(image1->width,image1->height),IPL_DEPTH_8U,1);
outimage = cvCreateImage(cvSize(image1->width,image1->height),IPL_DEPTH_8U,1);

cvCvtColor(src, image1, CV_BGR2GRAY);
cvCvtColor(src,imageinit,CV_BGR2GRAY);
cvCvtColor(src,imagecanny,CV_BGR2GRAY);
cvCvtColor(src,imagecontour,CV_BGR2GRAY);
cvCvtColor(src,outimage,CV_BGR2GRAY);

cvNamedWindow("gray image1",2);
cvNamedWindow("imagesmooth",3);
cvNamedWindow("imagecanny",4);
cvNamedWindow("imagecontour",5);
cvNamedWindow("thresholding",6);

cvCreateTrackbar("threshold", "imagecanny", &edge_thresh, 100, on_trackbar);
on_trackbar(0);
CvBox2D * _mybox;
CvPoint2D32f * _PointArray32f;
CvPoint _myCenter;
CvPoint * _PointArray;
_mybox = (CvBox2D *) malloc(sizeof(CvBox2D));
CvMemStorage * storage = cvCreateMemStorage(1000);
CvSeq * contour;
int mode = CV_RETR_EXTERNAL;
int count,i;

while (1){
iKey=cvWaitKey(10);
printf(" %d \n", iKey);
switch(iKey){
    case 115:// s for save mask
        cvSaveImage("mask.jpg",outimage);
        break;
    case 109: //m for use mask
        usemask(filename);
        break;
    case 116: //t for thresholding
        cvNamedWindow("thresholding",6);
        cvCreateTrackbar("thresholding", "thresholding", &edge_thresholding, 256,
        on_trackbar_thresholding);
        on_trackbar_thresholding(128);
        break;
    case  99: //c for Contours
        cvFindContours(imagecanny, storage, &contour, sizeof(CvContour),
        mode, CV_CHAIN_APPROX_SIMPLE);
cvDrawContours(imagecontour, contour,
    CV_RGB(0,0,255), CV_RGB(255, 0, 0),
    2, 2, 8, cvPoint(0,0));

cvShowImage("imagecontour",imagecontour);
break;
//case 114: //r for save image
    // cvSaveImage();
    // break;
    case 113: //q
        exit(0);
    }
}

free(contour);
free(_mybox);

cvReleaseMemStorage(&storage);
cvReleaseImage(&src);
cvReleaseImage(&image1);
cvReleaseImage(&imageinit);
cvReleaseImage(&imagesmooth);
cvReleaseImage(&imagecanny);
cvReleaseImage(&imagecontour);
cvReleaseImage(&outimage);

return 0;
void thresholding(IplImage *p, int _t) {
    uchar *pixel;
    for(int pos_y=0; pos_y<p->height; pos_y++) {
        for(int pos_x=0; pos_x<p->width; pos_x++) {
            pixel=(uchar*)(p->imageData+p->widthStep*pos_y)+pos_x;
            if(*pixel<=_t) {
                *pixel=0;
            } else {
                *pixel=255;
            }
        }
    }
}
APPENDIX 4 Replace background to random background

#include <cv.h>
#include <highgui.h>
#include <cxcore.h>
#include <cvaux.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>

//Macros for colour pixels
#define pixelB(image,x,y)  
 ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels]
#define pixelG(image,x,y)  
 ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+1]
#define pixelR(image,x,y)  
 ((uchar *)(image->imageData + (y)*image->widthStep))[(x)*image->nChannels+2]

char * combinchar(char * achar1,char * achar2);
char * getsaveimagename(int _num);
int number = 0;   //loop count
int gennum = 0;  //begin name no.
char * combinchar(char * achar1, char * achar2) {
    char * TotalChar;
    TotalChar = (char *)malloc((strlen(achar1) + strlen(achar2)) * sizeof(char));
    strcpy(TotalChar, achar1);
    strcat(TotalChar, achar2);
    return TotalChar;
}

char * getsaveimagename(int _num) {
    int num = _num + 1;
    char * name = itoa(num, name, 10);
    char * suffix = ".jpg";
    char * imagename = combinchar(name, suffix);
    number = num;
    return imagename;
}

char filename[255];
IplImage *image = 0;
IplImage *bgimage = 0;
CvPoint cp1;
CvPoint cp2;
int minx = 0;
int miny = 0;
int maxx = 0;
int maxy = 0;
int width = 0;
int high = 0;
int num = 0;
int main( int argc, char** argv ){
    if (argc !=5) {
        printf("Usage <source image dir><start image number><background image><number of each image> \n");
        exit(0);
    }
    cvAddSearchPath(argv[1]);
    printf("search path: %s\n",argv[1]);
    bgimage = cvLoadImage(argv[3],1);
    num = atoi(argv[4]);
    int inicio = 1;
    inicio = atoi(argv[2]);
    //------------Create Windows----------------
    cvNamedWindow( "positive", 1 );//
    cvNamedWindow( "negative", 1 );//
    //-------------------get subwindow--------------
    //sprintf(filename, "%s/%d.jpg",argv[1],i);
    for (int i = inicio; image = 0, sprintf(filename, "%s/%d.jpg",argv[1],i),
        printf("trying to open %s\n",filename),image = cvLoadImage(filename,1),image != 0; i++) {
        printf("picture : %s \n",filename);
        for (int x=0;x<image->width;x++){
            for (int y=0;y<image->height;y++){
                if((pixelB(image,x,y)>=2 && pixelB(image,x,y)<=255)
                    && (pixelG(image,x,y)>=36 && pixelG(image,x,y)<=255)
                    && (pixelR(image,x,y)>=126 && pixelR(image,x,y)<=255))
                {
                    if ((maxx == 0)&&(maxy == 0)&&(minx == 0)&&(miny == 0))
                    {
                        printf("positive\n\n");
                        printf("negative\n\n");
                    }
                }
            }
        }
    }
}
0)) {
    maxx = x;
    maxy = y;
    minx = x;
    miny = y;
}
    if (x > maxx) maxx = x;
    if (x < minx) minx = x;
    if (y > maxy) maxy = y;
    if (y < miny) miny = y;
}
}
}

cp1 = cvPoint(minx,miny);
cp2 = cvPoint(maxx,maxy);
width = maxx - minx;
high = maxy - miny;
CvRect Rect1 = cvRect(minx,miny,width,high);

cvSetImageROI(image,Rect1);

CvSize ImageSize = cvSize(image->width,image->height); //new size

//----------show image-------------------------
    minx = 0;
    miny = 0;
    maxx = 0;
    maxy = 0;
    width = 0;
    high = 0;
//--------Loop for get number of images

    while (number < num) {
        IplImage * B = 0;  //background
        B = cvCreateImage(ImageSize,image->depth,image->nChannels);

        int ax = 0;
        int ay = 0;

        int bx = random(500);  //random
        int by = random(500);  //random

        for (int x = bx; x < (image->width + bx); x++) {
            for (int y = by; y < (image->height + by); y++) {
                pixelB(B,ax,ay)=pixelB(bgimage,x,y);
                pixelG(B,ax,ay)=pixelG(bgimage,x,y);
                pixelR(B,ax,ay)=pixelR(bgimage,x,y);
                ay ++;
            }
            ay = 0;
            ax ++;
        }

        for (int i = 0; i < image->width; i++) {
            for (int j = 0; j < image->height ; j++) {
                if((pixelB(image,i,j)>15 && pixelB(image,i,j)<255)
                && (pixelG(image,i,j)>15 && pixelG(image,i,j)<255)
                && (pixelR(image,i,j)>15 && pixelR(image,i,j)<255)){
                    pixelB(B,i,j)=pixelB(image,i,j);
                    pixelG(B,i,j)=pixelG(image,i,j);
                    pixelR(B,i,j)=pixelR(image,i,j);
                }
            }
        }
    }
cvSetImageROI(B,Rect1);

char imagename[8];
sprintf(imagename,"%d.jpg",gennum);
cvSaveImage(imagename,B);
printf("saved: %s\n",imagename);
cvDestroyWindow( "B" );//
cvReleaseImage( &B ); //
gennum++;
number++;
}
number = 0;
cvReleaseImage( &image );
}

cvDestroyWindow( "positive" );
cvDestroyWindow( "negative" );

return 0;
}
APPENDIX 5 Using Classifier

#include "cv.h"
#include "highgui.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>

#ifdef _EiC
#define WIN32
#endif

static CvMemStorage* storage = 0;
static CvHaarClassifierCascade* cascade_0 = 0;
static CvHaarClassifierCascade* cascade_1 = 0;
static CvHaarClassifierCascade* cascade_2 = 0;
static CvHaarClassifierCascade* cascade_3 = 0;
static CvHaarClassifierCascade* cascade_4 = 0;
static CvHaarClassifierCascade* cascade_5 = 0;

void detect_and_draw( IplImage* image );

const char* cascade_name_0 = 
"gestrue0.xml";

const char* cascade_name_1 =
"gestrue1.xml";

const char* cascade_name_2 =
"gestrue2.xml";

const char* cascade_name_3 =
"gestrue3.xml";

const char* cascade_name_4 =
"gestrue4.xml";

const char* cascade_name_5 =
"gestrue5.xml";

int findcount = 0;

int main( int argc, char** argv )
{
    CvCapture* capture = 0;
    IplImage *frame, *frame_copy = 0;
    int optlen = strlen("--cascade=");
    const char* input_name;

    if( argc > 1 && strcmp( argv[1], "--cascade=" ) == 0 )
    {
        cascade_name_0 = argv[1] + optlen;
        cascade_name_1 = argv[2];
cascade_name_2 = argv[3];
cascade_name_3 = argv[4];
cascade_name_4 = argv[5];
cascade_name_5 = argv[6];

input_name = argc > 7 ? argv[7] : 0;
}
else
{
    //cascade_name = "/data/haarcascades/haarcascade_frontalface_alt2.xml";
cascade_name_0 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue0_stage8.xml";
cascade_name_1 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue1_stage9.xml";
cascade_name_2 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue2_stage12.xml";
cascade_name_3 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue3_stage13.xml";
cascade_name_4 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue4_stage14.xml";
cascade_name_5 = "E:\kk\Study at Massey\Project(159.793)\use cascade to find gesture\Debug\gestrue5_stage15.xml";

    //input_name = argc > 1 ? argv[1] : 0;
    input_name = 0;  //default use webcam
}

cascade_0 = (CvHaarClassifierCascade*)cvLoad( cascade_name_0, 0, 0, 0 );
cascade_1 = (CvHaarClassifierCascade*)cvLoad( cascade_name_1, 0, 0, 0 );
cascade_2 = (CvHaarClassifierCascade*)cvLoad( cascade_name_2, 0, 0, 0 );
cascade_3 = (CvHaarClassifierCascade*)cvLoad( cascade_name_3, 0, 0, 0 );
cascade_4 = (CvHaarClassifierCascade*)cvLoad( cascade_name_4, 0, 0, 0 );
cascade_5 = (CvHaarClassifierCascade*)cvLoad( cascade_name_5, 0, 0, 0 );

if(
(!cascade_0) && (cascade_1) && (cascade_2) && (cascade_3) && (cascade_4) && (cascade_5) )
{
    fprintf( stderr, "ERROR: Could not load classifier cascade.\n" );
    fprintf( stderr,
"Usage: facedetect --cascade="<cascade_path>" [filename|camera_index]\n"
    );
    return -1;
}
storage = cvCreateMemStorage(0);

if (!input_name || (isdigit(input_name[0]) && input_name[1] == '0') )
capture = cvCaptureFromCAM( !input_name ? 0 : input_name[0] - '0' );
else
    capture = cvCaptureFromAVI( input_name );

cvNamedWindow( "result", 1 );
//cvNamedWindow( "Recognition", 1 );

if( capture )
{
    for(;;)
    {
        if( !cvGrabFrame( capture ) )
            break;
        frame = cvRetrieveFrame( capture );
        if( !frame )
break;
if( !frame_copy )
    frame_copy = cvCreateImage( cvSize(frame->width,frame->height),
        IPL_DEPTH_8U, frame->nChannels );
if( frame->origin == IPL_ORIGIN_TL )
    cvCopy( frame, frame_copy, 0 );
else
    cvFlip( frame, frame_copy, 0 );
detect_and_draw( frame_copy );

if( cvWaitKey( 10 ) >= 0 )
    break;
}
cvReleaseImage( &frame_copy );
cvReleaseCapture( &capture );
}
else
{
    const char* filename = input_name ? input_name : (char*)"lena.jpg";
    IplImage* image = cvLoadImage( filename, 1 );

    if( image )
    {
        detect_and_draw( image );
        cvWaitKey(0);
        cvReleaseImage( &image );
    }
    else
Hand Detection and Gesture Recognition using ASL Gestures

{ /* assume it is a text file containing the
   list of the image filenames to be processed - one per line */

    FILE* f = fopen( filename, "rt" );
    if( f )
    {
        char buf[1000+1];
        while( fgets( buf, 1000, f ) )
        {
            int len = (int)strlen(buf);
            while( len > 0 && isspace(buf[len-1]) )
                len--;
            buf[len] = '0';
            image = cvLoadImage( buf, 1 );
            if( image )
            {
                detect_and_draw( image );
                cvWaitKey(0);
                cvReleaseImage( &image );
            }
        }
    }
    fclose(f);
}

cvDestroyWindow("result");

//=cvDestroyWindow("Recognition");
void detect_and_draw( IplImage* img )
{
static CvScalar colors[] =
{
    {{0,0,255}},
    {{0,128,255}},
    {{0,255,255}},
    {{0,255,0}},
    {{255,128,0}},
    {{255,255,0}},
    {{255,0,0}},
    {{255,0,255}}
};

double scale = 1.3;
IplImage* gray = cvCreateImage( cvSize(img->width,img->height), 8, 1 );
IplImage* small_img = cvCreateImage( cvSize( cvRound (img->width/scale), cvRound (img->height/scale)), 8, 1 );
cvCvtColor( img, gray, CV_BGR2GRAY);
cvResize( gray, small_img, CV_INTER_LINEAR );
cvEqualizeHist( small_img, small_img );

int i;
int R = -1; //nothing

cvCvtColor( img, gray, CV_BGR2GRAY);
cvResize( gray, small_img, CV_INTER_LINEAR );
cvEqualizeHist( small_img, small_img );
cvClearMemStorage( storage );

if( cascade_0 && cascade_1 && cascade_2 && cascade_3 && cascade_4 && cascade_5 )
{
    double t = (double)cvGetTickCount();

    CvSeq* gesture_0 = cvHaarDetectObjects( small_img, cascade_0, storage,
                                        1.1, 1,
                                        CV_HAAR_DO_CANNY_PRUNING*/,
                                        cvSize(20, 25) );
    for( i = 0; i < ( gesture_0 ? gesture_0->total : 0); i++ )
    {
        CvRect* r = (CvRect*)cvGetSeqElem( gesture_0, i );
        CvPoint center;
        int radius;
        center.x = cvRound((r->x + r->width*0.5)*scale);
        center.y = cvRound((r->y + r->height*0.5)*scale);
        radius = cvRound((r->width + r->height)*0.25*scale);
        cvCircle( img, center, radius, colors[i%8], 3, 8, 0 );
        R = 0;
        findcount++;
    }

    CvSeq* gesture_1 = cvHaarDetectObjects( small_img, cascade_1, storage,
                                        1.1, 1,
                                        CV_HAAR_DO_CANNY_PRUNING*/,
                                        cvSize(20, 34) );
    for( i = 0; i < ( gesture_1 ? gesture_1->total : 0); i++ )
    {
    ...
CvRect* r = (CvRect*)cvGetSeqElem( gesture_1, i );

CvPoint center;
int radius;
center.x = cvRound((r->x + r->width*0.5)*scale);
center.y = cvRound((r->y + r->height*0.5)*scale);
radius = cvRound((r->width + r->height)*0.25*scale);
cvCircle( img, center, radius, colors[i%8], 3, 8, 0 );
R = 1;
findcount++;
}

CvSeq* gesture_2 = cvHaarDetectObjects( small_img, cascade_2, storage,
1.1,
1, 1/*CV_HAAR_DO_CANNY_PRUNING*/,
cvSize(20, 29) );

for( i = 0; i < ( gesture_2 ? gesture_2->total : 0); i++ )
{
CvRect* r = (CvRect*)cvGetSeqElem( gesture_2, i );
CvPoint center;
int radius;
center.x = cvRound((r->x + r->width*0.5)*scale);
center.y = cvRound((r->y + r->height*0.5)*scale);
radius = cvRound((r->width + r->height)*0.25*scale);
cvCircle( img, center, radius, colors[i%8], 3, 8, 0 );
R = 2;
findcount++;
}
CvSeq* gesture_3 = cvHaarDetectObjects( small_img, cascade_3, storage,
1.1, 1,
1/*CV_HAAR_DO_CANNY_PRUNING*/,
_cvSize(20, 19);
for( i = 0; i < (gesture_3 ? gesture_3->total : 0); i++ )
{
    CvRect* r = (CvRect*)cvGetSeqElem( gesture_3, i);
    CvPoint center;
    int radius;
    center.x = cvRound((r->x + r->width*0.5)*scale);
    center.y = cvRound((r->y + r->height*0.5)*scale);
    radius = cvRound((r->width + r->height)*0.25*scale);
    cvCircle( img, center, radius, colors[i%8], 3, 8, 0);
    R = 3;
    findcount++;
}

CvSeq* gesture_4 = cvHaarDetectObjects( small_img, cascade_4, storage,
1.1, 1,
1/*CV_HAAR_DO_CANNY_PRUNING*/,
_cvSize(20, 21);
for( i = 0; i < (gesture_4 ? gesture_4->total : 0); i++ )
{
    CvRect* r = (CvRect*)cvGetSeqElem( gesture_4, i);
    CvPoint center;
    int radius;
    center.x = cvRound((r->x + r->width*0.5)*scale);
    center.y = cvRound((r->y + r->height*0.5)*scale);
    radius = cvRound((r->width + r->height)*0.25*scale);
    cvCircle( img, center, radius, colors[i%8], 3, 8, 0);
R = 4;
findcount++;
}

CvSeq* gesture_5 = cvHaarDetectObjects( small_img, cascade_5, storage,
1.1,
1/*CV_HAAR_DO_CANNY_PRUNING*/,
cvSize(20, 18);)
for( i = 0; i < ( gesture_5 ? gesture_5->total : 0); i++ )
{
    CvRect* r = (CvRect*)cvGetSeqElem( gesture_5, i );
    CvPoint center;
    int radius;
    center.x = cvRound((r->x + r->width*0.5)*scale);
    center.y = cvRound((r->y + r->height*0.5)*scale);
    radius = cvRound((r->width + r->height)*0.25*scale);
    cvCircle( img, center, radius, colors[i%8], 3, 8, 0 );
    R = 5;
    findcount++;
}

t = (double)cvGetTickCount() - t;
printf( "detection time = %gms\n", t/((double)cvGetTickFrequency()*1000.) );
if (R != -1) {
    printf( "Recognition : %d\n", R);
}
}
Hand Detection and Gesture Recognition using ASL Gestures

CvFont Font1;
cvInitFont(&Font1,CV_FONT_VECTOR0,5,5,0,7,8);
CvPoint TP = cvPoint(50,150);
CvScalar Color = CV_RGB(255,0,0);
char * info = "info";
info = itoa(R,info,10);
cvPutText(img,info,TP,&Font1,Color);

cvShowImage( "result'', img );
//cvShowImage("Recognition",simg);
cvReleaseImage( &gray );
cvReleaseImage( &small_img );
//cvReleaseImage( &simg );
}