Image-Based
Face Detection System

Candidate : Tsang Tat Man
Student No.: 0301-0506-0333
Supervisor : Lo Chi Wing, Peter
Module Number: U08096
Programme: Bsc (Hons) Computer and Information System
University: Oxford Brookes University

STUDENT'S PROJECT SUMMARY

SURNAME TSANG FORENAME TAT MAN

PROJECT TITLE IMAGE-BASED FACE DETECTION SYSTEM

1) Use not more then 150 words to descript your project briefly and what are the main objects of your project.

Face is one of the physiological biometrics based on stable features. Face detection is a challenging mission. It is because faces in the images are all uncontrolled.

AdaBoost is deployed and used to detect face. AdaBoost is a machine learning algorithms which if formed from a sequence of weak classifier. Each of weak classifier forms a cascade and contributes a strong classifier in this project as face detector.

The given input image is experienced with pre-processing section before doing face detection, because of enhance images quality. They are "Image Resizing", "Grayscale Conversion" and "Histogram Equalization".

The aim of this project is to develop and propose a system to detect human faces in digital images effectively, no matters what person's ethnic, pose.

2) What is the implication of your project? Eg .your idea can it be used as a solution for some other problem ? Or any relevance to the current issues in computing and the application domain

In this project, AdaBoost algorithm is used. This algorithms can be solved any binary classification problem. This algorithm not only can be solved face detection. It can be deployed in any problem about binary classification. E.g. License Plate Detection.

3) What methodology you applied in your project and how you applied . Eg. Any Data structure stack, tree..), algorithm (Searching, Sorting ,graph theory..) or database design, etc.

Methodology in the project can be divided into two parts.

The first part is pre-processing section, which can be divided into “Grayscale conversion”, “image resizing” and “Histogram Equalization”.

Second part is algorithm section, which are “Integral Image”, “AdaBoost” and “Cascade”.

4) What techniques you used in solving the problem and how the problem was solved. Eg. ASP, JSP, DAO, CGI , Active-X, Database Server, SQL, Find Shortest Path or you own method.

I used “Python” programming language. It is a script programming language and can be executed with
different system platform. e.g. Linux, Windows, Mac OSX platform. Besides using python, I also used OpenCV and PyQt4 in the programming code, these methods can be speed up my developing time.

5) If you do this project again which area you should / can improve ?eg. Technique , written presentation, module, research area, scope ,complexity, capacity , etc.

If I have time to improve this project, I will try to add more detectors in my project to increase the detection rate. E.g. Eye detector can be added for each detected face, as a result, face detection rate should be improved.

6) What you have learned from doing this project ?

I learned many stuff in this project. First, I know how to digest data from the internet and contribute to my project. Second, I learned about the algorithm of AdaBoost, which can be solved about the pattern classification. I knew this algorithm can be worked in real time, because of deploy integral image.

Also, I found more algorithms can be handled face detection, not only single method can do. E.g. PCA, SVM.

Acknowledgements

I would like to sincere thank my project supervisor, Peter Lo. His always suggests useful information and give me the guidance of the project.

Abstract

Face detection is an important step in face recognition, which is one of the more representative and classic application in computer vision. Face is one of the physiological biometrics based on stable features. Face detection is a challenging mission. It is because faces in the images are all uncontrolled. E.g. illumination condition, vary pose, different facial expressions.

Four types of face detection are developed. They are knowledge based methods, template marching, invariant feature methods and learning based methods. In this project, learning based methods are deployed: AdaBoost Method.

AdaBoost is a machine learning algorithms which if formed from a sequence of weak classifier. Each of weak classifier forms a cascade and contributes a strong classifier in this project as face detector.

The given input image is experienced with pre-processing section before doing face detection, because of enhance images quality. They are “image resizing”, “grayscale conversion” and “histogram equalization”.

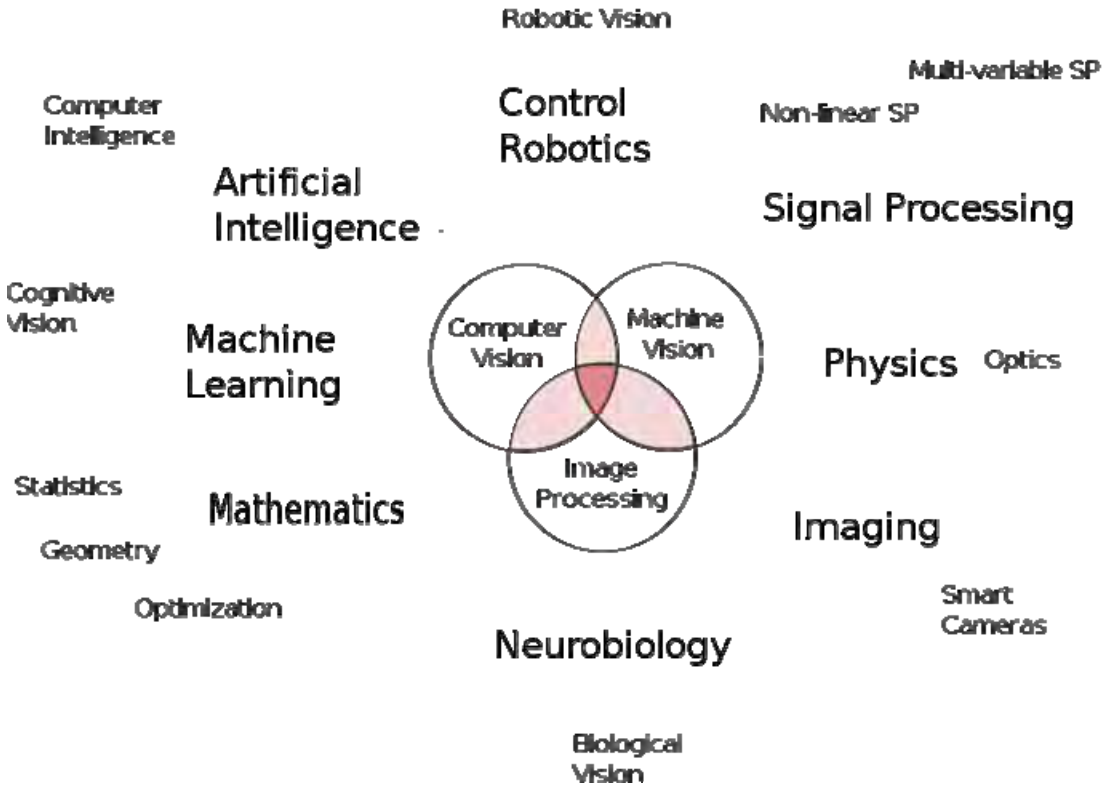
List of abbreviations

CCTV	-	Closed-circuit television
CDF	-	Cumulative Distribution Function
FDS	-	Face Detection System
GUI	-	Graphic User Interface
NN	-	Neural Network
PCA	-	Principle Component analysis
PDF	-	Probability Density Function
RGB	-	RGB color model
SVM	-	Support Vector Machine
YCbCr	-	Color space for a digital color system

1. Introduction

Background

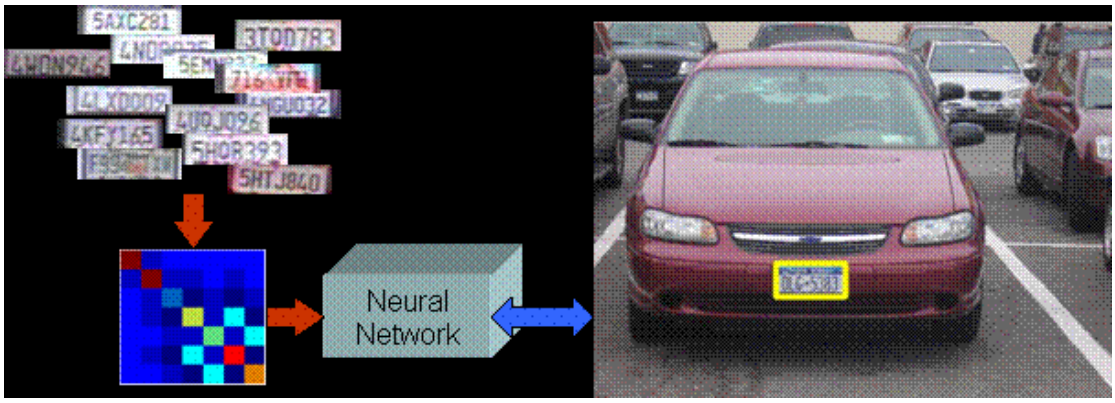
Computer Vision is a very new area of research is currently hanging in the Computer Science under. It mainly want to solve the problem is to build an intelligent system, extracted from the images useful information, of course, we could simply say, the ultimate purpose of computer vision is to create such as like humans can be made for the image the wisdom of response and identification systems. The following chart is the relation between different fields.[1]



Overview of Computer vision

Face Recognition is one of the more representative and classic application in Computer Vision. Face Detection is the most important step of face recognition. Not only the face recognition, face detection also a first step in Human Computer Interaction (HCI) systems. E.g. expression recognition. Unlike traditional HCI device, keyboard, mouse and display, it provides more effective methods to increase user experience with computer used. As a result, it speeds up human's work. It's conveyed information from physics world into logical thinking to control the computer system.

In additional, face detection is one of an object detection which to classify the desired object from the given images/video and located it. License Plate Detection is one of examples about the object detection.



In biometric approaches, human faces are unique object like fingerprints, iris which widely used in security issues. Many types of personal authentications system have been developed related this approaches which takes advantage of unique and special characteristics. System can be searched effectively to screen out useful information (face) from dozes of video media or photos from internets.

For example, Video surveillance in UK, there is one CCTV cameras for every 14 people.[2] They also need to analyze all these video, which use face detection to extract any useful information and store it for further used.

Human faces are non-rigid objects and appeared in different scale, pose, angle and facial expressions. Human faces always have variations for example, glasses. In additional, the images have different brightness, contrast. These results are the challenge of the face detection.

Objectives

The aim of this project is to develop and propose a system to detect human faces in digital images effectively, no matters what person’s ethnic, pose. Input images may be varied with face size, complex of background and illumination condition.

Also, cross-platform is preferred because of it can be worked not only the windows-platform to increase the utilization.

Hardware and Software requirements

Hardware

This system has been run and tested on following configuration. In order to get better performance, update RAM to 2GB or more is recommended.

Here is a list of minimum hardware requirements:

1. Intel Pentium Processor or any other compatible Processor, 1 GHz or greater
2. Minimum 128 MB of RAM capacity or more
3. Minimum of 32 MB Graphic Card RAM capacities or more
4. Recommended hard disk space of 1GB or more.

Software

This system was developed by cross-platform component. E.g. python and pyqt4. It means that face detection system can be operated in windows-platform, mac system, Linux-related system. These software tools will be discussed in later section.

Here is a list of software requirements:

1. Windows 2000/XP or above, Linux. Mac OS X, Unix
2. Python 2.5
3. PyQt-Py2.5

Tools to be used in this project

In this section, tools will be described into following:

Python

Python is high-level programming language like a Perl, Ruby and Tcl which are used as a scripting language.[3] It was conceived by Guido van Rossum in 1989.[4] Also, python is one of the three “official languages” in Google which means that more application in Google was deployed this language. E.g. Google App Engine SDK. Here are the points that make python is selected:

1. Free, Python is product of open source. People allows to use it in business or commercial without any charge.
2. Easy to read, Syntax in Python is clear and readable. Beginner can be easily to read and handle Python’s coding very well.
3. Rapid development, it is because it likes pseudo code. Everything coding in Python is direct result.
4. Highly portability, Python is working on different platforms, because of Python is written portable ANSI C.
5. Reusability, Python is easily reused modules and packages. Peoples can be developed their own library and reused it later project.
6. Object-Oriented Programming. Unlike scripting language, Python is designed to be object-oriented. OO programming means you can implemented using idea of inheritance and polymorphism.

PyQt4

PyQt4 is a set of Python bindings for Qt4 [5]. PyQt4 has at least 300 classes and 6,000 functions and methods. PyQt4 support Linux, Windows and Mac OSX platform. Classes are including with different field of programming. E.g. GUI, Network sockets, thread, Unicode, SVG, OpenGL, SQL database, Web browser, XML, Active X.[6]

PyQt4 also has the following advantages:

1. Cross-application framework. like Qt, it’s worked in different system platform.
2. GUI designer, PyQt4 has own GUI designer, it increases the time of development timing.
3. PyQt4 has all the advantage of Python, easy to read/code, resusability

OpenCV

OpenCV is a synonym of Open Computer Vision Library, which has at least 500 algorithms, documentation and sample code for real time computer vision.[7] OpenCV is originally developed by Intel and launched in 1999. It free for commercial and research used. [8] OpenCV library is cross-platform which means its can execute on Windows, Mac OS X, Linux, PSP and other embedded devices.

The library is mainly written in C, which makes it easily possible to transfer into specific platforms.

Example application of OpenCV library is Human-Computer Interaction, Object Identification, Segmentation and Recognition and so on.[9] Stanley was implemented by OpenCV, which was the winning entry to the 2005 DARPA Grand Challenge race. [10]

Chapter Summary

This chapter is presented the background of computer vision to address the important of face detection in our real life. Objective of project is being presented and let the people know the basic requirement of hardware and software which tools are used to implement.

2. Literature Review

In this section, face detection methods will be introduced with short overview to give any general ideal about the history of face detection and the future approaches of it.

Introduction

Face Detection is a technology to determine human face in videos and arbitrary (digital) images[11]. The aim of face detection is detect faces in any images or videos. If yes, it's reported the location on the images or videos. It just only concerns facial features in testing images. In other words, face detection can be regarded as a pattern classification. Face detection is the first part of the face recognition, it is because we need to clarify facial feature before doing recognition step.

Face detection can be regarded as a specific case of object-class detection.[12] For the detection is to locate the face in the digital images/video stream, no matter what the pose, scale, facial expressions. In other words, face detection algorithms to handle pattern classification. It task to identify a given images to decides it has face or not.

The following items are the difference between face detection and other face processing [13]:

1. face detection: To determine any face in given images/ video
2. face localization: To locate the location of the face in given images/ video
3. face recognition: To recognize faces in given images/ video

History of face detection

In the early stage, face detection algorithms mainly focused to detect the frontal human face. However, newer algorithms try to consider the different view of face as a core of face detection.

The first of face detection system has been developed since in early 1970's.[14] Due to the limitation of computation, system can't be satisfied the requirement of users, which is identify passport photograph real time.

At the beginning of 1990's, techniques are proposed focused on the face recognition on and increase the need of face detection. Many system were constructed to deal with video streaming. In the past few years, lots of methods are developed at least more than 150 methods.

Important for Face Detection

Face detection is not just classified the target object (face) in the images, because of lot of variation, pose variation, facial occlusion. The challenges of face detection discuss on later section of this chapter.

However, the following items are explained how the important of face detection:

1. Face recognition: Face recognition consists in the meaning of identification and verification of the people. These considerations are the security issues.
2. Human Computer Interaction (HCI): Human Computer Interaction is the study of interaction between human and computer.[15] It is important that it is increased the user experience with machine, as a result intelligent human computer interaction system is construct. For example, facial expression recognition, helping the disabled people.
3. Facial Expression Recognition: This technique try to figure out the meaning of expression from detected people.

The following tables are the example application use of the face detection:

Typical Application	Research Area
Immigration Management	Public Security
Monitoring sensitive characters (spies, terrorists, etc.)	
Automated Login system	Human Computer Interaction (HCI)
Realistic virtual games	
E-commerce authentication	Financial Security

The following research areas are the summary of Face detection/recognition:

1. Biometrics
2. Human Computer Interaction
3. Facial Image coding/ compression
4. Facial Expression analysis
5. Emotional computing
6. Face attribute classification
7. Attractiveness judge

Method Approaches for face detection

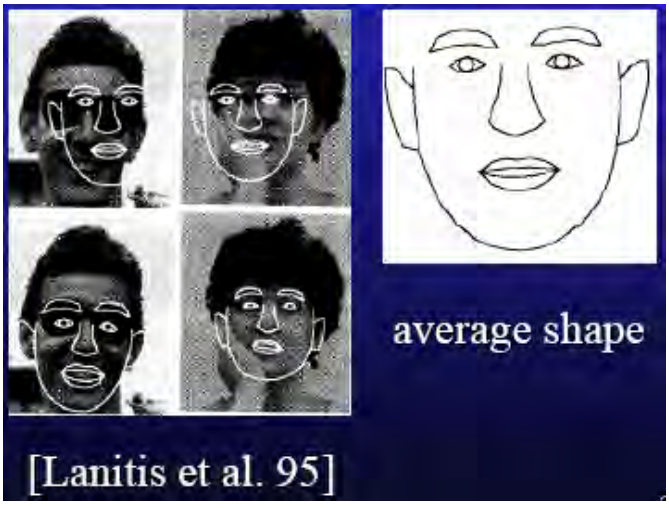
In general, FD can be implemented by four methods: knowledge based methods, template matching, invariant feature methods and learning based methods. These methods will be introduced with the following:[16]

1. **Knowledge based methods:** The models are used human knowledge to find a face patterns from the testing images. Based on the nature of human faces, algorithms scan the image from top-to-bottom and left-to-right order to find facial feature. For instance, face should be including two eyes and mouth…

   Pros: Easy applied into simple rules

   Cons: difficult to detect in invariant background, such as different pose, uncontrolled illumination. Well results based on well-defined rules. This algorithm does not work on the pose.

2. **Template marching:** The model is used several templates to find out the face class and extract facial features. Rules are pre-defined and decide whether there is face in the image. For instance, using filters to extract the contours of face shape



Sample of template marching

   Pros: Simple to apply this method.
   Cons: similar to knowledge based method, hard to detect face in different poses. Algorithms are sensitive to scale size, face shape and pose.

3. **Invariant feature methods:** The model is bottom-up approaches and used to find a facial feature (eyebrows, nose), even in the presence of composition, perspective vary, so it is difficult to find a face real time using this method. Statistical models are developed to determine the faces. Facial features of human faces are: shape, texture, skin.

   Pros: Unlike knowledge-based method, it is invariant to pose and expression.
   Cons: not suitable to detect facial features from uncontrolled background, time consuming algorithms. Detection rate is not accuracy, because of need to combine different feature and processing it.

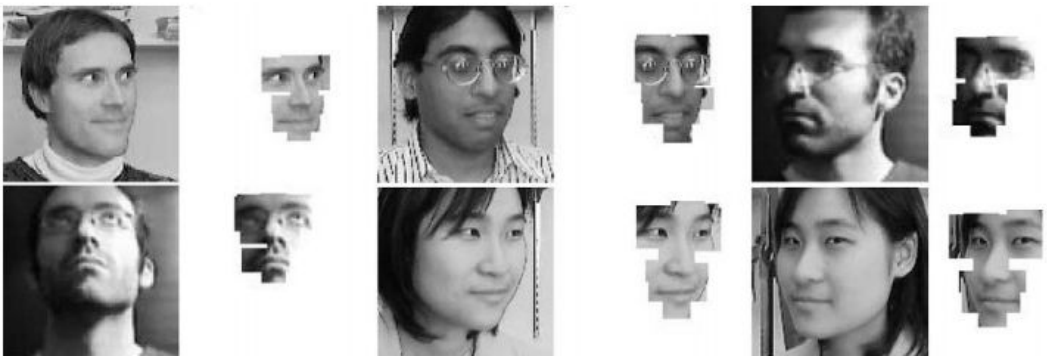
4. **Learning based methods:** The models are trained from a set of training set before doing detection. For the large amount of training data, it can be provided high accuracy recognition rate to resist variation, expression and pose of faces images. For instance, Many of “non-face” and “face” images import into the system. Machine learning techniques are employed to train the system based on the statistical properties and probability distribution function. Principle Component Analysis (PCA), Support Vector Machine (SVM), Naïve Bayes Classifier, Hidden Markov model, Neural Network and Adaboost are well-known classifiers to use for face detection.

   Pros: fast to detect face. Can be detected different pose and orientation if have enough training set. Showed a good empirical results.
   Cons: need more and more “non-face” and “face” sample for training, need to scan different scale.

Global and component methods

Above 4 methods can be grouped into two categories: global and component.[17] In global method, every feature vector that represents a whole face image. A Major problem in global method is variation in the appearance of object. However, in component method, faces are extracted and become facial components and combine them into single feature vector which is classified by classifier. These setting will be increased the processing time and resource.

Example of global approaches face detection



Example of component-based approaches face detection.

Major Challenges

Although many of the algorithms have been proposed and improved within a few years. Face detection are still limited by some uncollected factors. E.g. variant pose, illumination condition, expression, hairstyle and different camera equipment. Either one of the problems can decrease the performance of face detection. At the worst case, it should be the combination of the above problem. However, this situation often happens in our real worlds. E.g. surveillance with low quality.

In addition, occlusion is another of factor to effect the result of the FD rate. E.g make-up, sunglasses. In other words, occlusion sometime will be greatly changed the appearance of faces.

The following items are the summary of the main challenge in face detection:

1. Variant pose: Variant pose is occurred because of peoples not always orient to camera.
2. Illumination condition: Different lighting and the quality of camera directly affect the quality of face. Sometimes it can be varied greater than facial expression and occlusion.
3. Facial expression: Different expression in the face is presented different information to the machine. Face is non-rigid objects which is changed by different expression.
4. Occlusion: Face detection not only deals with different faces, however, it need deal with any optional object. E.g. Hairstyle, sunglasses are the example of occlusion in face detection. For global feature, occlusion is one of major difficulty factor in face detection.
5. Uncontrolled background: Face Detection systems are not only detected faces on uniform environment. In reality, Peoples are always located on complex background with different texture and object. These ‘thing’ are the major factors to effect the performance of face detection system.

The following photos are the example of uncollected factors.

1. Different pose of the same person

2. Same person with different illumination

3. Different facial expressions.

4. Example of occlusions by object.

Face Database

In recent decade, most of the researches are focused on the face detection. Many of the algorithms are proposed into different method. We need to use standard test data set for researches to compare the results. The following is the description of the face databases which are widely used into face detection community.

1. FERET, http://www.itl.nist.gov/iad/humanid/feret/, it includes 1000 people with male and female faces. It was created by FERET program start from 1993 to 1997. Each face is a single person with expression. This database has 14051 images with different views. The time-range of each people was taken at least one year or more. As a result, it has robust information for testing face-related algorithms over time.[18]

Limitation: Only a limited pose of each one. No information about the captured parameter

2. CMU Face Database, http://vasc.ri.cmu.edu//idb/html/face/frontal_images/index.html, it is a test set and composed of 4 set of testing images. Each test set has faces and non-faces picture. It has 511 frontal faces with 130 images. Images are grayscale with face sizes are varied.[19]

3. BioID, http://www.bioid.com, it has 1521 gray level images with images are 384x286 pixel. [20]. 23 peoples are involved.

4. CAS-PEAL, http://www.jdl.ac.cn/peal/home.htm, it contains 99594 images with varying pose, expression and illumination. 595 males and 445 females are in this database. [21]

5. Yale Face Database, http://cvc.yale.edu/projects/yalefaces/yalefaces.html, It has 15 peoples with 11 gray level images per each one with glasses, happy, center-light, normal, right-light, sad, sleepy, surprised and wink. Each image is 320x243 .[22]

Limitation: limited number of people. Light parameters are not mentioned. Every faces are frontal and no pose angle.

5. XM2VTS, http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/, it has 295 subjects with different pose which record 4 times with spread 4 months. Database contains 1180 color frontal images. Each image is 720x576 . [23]

Limitation: No additional information about the captured parameters.

6. Purdue AR, http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html, it has 4000 color images consist of 126 people's face. Images are included in variation of facial expressions, illumination conditions and occlusions.

Limitation: No information about the illumination condition of the images.

7. Yale Face Database B, http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html, it has 5760 images with 10 person. Each person have been taken 576 different conditions (9poses with 64 illumination conditions.) [24]

Limitation: limited number of group.

8. AT&T (Olivettti) Database (ORL), http://www.uk.research.att.com, it has 40 peoples with 10 images per person. Each image is 92x112.

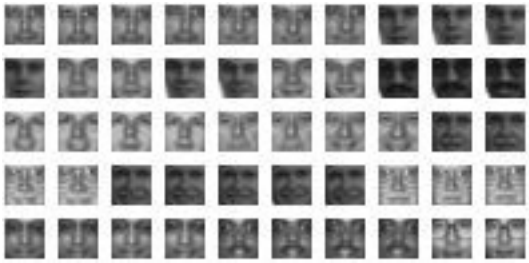
Limitation: limited number of people. Illumination conditions are inconsistent.

9. PIE Database, http://www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261, it has 41368 images included 68 peoples. Each person has 13 different poses, 43 different illumination conditions, and 4 different expressions. [25] Each image is 640x486.

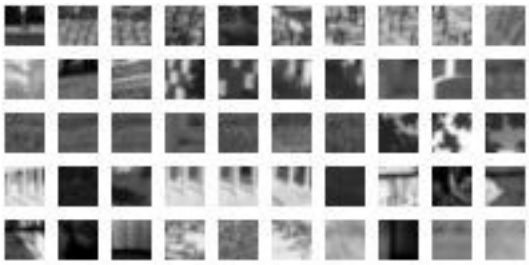
Limitation: Pose angle in the each images is not mentioned. Complex background of each images.

10. Japanese Female Facial Expression (JAFFE) Database, http://www.kasrl.org/jaffe.html, it has 213 images with 7 facial expressions posed by 10 Japanese female models. Each model is contributed 6 emotional expressions. Each image is 256x256. [26]

11. MIT Face Database, http://cbcl.mit.edu/software-datasets/FaceData2.html, it has two types of images, one is training set and other is test set. They are included 2429 faces and 4548 non-face in training set and 472 faces and 23573 non-faces in test set respectively. They are all 19x19 size. Here is the faces sample:



Here is the non-face sample:



Commercial Software

In the last decade, numerous of software related to face is developed. The following are the list of commercial software: [27]

1. Cognitec AG – FaceVACS , http://www.cognitec-systems.de/index.html
2. Identix (LFA) – FaceIt, http://www.l1id.com/pages/17
3. Neven Vision, http://www.nevenvision.com/
4. Vissage, http://www.viisage.com/
5. FaceFINDER, http://www.facefinder.com/
6. FaceSnap, http://www.crossmatch.com/FaceSnapFotoshot.html
7. TrueFace, http://www.miros.com/
8. SpotIt, http://spotit.itc.it/
9. BioID, http://www.bioid.com/

Performance Evaluation

The result of evaluation of face detection system is varied from different face database. In order to get better evaluation of face detection, regular basis should be applied. The result of their analysis will played a certain role for future work.

For face detection, two parameters are introduced to evaluate the performance:

1. Detection rate: It is the rate to find face in the given images/ video.
2. False positive rate: It is the rate of non-face objects are reported as positive.[28]

Here is the equation of false positive rate.

$$\text{false positive rate} = \frac{\text{number of false positives}}{\text{total number of actual negative instances}}$$

If we want to achieve a high detection rate, then the false positive rate will tend to be higher. In the other word, if you want to achieve low false positive rate, then the detection rate would be under the expectation.

In perfect algorithm of face detection, it must be 100% of detection rate and 0% false positive rate. However, at this stage only the human brain can do this.

Current status of overseas research

The face detection algorithm is well classified with different methods. These algorithms are used base on which condition are employed. Well-know university, research institutes, enterprise institute have established a research group associated with face related project. They are CMU, MIT, Michigan State University, UCLA… and so on.

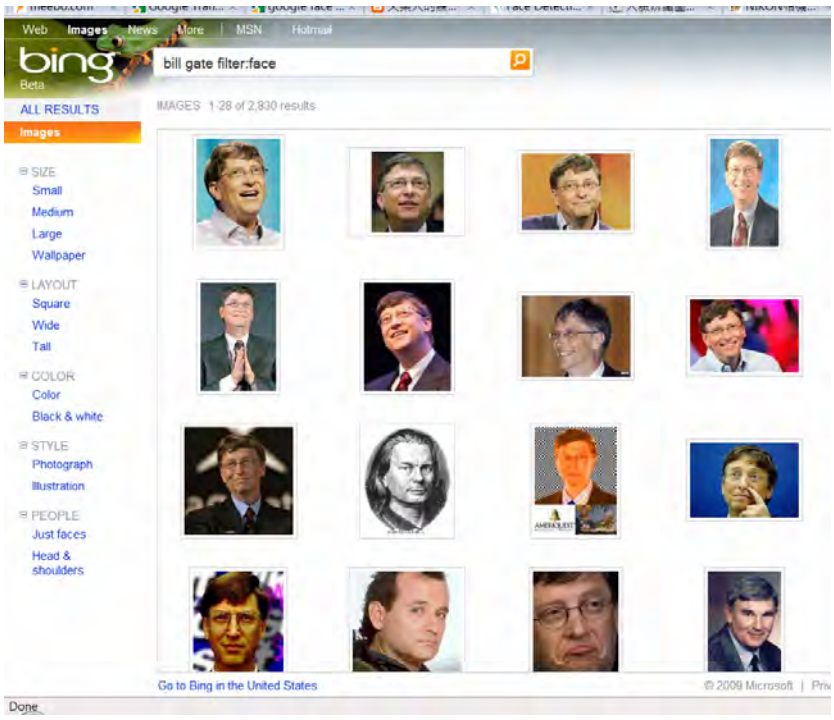
In the good controlled conditions, the detection rate can be achieved at least 95 % from the case about 1000 people. However, in the environment is relatively poor circumstances, the detection rate just got 80% or below. [29]

Example of Face Detection

Nowadays, the technology of face detection is widely applied within different field of business, Photo-taking, search engine, biometrics and so on. These technologies become popularity because of increase human interest in life, as a well-know mobile phone manufacturers said: connecting people.

The following is an example of face detection:

1. Search engine – Microsoft:
   The search result of ‘bill gate’ is shown below.



2. Camera Manufacturer – Nikon:

   Pictures from advisement of Nikon are shown below.[30]

Chapter Summary

This chapter is presented about the introduction of face detection, face database, the major challenges of face detection. In addition, four face detection methods are introduced to give general ideal to user about the approaches of research area. Also, the performance evaluation is addressed to show how is the successful detection system should be achieved.

3. Methodology for Project

Introduction

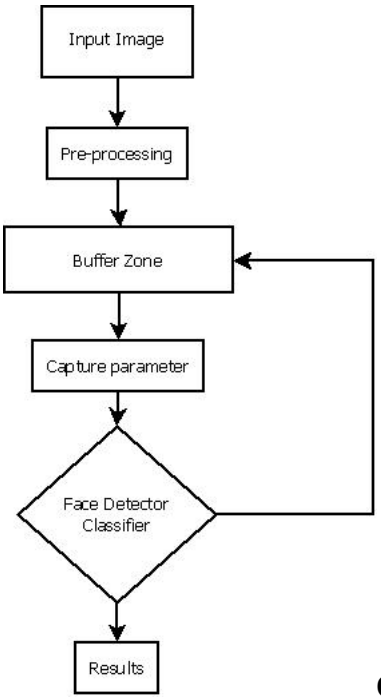
Each input image is processed with resizing and converting to grey color for further detection. When given image input to system, it will be experienced with numerous of classifier (cascade). “1” will be marked by classifier if it’s has face otherwise will ‘0’.

The face detector was proposed by Viola P. and Jones M.[31]. Adaboost used as a classifier which is trained with a few hundreds sample images. It’s included with positive sample (faces) and negative sample (non-faces). Each trained samples were resize and scaled into same size (20x20).[32]

In order to search the face efficient, the given image will be “resized” certain ratio to make sure faces in the whole images are searched every location and size. In other words, each given image should be dong several times. Details will be discussed later in this section.

Face detection in this project can be regarded as two parts and illustrated as follows:

1. Pre-processing: This stage includes images processing to enhance the quality of input images.
2. Face detector stage: The core section in face detection includes Adaboost filter.



Overview of FDS

Pre-processing

System input is color images which included images of human faces or not, output is the human faces which is extracted from original images. In order to get the better result of detection, pre-processing is essential. In this section, pre-processing is addressed with giving detail description.

Grayscale conversion

For getting to reduce the information of images, image should be done a converting to grayscale. Each color images (RGB images) are composed of 3 channels to present red, green and blue components in RGB space. Below is the example to giving the general ideal of the RGB color image.

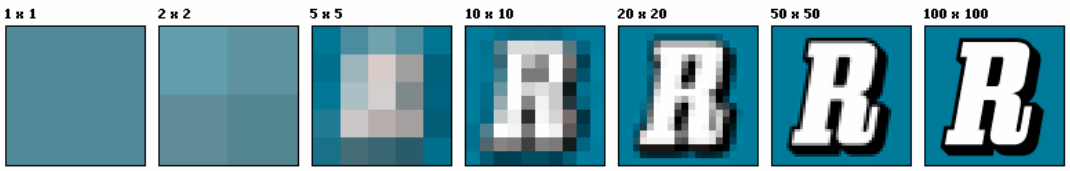
Image is defined with grayscale level, which means pixels in images are stored 8-bit integer to represent color from black to white[33]. In addition, grayscale images are sufficient enough to process face detection.

Convert to Grayscale Image algorithms

1. Given example images (R_1,G_1,B_1),….(R_n,G_n,B_n) where R, G, B are the value of red, green and blue respectively and ‘n’ is total number of pixel in given image.
2. The new grayscale images has pixel from G_1,…..G_n, where using formula is as follows: 0.21R + 0.71G + 0.07 B = G . Unlike averages method, this form is considering the ratio because of human perception.

Image resizing

Images are synthesized by numerous of pixel which is the small unit in the image. Also, images are the 2-dimensional matrix pattern, every pixel in the images is represented something information. For example, ‘0’ is white and ‘255’ is black in gray scale images. Because there are a lot of information to deal with, input images are employed the resizing processing to reduce the images resolution with keeping same quantity. Below example is an illustration about the different resolution to describe the same image.

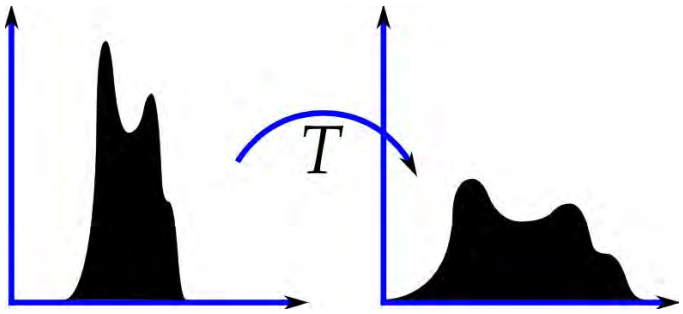


The top-left side of each image is the resolution of each one. Left-side’s image is the original.

Image has 3000 pixels in width and 2000 pixels in height which means it has 3000 x 2000 = 6,000,000 pixels or 6 megapixels. If the image has been resized into 1000 pixels in width and 600 pixels in height, image only has 0.6 megapixels. At least system only use 1/10 timing to handle it. However, if you had to adjust the size of the image will also affect the face detection rate.

Histogram Equalization

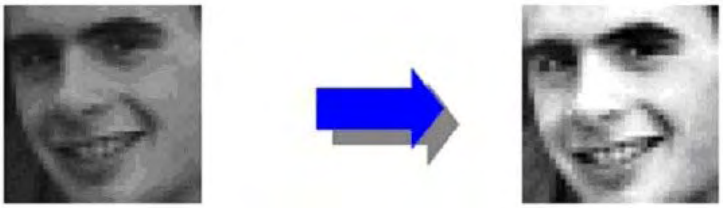
Histogram equalization is a statistical method of images processing. It works as a statistical histogram of color distribution of the average scattered in the histogram, so that the distribution of a histogram graph homogenization. The ideal is present as a follows:



The change of the histogram after perform histogram equalization

In the above chart, the shape of graph has been widened which is the meaning of average scattered in the histogram.

This method usually increases the contrast of the input images.[34] In face detection system, The left-hand side of below images is resized grayscale images. Other is output images after proceed the processing of histogram equalization. You will see very significant results.



Example of the process of histogram equalization

Algorithms of Histogram equalization:

1. Grayscale images has X_n pixels with i represent a value of gray level in each pixel. The following chart is represent the relationship between probability of occurrence and the value of each pixel:

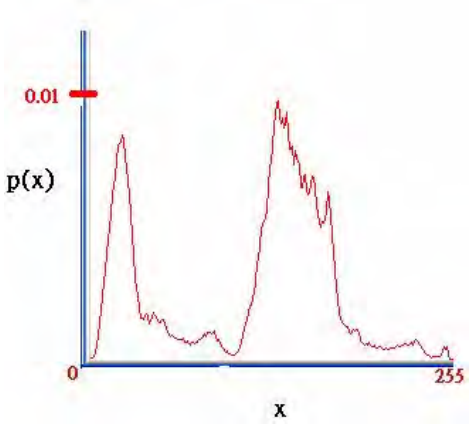


Chart of Probability Density Function (PDF)
And

$$\sum_{i=0}^{255} p(x_i) = 1$$

Px is being historgram of images and normalized to [0,1]

2. Let us define the cumulative distribution function as follows:

$$F(n) = \sum_{i=0}^{n} p(x_i) \qquad n = 0,1,2...255$$

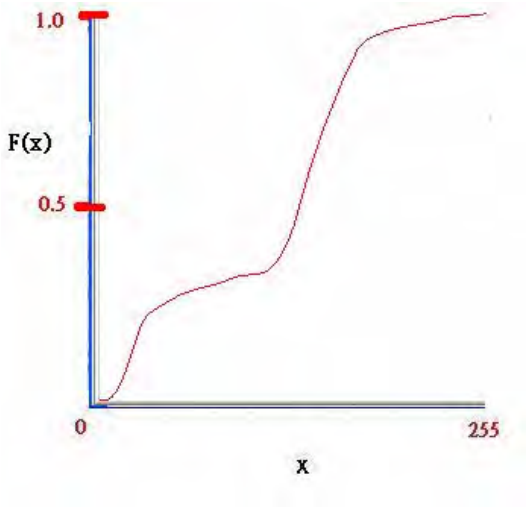


Chart of Cumulative distribution function

3. Minimum and maximum value are found and applied into following equation to find out the histogram equalization of each pixel:

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

Where cdf_{min} is the minimum value of CDF, M is the width of image and N is the height of image. L represent a large value of grey level, = 256.

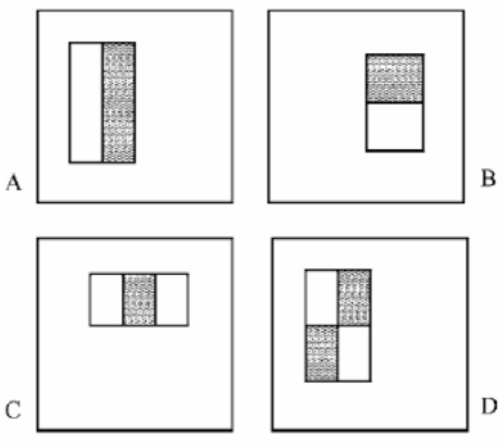
Algorithms of Face Detection

This system is employed Viola and Jone's algorithms[35] to do quickly face detection. These algorithms are consists of 3 methods, they are 'integral image', 'AdaBoost' and 'Cascade'. In this section, we discuss them more details.

Integral Image

What is Integral image? Rectangular 2-D image features can be computed quickly using an intermediate representation called the integral image.[36]

The following figure is the 4 different type of "Rectangle Feature" which is not fixed area size. The rectangle features like a filter and scan the images to collect the value of Integral. The Integral value is calculated from the different between white and gray area of Integral. "A" and "B" are the 2-rectangle features. "C" is the 3-rectangle feature and "D" is 4-rectangle feature.



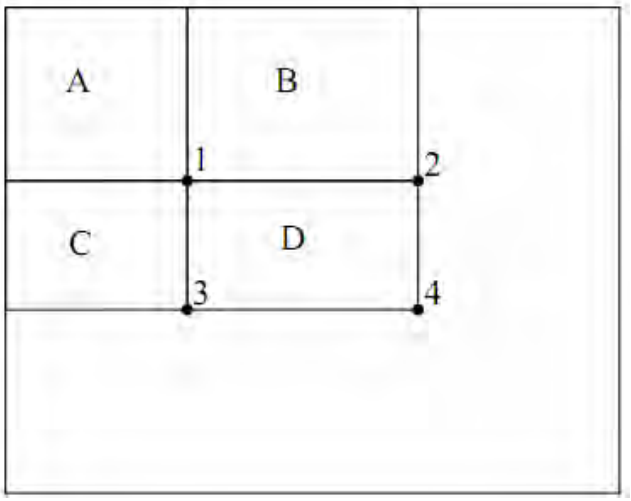
The rectangle feature are placed the image random size and location, one feature is defined as rectangle feature is placed on the image one time with specific size and specific location. For the detector with 24x24 resolution, the set of rectangle features is over 180,000.

Table, the relationship of size of windows and number and features

Size of windows	**Number of features**
36 x 36	819,264
30 x 30	394,725
24 x 24	162,336
20 x 20	78,460
16 x 16	32,384

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

For the point 1 (x,y) below, Integral Image are defined as above. *i*(x,y) is the original image and *ii*(x,y) is Integral Image. For the grayscale images, values in pixels is the range from 0 ~255. However, color images are strongly recommended to convert into grayscale.



The value of the integral image at 3 is the sum of the area in pixels of A and C. The sum of pixels in area 'D' is : $ii_4 + ii_1 - (ii_2 + ii_3)$

ii_1 = Sum of Pixels in Area 'A'
ii_2 = Sum of Pixels in Area 'A' + Sum of Pixels in Area 'B'
ii_3 = Sum of Pixels in Area 'A' + Sum of Pixels in Area 'C'
ii_4 = Sum of Pixels in Area 'A' + Sum of Pixels in Area 'B' + Sum of Pixels in Area 'C' + Sum of Pixels in Area 'D'

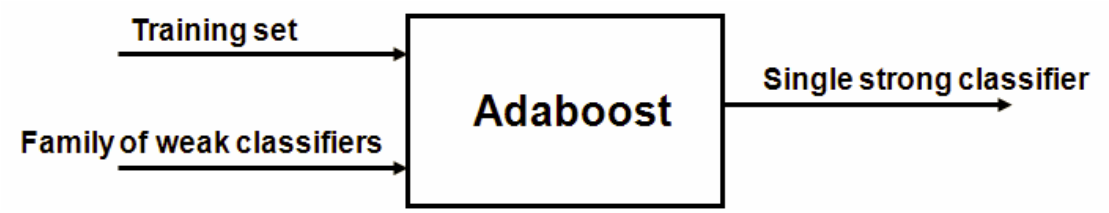
as a result:

The sum of pixels in Area 'D' = $ii_4 + ii_1 - (ii_2 + ii_3)$

Thus, the calculation of rectangle feature is only related to the point of rectangle feature of integral image, instead of the coordination of the point. System only use constant time to calculate the integral image what matter the size of the rectangle feature. This method has been increased greatly to detection speed.

AdaBoost and Cascade

AdaBoost is a short form of **Ada**ptive **Boost**ing. AdaBoost is machine learning algorithms and is proposed by Yoav Freund and Robert Schapire on 1995. [37] It is to pick up a few thousand features and assign weights to each one based on a set of training images. The aim of AdaBoost is assign each weak classifier with best combining weights. More results were proved that it works in generalization performance.[38] It consists of numerous of weak classifier to construct a strong classifier. Weak classifier just handle a slightly features, however, numerous of weak classifier can be used to increase overall performance. One strong classifier can't be computed a large input features in real time.



Basic idea of AdaBoost

The weak classifier selects and rejects single rectangle feature from the optimal threshold classification function. This process can be presented as follows, $h_j(x)$ is weak classifier which is computed by feature f_j, threshold θ_j and parity p_j.

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

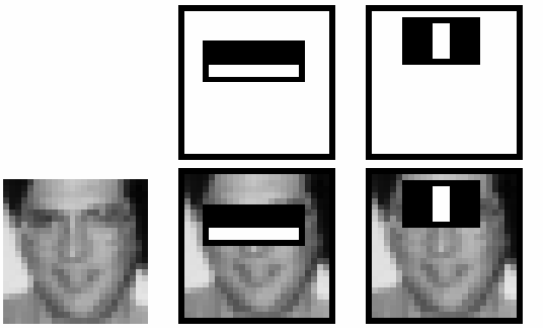
The threshold of each weak classifier can be changed, as a result the false negative rate is trending to zero. The key of this algorithm is simple in individual and achieve height detection rate in overall performance.

The following is the procedure of Adaboost:

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon t}$.
- 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 $\alpha_t = \log \frac{1}{\beta_t}$

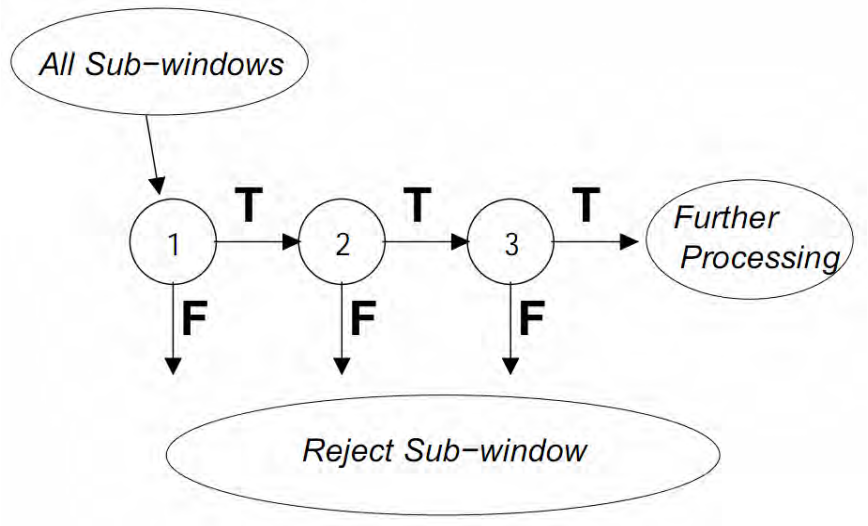
AdaBoost algorithm [39]

The following example shows AdaBoost how to select and measure the integral images in the face's images. First, two rectangle features are selected. They measure the integral images between eyes and upper cheeks.



The overall of performance of detection is a decision tree from one to one which is called "Cascade" (see below figure). First Classifier will reject most of the features and positive result transfer to next classifier and so on. A negative result can be rejected immediately to "Reject Sub-window". At the last stage of classifier, it just handles few sub-windows. After a few round of the classification, the quantity of detection rate can be increased.

For more information, the former classifier only has low detection resolution, and filter out majority part of the non-face features. The rests are handled by subsequence classifier one by one, until the last of the classifier is completed. Finally left behind will be what we want pictures of human face.



In addition, the factors to be optimized (highly detection rate) are increase the number of classifier stages, number of features in each stage and the threshold value of each stage.

Why AdaBoost is used in this project?

AdaBoost algorithm is core methods in this project, the following are the advantages:

1. **Simple to implementation**: As you know, AdaBoost is a machine learning algorithm which only input two set of data. One is training dataset, other is a sequence of weak classifier. System is no need to concern about the facial structure.
2. **Adaptive algorithm**: AdaBoost is a short name of adaptive boosting, which means each of weak classifier will be adjusted their own weight when import positive and negative samples in learning stage of each iteration.
3. **Theoretical training error index can tend to 0**: Freund and Schapire in [40] are proposed, given a numerous of positive and negative samples, the training error can reach to 0 value in numerous of iterations.
4. **Fast Detection methods**: In other research, AdaBoost got 15 frames per second of face detection with the size of resolution is 384x288 pixels which ran on PIII 700MHz computer. [41]

How to train the classifier

This section describes how to train the classifier for using the ideal of rapid object detection. OpenCV comes with a file which is trained for face detection. However, it didn't explain more detail about the training file. We need to train the classifier with numerous of positive and negative sample.

As a spoken before, AdaBoost is a machine learning algorithm. It should be trained and became a strong classifier. Training data is prepared and divided into two parts. One is positive training sets and other is negative training sets. Positive training set is a set of face images cropped with fixed size. While, negative training set is a sequence of non-face images.

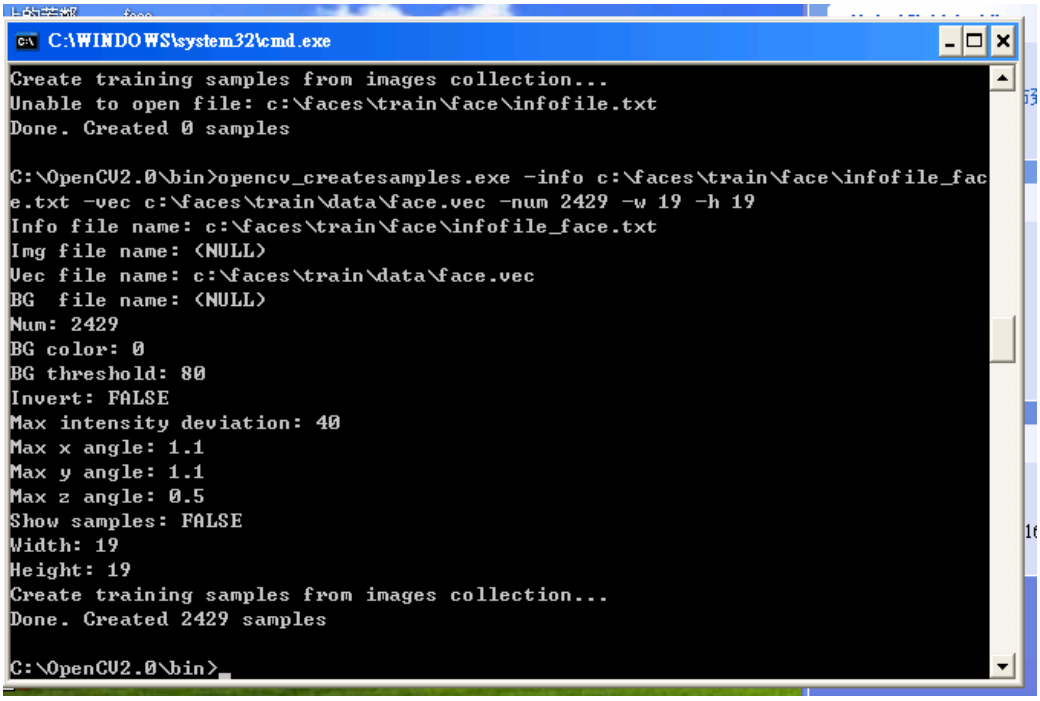
Face Database used for training

MIT Face Database is used for training, it is because every single image in this database are resized, normalized. Also, it has more training set (6977) which fit the requirement of training classifier.

Preparation Sample

In order to get highly detection rate in the images, training the system using positive and negative sample are necessary. Positive samples in this case are face images, negative samples are non-face images. Sample files must be resized into same size and grayscale mode to make sure no any variation is occurred.

"opencv_createsamples" utility is used to create vector file of positive sample for "haar-training".



2429 face images is inserted into training set and ‘face.vec’ file is generated

Training

It’s used “opencv_haartraining” utility to perform the training step. The following is the output of “opencv_haartraining” utility:

N	%SMP	F	ST.THR	HR	FA	EXP. ERR
1	100%	-	-0.731273	1.000000	1.000000	0.339500
2	100%	+	-1.503060	1.000000	1.000000	0.259000
3	93%	-	-1.089103	0.998000	0.982000	0.295500
4	99%	+	-1.810714	0.999500	1.000000	0.216000
5	84%	-	-1.532054	0.996000	0.901500	0.197250
6	61%	+	-2.252650	0.998000	0.883000	0.208000
7	66%	-	-1.730624	0.995500	0.776500	0.198000
8	73%	+	-2.034414	0.996500	0.803000	0.197750
9	68%	-	-1.834773	0.995500	0.804000	0.097750
10	69%	+	-2.076961	0.996000	0.812500	0.095250
11	67%	-	-1.754567	0.995500	0.679000	0.086500

N = Number of iteration of feature selection training

% SMP = % of original sample left

F = + indicates the feature is flipped.

ST. THR = stage threshold

HR = Hit rate

FA = False alarm rate

EXP. ERR = Expected (misclassification) error.

This program create xml file after the training is finished. These file is the one of the important file to perform face detection.

Chapter Summary

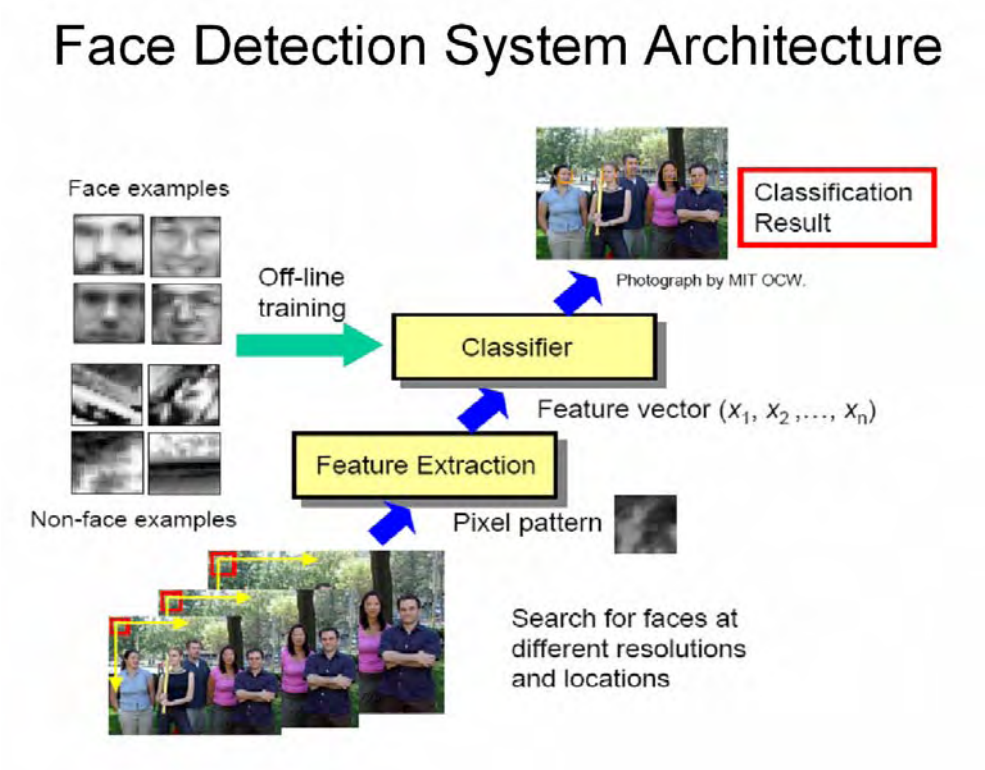
This section present the methodology used in the project. They are included two parts. The first part is pre-processing section, which can be divided into “Grayscale Conversion”, “Image Resizing” and “Histogram Equalization”. Second part is algorithm section, which are “Integral Image”, “AdaBoost” and “Cascade”.

4. Analysis and Design - Face Detection System

This section gives a list of system requirement and more detail information on the application of this project. It is named as FDS (Face Detection System).

Overview

The core ideal of the Face Detection System (FDS) is illustrated as below picture:



Functional Requirements

The following list is a basic requirement of the FDS:

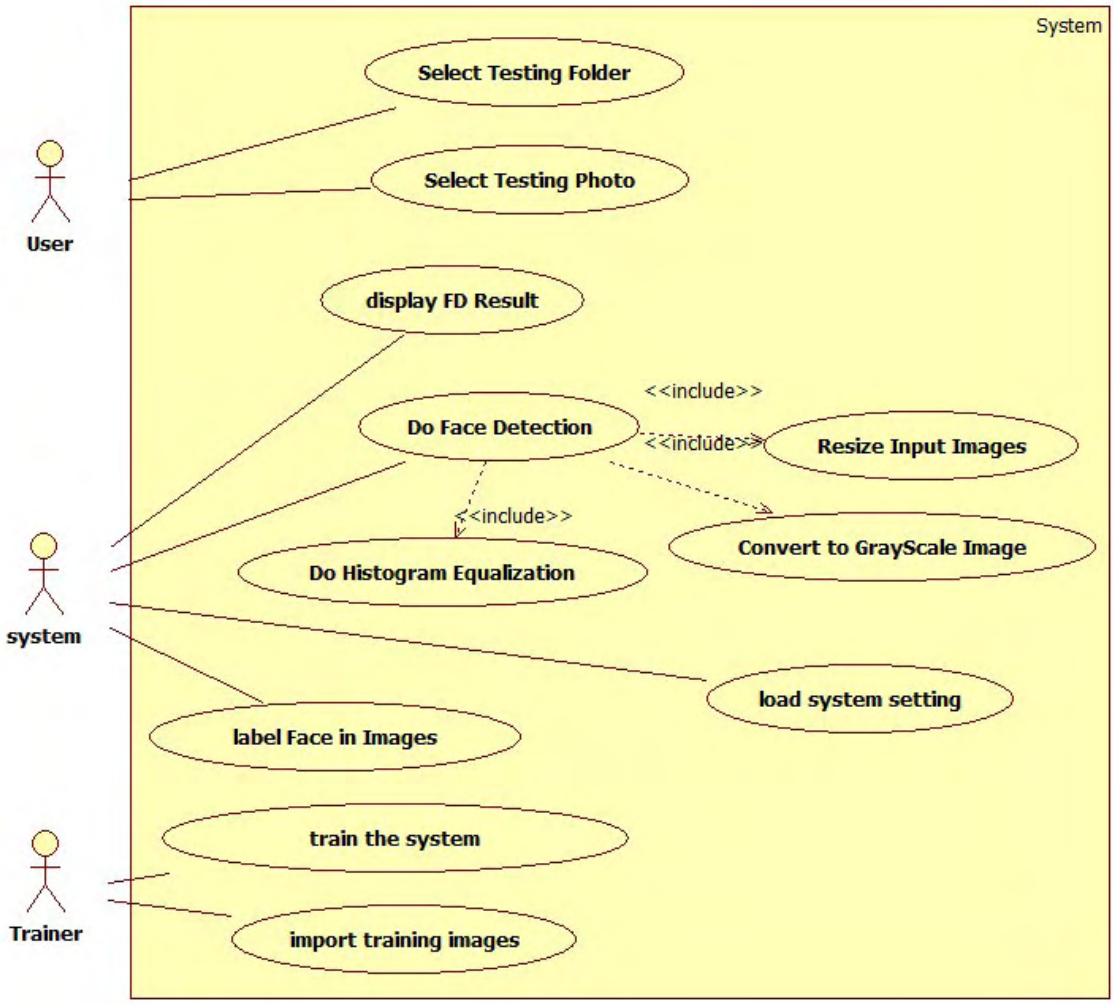
1. Support Face Detection
2. Label each detected face in the Picture
3. Support select images from other folder
4. Show clear images list to user
5. Display the number of ‘detected’ faces
6. cross-platform

System Structure – UML

The following sections are the list of diagrams which are use case, activity diagram, class diagram.

1. Use Case Descriptions

Figure is overview of use case in FDS. These use cases are described more detail in specific condition.



Case No.	1
Name	Import training images
Actor	Trainer
Description	FDS should be trained with a numerous of training Image. These training images are ‘face’ and ‘non-face’ images with fixed size. This use case is a method to import training images.
Pre-condition	1. Trained Images with fixed size are location same folder and waiting the trainer to execute this method
Post-condition	1. Selected Training Images are located in “training” folder
Remark	

Case No.	2
Name	Train the system
Actor	Trainer
Description	System should be learnt to know how to classify ‘face’ in input testing images.
Pre-condition	1. Trained Images are located in “training” folder
Post-condition	1. XML file is generated which is the parameter of the feature of ‘face’. This file will be loaded when the system is executed for face detection.
Remark	

Case No.	3
Name	Load system setting
Actor	System
Description	Configuration file is loaded before any further face detection
Pre-condition	1. ‘config.ini’ and “haarcascades” are located on the root folder of the program
Post-condition	1. System will be executed
Remark	

Case No.	4
Name	Select Testing folder
Actor	User
Description	User select folder which want to perform face detection
Pre-condition	1. User should make sure the “selected” folder has ‘.jpg’ format in it.
Post-condition	1. Pictures in “Selected” folder may be loaded on the right-hand side of list.
Remark	

Case No.	5
Name	Select Testing Photo
Actor	User
Description	User click the filename of picture in the right-hand side of list to do face detection
Pre-condition	1. List in UI should be displayed any filename
Post-condition	1. Main Windows will be displayed ‘resize’ photo on the top 2. Face in the selected images will be displayed one by one on the below of the main windows.
Remark	

Case No.	6
Name	Do Face Detection
Actor	System
Description	This is a main function of the system to detect face in the Picture.
Pre-condition	1. Testing image should be resize, convert into grayscale images 2. Doing histogram equalization
Post-condition	1. Execute “Display FD Result”
Remark	

Case No.	7
Name	Resize Input Images
Actor	System
Description	Testing images may be resized to speed up the face detection process.
Pre-condition	1. Original Testing Images is selected 2. if the width of testing image is over 1000 pixels, it will continuous to resize
Post-condition	1. Resized Picture will be generated.
Remark	

Case No.	8
Name	Convert to GrayScale Image
Actor	System
Description	Testing images will be converted into grayscale image to reduce the information of input color picture.
Pre-condition	1. Only accept the picture which is below 1000 pixel of width 2. Input images must be color-images.
Post-condition	1. Grayscale images will be generated.
Remark	

Case No.	9
Name	Do Histogram Equalization
Actor	System
Description	Histogram Equalization is applied into pre-processing section to enhance the contrast of picture
Pre-condition	1. Testing image must be converted into grayscale and resized images.
Post-condition	1. More contrast’s testing image is generated.
Remark	

Case No.	10
Name	Label Face in Images
Actor	System
Description	FDS will be labeled the face with rectangle in the testing images.
Pre-condition	1. Label option should be checked
Post-condition	1. Testing images will be displayed with red rectangle for each detected face.
Remark	

Case No.	11
Name	Display FD Result
Actor	System
Description	FDS will be displayed “detected” face.
Pre-condition	1. Testing image should be selected on the right-hand side of list. 2. Testing image should be resized and converted into grayscale 3. Testing image should be done the processing of histogram equalization.
Post-condition	1. FDS will displayed the result of detection (Human face)
Remark	

2. Sequence Diagram

In this section, it will shown how the step of the system to perform specific task.

Sequence Diagram – Face detection

The following diagram is face detection's sequence diagram which is core section in the system.

1. The class of FileHandle is prepared the list of the filenames that are selected by User and displayed in the User interface.
2. User select specific "image" to perform face detection
3. FDS call and create new object from FD Modules.
4. Input image is being to experience resize processing.
5. Input image is being to convert greyscale mode.
6. Input image is being to perform histogram equalization.
7. Done Message will be sent out to FD Modules.
8. Pre-processing is finished and performs face detection.
9. The result of face detection is shown in FDS's User Interface.

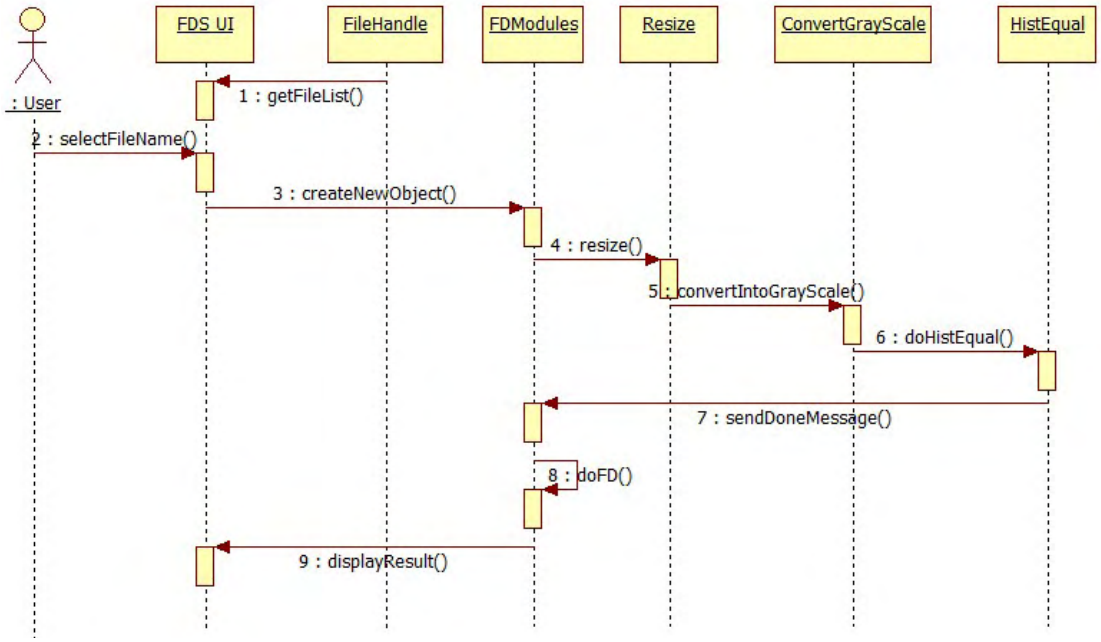


Figure , sequence diagram of face detection

Sequence Diagram – Preparation Sample

The following diagram is sequence diagram to show how to prepare the sample.

1. Trainer select sample folder
2. TrainTools call “opencv_createsample” and execute itself.
3. output screen is shown and acknowledge trainer.

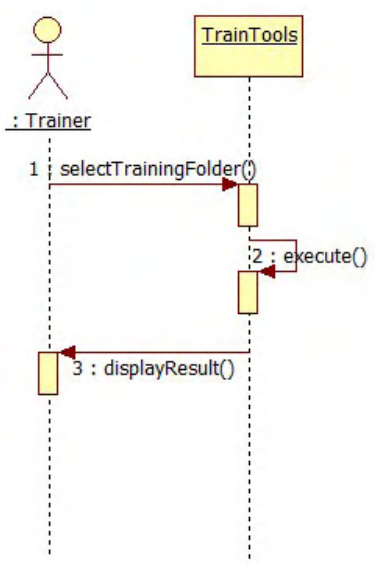
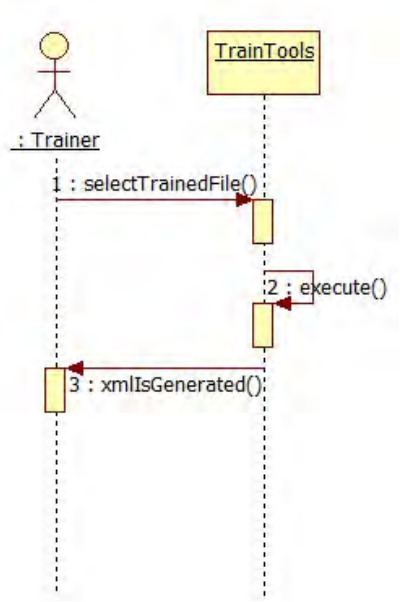


Figure , sequence diagram of preparation sample.

Sequence Diagram – Training Sample

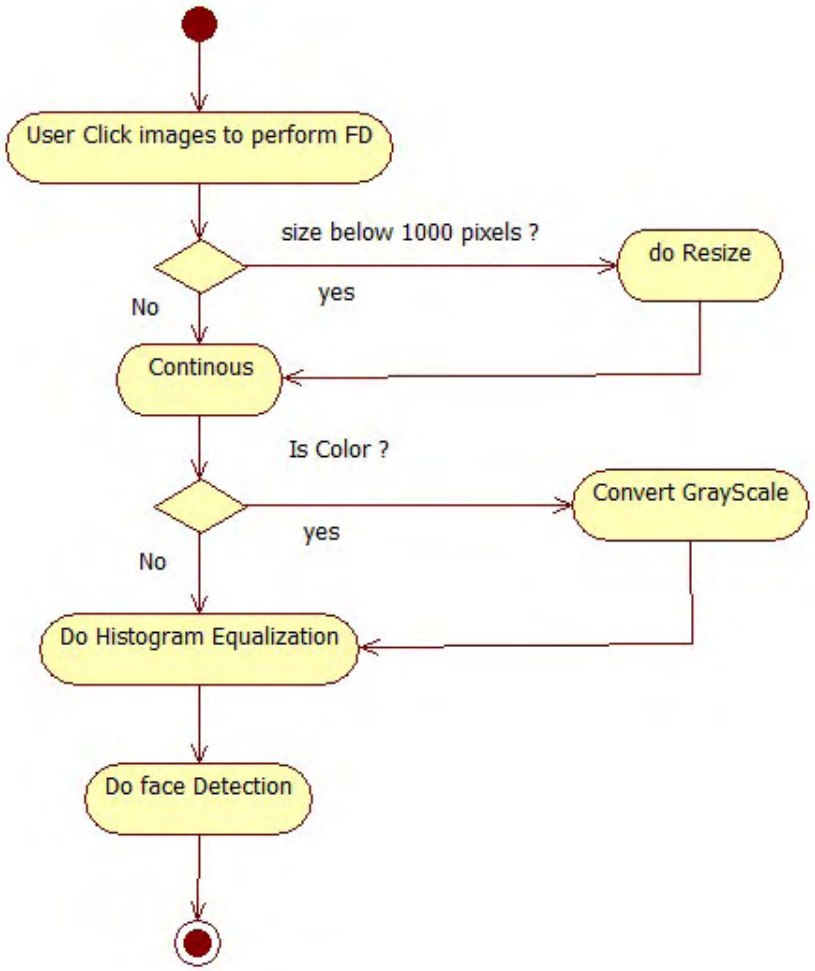
After the step of preparation sample, the following diagram is illustrated how to train the sample step by step.

1. Trainer select vector file which is generated from before.
2. Trainer activate “opencv_haartraining” utility to perform training the sample.
3. XML file is generated when training is finished.



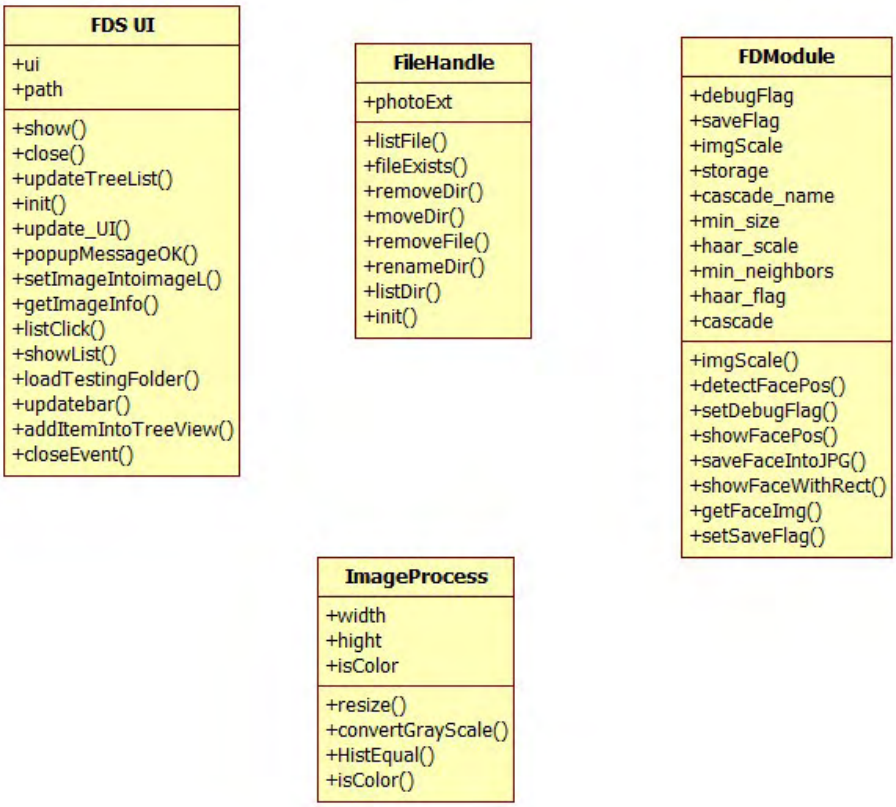
3. Activity Diagram

The following diagram is workflows of face detection to show system's operation.



4. Class Diagram

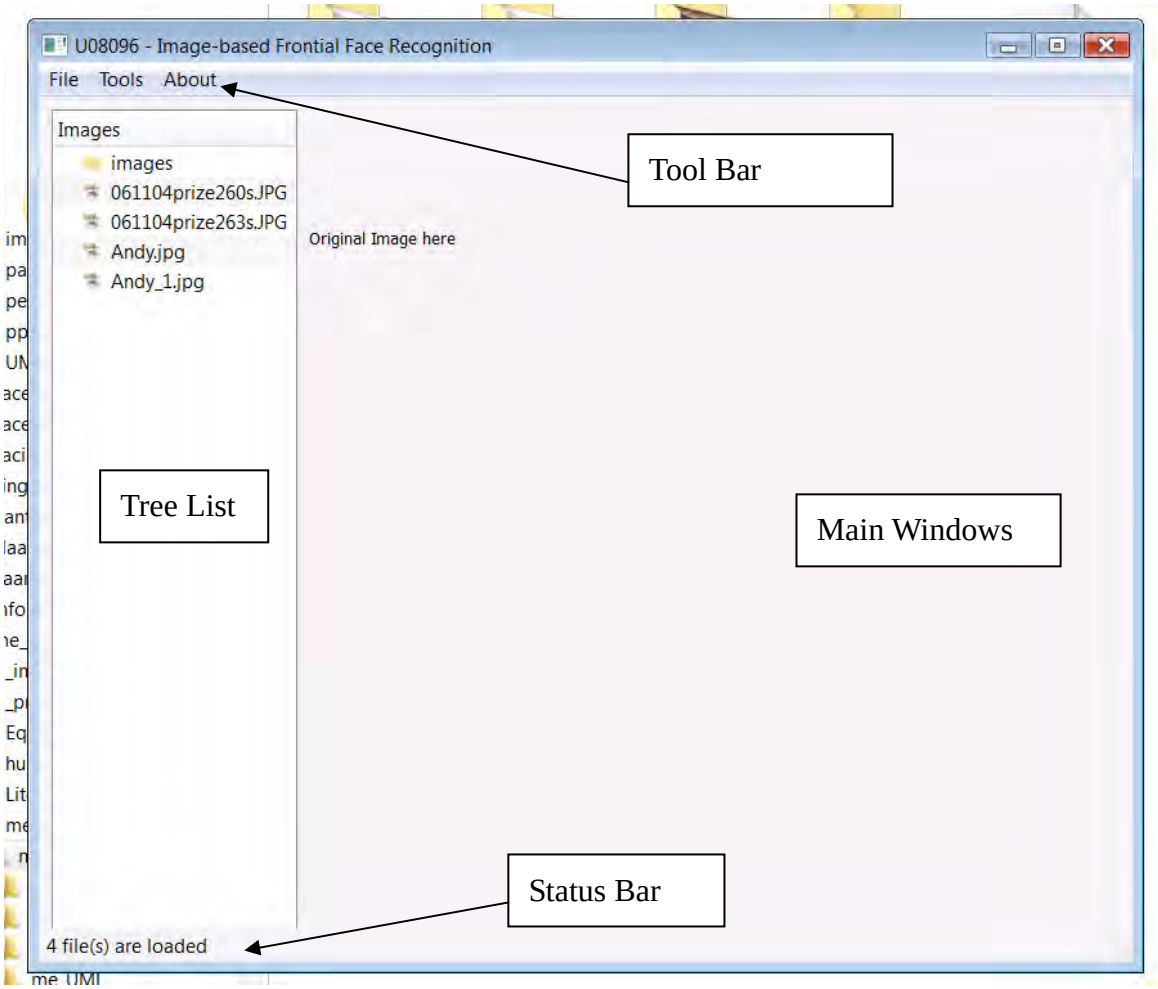
The following figure is shown the main class of FDS. The “FDS UI” is the user interface class mainly handle any GUI related work. For example, update the UI widget. The “FileHandle” class is handled any file operation. For example, check the file is exists or not. The “ImageProcess” class is the image processing tools. For example, it handles resizing, convert into greyscale. The “FD Module” class perform face detection.



User Interface

The FDS only has single user interface. Four sections are defined in this program:

1. Tree List section is a list to show filename here.
2. Tool Bar section
3. Status Bar is shown any system information to user. E.g. how many files are loaded and selected.
4. Main Windows is shown the result of face detection.



Chapter Summary

This section presents UML diagram to give the ideal about the analysis and design approaches of the project.

5. Testing and Implementation

Overview

This section is pointed out the test case and test documentation. The aim of this section is to make sure ‘FDS’ to achieve the objective without any error. The details information is shown below.

The main testing is defined the ability of face detection. Testing will be focused on pose, illuminated condition, occlusion by object. These testing will be tested with different face database to get the face detection rate of the system.

Testing Objective

1. Make Sure 100% correct code.
2. Meet the functional requirements of the FDS

Tools in Testing

Hardware Configuration:
- CPU: Intel 3.00GHz
- Ram: 4.00GB

Software Configuration:
- Windows Vista with Service Pack 1
- Python 2.5.4.3
- PyQt v 4.4.3
- OpenCV 1.0

Face Databases:

The following face databases are used for testing a group of pictures:

1. BioID is a test set A, it has larger vary of illumination, complex of background and different face size of each picture. The condition of environment in this set like an indoor condition. In addition, each picture is involved one person. At least, 1521 pictures are involved in this database.

2. MIT-CMU is a test set B, it has larger number of person (2000 samples) with different pose.

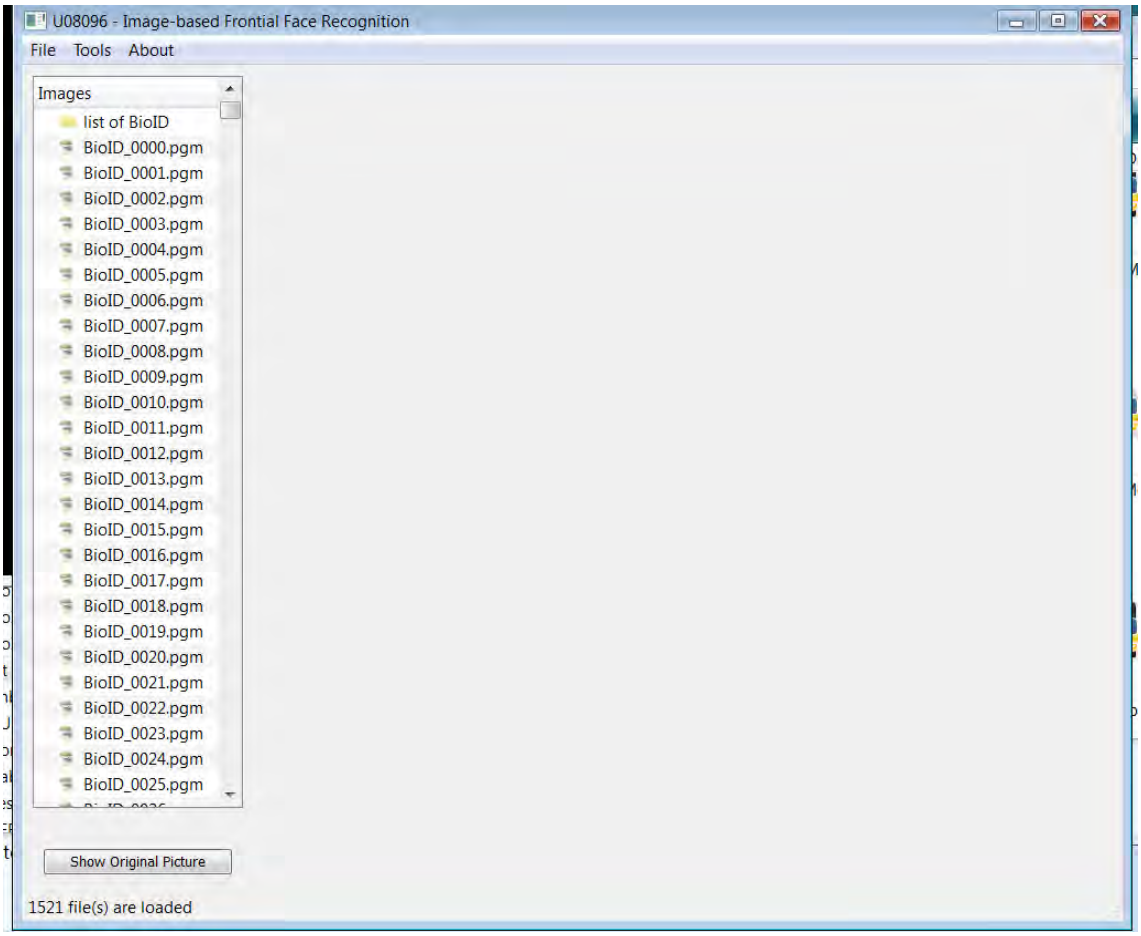
3. CMU Face Database is a test set C, it composed of 4 set of testing images, newtest, rotated, test and test-low. It has 511 frontal faces with 130 images in newtest, test and test-low folder. Some images have not any faces for testing false positive rate. Below images is the sample of newtest folder and it used in test case.

Test Case Summary

Here is the summary of the testing:

Test Case ID	Test Description
001	Change Other folder for face detection
002	Conduct Face Detection with data set A (BioID)
003	Conduct Face Detection with data set B (MIT-CMU)
004	Conduct Face Detection with data set C (CMU)

Test Case

Test Case ID:	001
Test Description:	Change Other folder for face detection
Purpose:	
To make sure FDS can be worked different folder conveniently	
Input Data:	
The BioID Folder are selected (E:\me\informatics\U08096\Face Database\BioID\list of BioID)	
Except Result:	
After selecting BioID folder, 1521 pictures in folder are shown in the Tree List for user preview.	
Actual Result:	
	
Pass / Fail:	Pass
Remarks:	
N/A	

Test Case ID:	002
Test Description:	Conduct Face Detection with data set A (BioID)
Purpose:	
To find a detection rate of the FDS using a data set A of images.	
Input Data:	
1521 pictures in BioID (face database) are used.	
Except Result:	
Every picture in BioID is assumed to detect with single face and each face is labeled with red rectangle. Detection rate: 100%	
Actual Result:	
Detection rate: = 1460/1521 x 100 %= **95.99%** The results go to Appendix "**Result of Test Case 002**"	
Pass / Fail:	Pass
Remarks:	
N/A	

Test Case ID:	003
Test Description:	Conduct Face Detection with data set B (MIT-CMU)
Purpose:	
To find a detection rate of the FDS using a data set B of images.	
Input Data:	
2000 pictures in MIT-CMU (face database) are used.	
Except Result:	
Every picture in MIT-CMU is assumed to detect with single face and each one is labeled with red rectangle. Detection rate: 100%	
Actual Result:	
Detection rate: = 1763/2000 x 100% = **88.15%** The results go to Appendix "**Result of Test Case 003**"	
Pass / Fail:	Pass
Remarks:	
N/A	

Test Case ID:	004
Test Description:	Conduct Face Detection with data set C (CMU)
Purpose:	
To find a detection rate of FDS for uncontrolled condition.	
Input Data:	
Images in ‘newtest’ folder are used has 65 images with 180 faces.	
Except Result:	
Detection rate: 100%	
Actual Result:	
Detection rate: = 145/180 x 100% = **80.55%** The results go to Appendix “**Result of Test Case 004**”	
Pass / Fail:	Pass
Remarks:	
N/A	

Chapter Summary

This section is given more detail information for test case. Everything in test cases is under controlled condition to determine the function of system is worked correctly. Test cases are focused on detection rate to test the system with face database. The general face detection rate is over 80% with variation with different face size.

6. Conclusion

In this project, FDS is constructed and achieve highly detection rate at least over 80% for **Test Case 004** (CMU test set), which faces in images are uncontrolled environment.

In the fontal face Images testing, FDS got 95.99% of detection rate from **Test Case 002** (BioID), testers were all in indoor condition. That means FDS works very well with indoor activities or under controlled environment.

In the variation of pose testing, FDS got 88.15% of detection rate from **Test Case 003**. It means FDS can be handled controlled pose of human face.

Limitations

Although, FDS performs highly detection rate, achieved 80% detection rate for all test case, in this project. However, all the miss detected face can be concluded as follows:

1. illumination condition,
2. pose orientation,
3. facial expression and
4. occlusion

In **Test Case 002**, 61 images fail to hit as a face, it is because faces are blocked (occlusion). In other words, faces aren't shown fully, and face detector classified as a non-face object. There has 61 images are regarded as non-face objects. The following image is the example of regarding as non-face objects.

In **Test Case 003**, most of the faces are missed and reported as non-face images, it is because most of the face has unacceptable variation of pose. The following image is the example of “variation pose”.

In conclusion, this project presents a face detection system (FDS) using AdaBoost (machine learning algorithms). It got at least 80% of face detection rate. This method is suitable for the binary classification problem, whatever in any environment.

Also, the programming code of this project can be moved into other system platform because most of the programming codes are written by python, PyQt4 and OpenCV.

Future Work

At this time, researchers are still figure out the difficulty of face detection: vary illumination, occlusion on the face, pose variation. The algorithms of face detection are still improved by (1) to increase the face detection rate and (2) developing rapid object detection system in real time.

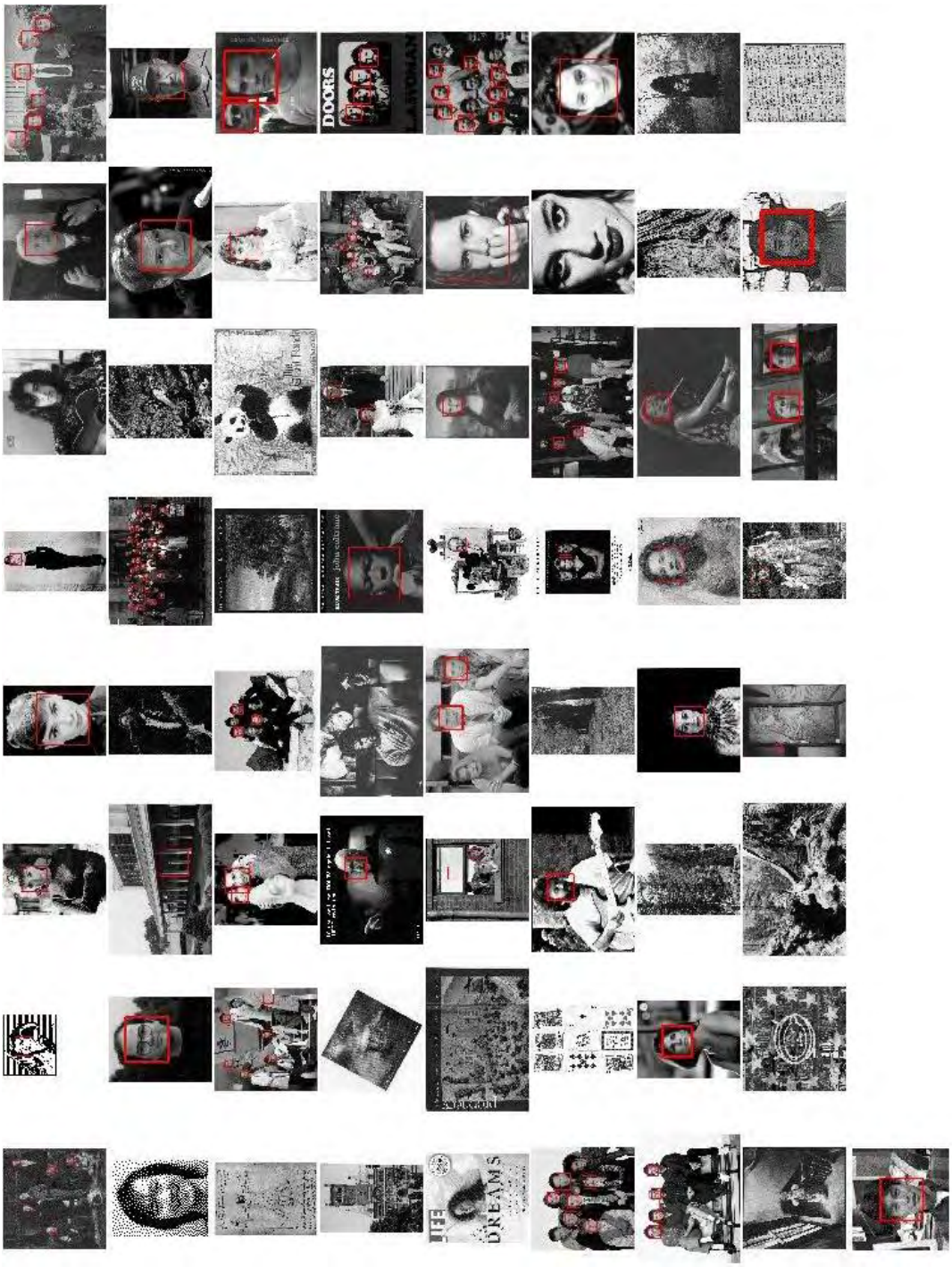
For the increasing the face detection rate, some techniques do a great job here. Eye detector or skin detector are the example of this improvement. New methods are added, as a result, it eliminated uncontrolled factors. For example, skin detector is applied in the step of pre-processing. System just considered a small part of area in the images.

Appendix

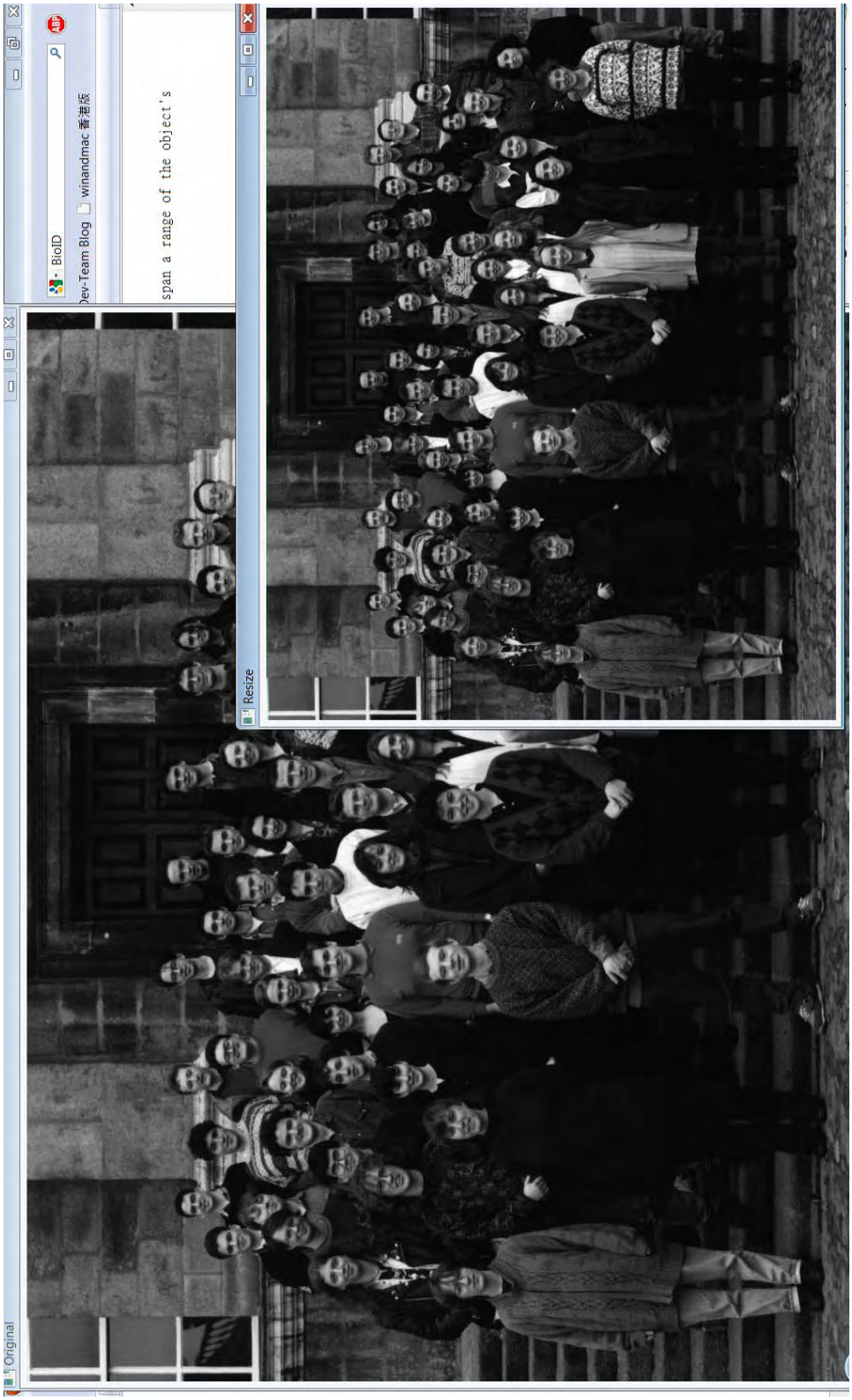
Result of Test case 002

Result of Test case 003

Result of Test case 004



Example screen shot of pre-processing



Convert into grayscale and resize the photo

After execute the histogram equalization method.

Result of face detection.

Source Code

FileHandleModule.py

```
import os,shutil
from opencv.cv import *


class file:
  def __init__(self):
    self.photoExt = ['jpg','pgm','png','JPG','bmp','gif','tiff']


  def fileExist(self,file):
    return os.path.exists(file)


  def isFile(self,file):
    return os.path.isfile(file)


  def isDir(self,dir):
    return os.path.isdir(dir)


  def getFileName(self,path):
    filename = os.path.basename(path)
    return filename


  def makeDir(self,path):
    os.mkdir(path,0777)


  def removeDirWithFile(self,path):
    ##remove any file in folder
    for filelist in os.listdir(path):
      file_path = os.path.join(path,filelist)
      try:
        if self.isFile(file_path):
          self.removeFile(file_path)
      except Exception, e:
        print e
    ##remove folder
```

```
    #self.removeDir(path)


  def removeDir(self,path):
    try:
      if self.isDir(path):
        os.unlink(path)
    except Exception, e:
      print e


  def moveDir(self,src,dst):
    shutil.move(src,dst)


  def removeFile(self,path):
    if self.isFile(path):
      os.remove(path)
      return 1
    else:
      print "Error: File(%s) is not found" % path
      return 0


  def renameDir(self,src,dst):
    if self.fileExist(src):
      os.rename(src,dst)
    else:
      print "Error: Src path is incorrect.Please check"
  #def isDir(self,file):
  #  return isdir(file)


  def listDir(self,dir):
    dirlist = []
    for d in os.listdir(dir):
      if os.path.isdir(os.path.join(dir,d)):
        dirlist.append(dir+d)
    return dirlist


  def listFile(self,dir):
    dir = dir+"/"
    file =[]
```

```
    #print os.listdir(dir)
    for filename in os.listdir(dir):
      if self.isFile(dir+ filename):
        #print filename
        ext= filename.split('.')[-1]
        #print basename,ext
        for i in range(len(self.photoExt)):
          if ext ==self.photoExt[i]:
            #print basename,ext
            file.append(dir+filename)
    return file
      #basename, extension = filename.split('.')
      #prefix,number = basename.split(' - ')


class xmlfile:
  def __init__(self,filename):
    self.storage = cvCreateMemStorage(0)
    self.fs = cvOpenFileStorage(filename,self.storage,CV_STORAGE_WRITE)

  #def openFile(self,filename):
  #  self.fs = cvOpenFileStorage(filename,0,CV_STORAGE_WRITE)

  def startWriteStruct(self,itemName):
    cvStartWriteStruct(self.fs,itemName,CV_NODE_MAP)

  def endWriteStruct(self):
    cvEndWriteStruct(self.fs)

  def loadFile(self,filename):
    return cvLoad(filename)

  def writeData(self,data,itemName):
    cvWrite(self.fs,itemName,data)

  def readData(self,itemName):
    cvRead(self.fs,itemName)

  def saveFile(self,filename):
```

```
    os.path.exist(filename)
    return cvSave(filename)

  def releaseFileStorage(self):
    cvReleaseFileStorage(self.fs)

  def readyByName(self,itemName):
      return cvReadByName(self.fs,0,itemName,0)

class textFile:
    def __init__(self):
      array = []

    def openFile(self,filename,type_):
      self.file = open(filename,type_)

    def readFile(self):
      return array

    def img2rowArray(self,imgName,img):
      data = imgName+"|"+matrix
      self.writeFile(data)

    def writeFile(self,data):
      self.file.write(data)

    def closeFile(self):
      self.file.close()

    #os.path.exists
if __name__ == '__main__':
  f = file()
  #filelist = f.listFile('BioID_Face_DB/training/1/')
  #for i in range(len(filelist)):
  f.listFile('E:/me/informatics/U08096/Face Database/MIT-CBCL-facerec-database/test/')
  #print "Total: %d" % len(filelist)
  #e = 'E:/me/informatics/U08096/me_work/Me_programming/images/'
  #print e.split('/')[-2]
```

```
  #f.listDir("./trainingImage/")
  #fs = xmlfile('trainingImage/test/face.xml')
```

ImageProcModule.py

```
from opencv.cv import *
from UIModule import *
from math import sin,cos,sqrt,exp,atan,degrees,radians
import sys,os

def vect2array(vect,o=False):
    arr=[]
    #print vect.rows,vect.cols
    for i in range(vect.rows):
      arr.append([])
      for j in range(vect.cols):
        if o:
            v = str(cvmGet(vect,i,j))
        else:
            v = str(int((cvmGet(vect,i,j))))
        arr[i].append(v)
    return arr

def isColor(img):
  if img.nChannels> 1: return 1
  else: return 0

def maskImage(img,mask):
  tmpMask = cvCreateImage(vSize(img.width,img.height),img.depth,img.nChannels)
  tmpMask = resize(mask,img.width,img.height)
  cvAnd(img,tmpMask,img)
  return img

def resize(src,w,h):
  dst = cvCreateImage(cvSize(int(w),int(h)),src.depth,src.nChannels)
  cvResize(src,dst,CV_INTER_AREA)
  return dst
```

```
#def reshapeIntoRowMat(src):
#    vect = cvReshape(src,0,1)
#    return vect
def displayImageData(img):
    for i in range(img.height):
      for j in range(img.width):
        s = cvGet2D(img,i,j)
        print s[0]
def image2Row(img):
  arr = []
  for i in range(img.width*img.height):
      arr.append(ord(img.imageData[i]))
      #arr.append = s
  print "%d elements are inserted in Array" % (img.width*img.height)
  return arr
def row2Image(mat,w,h):
  img = cvCreateImage(cvSize(w,h),IPL_DEPTH_8U,1)
  cvZero(img)
  ptr =0
  _min,_max = findMinMax(mat)
  for i in range(h):
    for j in range(w):
      s= cvmGet(mat,0,ptr)
      s = (s-_min)/(_max-_min)*255
      cvSet(img,i,j,cvScalar(round(s),0,0))
      ptr += 1
  return img


def findCenter(pt1,pt2):
    x = (pt1.x+pt2.x)/2
    y = (pt1.y+pt2.y)/2
    return cvPoint(int(x),int(y))


def findRotateAngle(pt1,pt2):
    #if pt2.y > pt1.y:
      val = float((pt2.y - pt1.y))/float((pt2.x - pt1.x))
      ang = atan(val)
```

```
      return degrees(ang)
    #else:
     # print "findRotateAngle"

def findMaxVauleInArray(arr):
  return max(arr)

def findDistanceBetweenTwoPoint(pt1,pt2):
  a = pow(float(pt2.y-pt1.y),2)
  b = pow(float(pt2.x-pt1.x),2)
  return sqrt(a+b)

def rotateArr(src,arr,ang):
  xc = cvGetSize(src).width/2
  yc = cvGetSize(src).height/2
  ang = radians(ang)
  #print xc,yc,ang
  newArr = []
  for i in range(len(arr)):
    #a = cos(ang)*(arr[i].x-xc)
    #b = sin(ang)*(arr[i].y-yc)
    #print cos(ang)
    #print a,b,arr[i].x,arr[i].y
    x = int(xc+float(cos(ang)*(arr[i].x-xc))+float(sin(ang)*(arr[i].y-yc)))
    y = int(yc+float(cos(ang)*(arr[i].y-yc))-float(sin(ang)*(arr[i].x-xc)))
    newArr.append(cvPoint(x,y))
  return newArr

def rotateImage(src,ang):
  dst = cvCloneImage(src)
  x = cvGetSize(src).width/2
  y = cvGetSize(src).height/2
  rotatetmp = cvCreateMat(2,3,CV_32FC1)
  cv2DRotationMatrix(cvPoint2D32f(x,y),ang,1.0,rotatetmp)
  cvWarpAffine(src,dst,rotatetmp)
  return dst

def loadImage(str):
```

```
  if not os.path.exists(str):
        print "Can't found:" + str
        print "Exit System"
        sys.exit(-1)
  else:
    src = cvLoadImage(str,CV_LOAD_IMAGE_ANYCOLOR)
    #/* 8 bit, color or gray - deprecated, use CV_LOAD_IMAGE_ANYCOLOR */
    return src

def saveImage(filename,img):
  try:
    cvSaveImage(filename,img)
    print "Saved"
    return 1
  finally:
    print "Save file:%s, Error" % (filename)
    return 0

def changeGray(src):
  gray = cvCreateImage(cvGetSize(src),8,1)
  cvCvtColor(src,gray, CV_BGR2GRAY )
  return gray

def changeYCrCb(src):
  gray = cvCreateImage(cvGetSize(src),8,1)
  cvCvtColor(src,gray, CV_BGR2YCrCb )
  return gray

def doGabor(img):
  kernel_size,var,w,phase,psi = 21,50,5,0,90
  if kernel_size%2==0:
          kernel_size += 1
  #print kernel_size
  kernel = cvCreateMat(kernel_size,kernel_size,CV_32FC1)
  kernelimg = cvCreateImage(cvSize(kernel_size,kernel_size),IPL_DEPTH_32F,1)
  big_kernelimg = cvCreateImage(cvSize(kernel_size*20,kernel_size*20),IPL_DEPTH_32F,1)
  src_f = cvCreateImage(cvGetSize(img),IPL_DEPTH_32F,1)
  cvConvertScale(img,src_f,1.0/255,0)
```

```
  dest = cvCloneImage(src_f)
  dest_mag = cvCloneImage(src_f)


  var = var/10.0
  w  = w/10.0
  phase = phase*CV_PI/180.0
  psi = CV_PI*psi/180.0
  cvZero(kernel)
  for x in range(-kernel_size/2+1,kernel_size/2+1):
      for y in range(-kernel_size/2+1,kernel_size/2+1):
          kernel_val = exp( -((x*x)+(y*y))/(2*var))*cos( w*x*cos(phase)+w*y*sin(phase)+psi)
          cvSet2D(kernel,y+kernel_size/2,x+kernel_size/2,cvScalar(kernel_val))
          cvSet2D(kernelimg,y+kernel_size/2,x+kernel_size/2,cvScalar(kernel_val/2+0.5))
  cvFilter2D(src_f,dest,kernel,cvPoint(-1,-1))
  cvResize(kernelimg,big_kernelimg)
  cvPow(dest,dest_mag,2)
  displayWindow('Mag',dest_mag)
  displayWindow('Kernel',big_kernelimg)
  displayWindow('Process',dest)
  return dest

def doLaplace(src):
  dst = cvCreateImage(cvGetSize(src),IPL_DEPTH_16S,1)
  cvLaplace(src,dst,3)
  return dst

def doSobelX(src):
  cvSobel(src,src,1,0,3)
  return src

def doSobelY(src):
  cvSobel(src,src,0,1,3)
  return src
"""
def hist(src):
  hist_size=[180,255]
  h_ranges = [0,180]
  s_ranges = [0,255]
```

```
  ranges = [h_ranges,s_ranges]
  frame_size = cvGetSize(src)
  hist = cvCreateHist( hist_size, cv.CV_HIST_ARRAY, ranges, 1 )
  hue = cvCreateImage (frame_size, 8, 1)
  saturation = cvCreateImage (frame_size, 8, 1)
  value = cvCreateImage (frame_size, 8, 1)
  hsv = cvCloneImage(src)
  cvCvtColor(src,hsv,CV_RGB2HSV)
  cvSplit(hsv,hue,saturation,value,None)
  cvCalcHist([hue,saturation],hist,0,None)
"""

def dilateImage(src):
  cvDilate(src,src,None,1)
  return src

def erodeImage(src):
  #element = cvCreateStructuringElementEx( pos*2+1, pos*2+1, pos, pos, CV_SHAPE_RECT, None );
  cvErode(src,src,None,1);
  return src

def smoothImage(src,int):
  cvSmooth(src,src,CV_GAUSSIAN,int,int)
  return src

def findContours(img):
  #create the storage area
  storage = cvCreateMemStorage(0)
  pImg = cvCreateImage(cvGetSize(img),8,1)
  n,contour = cvFindContours(img,storage,sizeof_CvContour,CV_RETR_EXTERNAL,CV_CHAIN_APPROX_SIMPLE)
  contour = cvApproxPoly(contour,sizeof_CvContour,storage,CV_POLY_APPROX_DP,3,0)
  #cvDrawContours(pImg,contour,CV_RGB(255,0,0),CV_RGB(0,0,255),0,1,CV_AA,cvPoint(0,0))
  #ellipse = cvFitEllipse2(contour)
  #cvEllipseBox(pImg,ellipse,CV_RGB(0,0,255),1)
  print contour.total
  displayWindow("findContours",pImg)

def doCanny(cImg,Thh):
```

```
    cannyImg = cvCreateImage(cvGetSize(cImg),8,1)
    cvCanny(cImg,cannyImg,Thh*0.4,Thh,3)
    displayWindow("Canny",cannyImg)
    return cannyImg

def doAdThresholdFilter(src,pos):
  dst = cvCreateImage(cvGetSize(src),8,1)
  cvAdaptiveThreshold( src, dst, pos , CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY, 3, 5 )
  return dst

def doThresholdFilter(src,pos):
  dst = cvCreateImage(cvGetSize(src),8,1)
  cvThreshold(src,dst,pos,255,CV_THRESH_BINARY)
  return dst

def doThresholdOtsu(src):
  cvThreshold(src,src,0,255,CV_THRESH_OTSU)
  return src

def doThreshold(src,pos):
  dst = cvCreateImage(cvGetSize(src),8,1)
  smoothImage(src)
  cvThreshold(src,dst,pos,255,CV_THRESH_BINARY)
  displayWindow("Result")

def findCircles(img):
  Img_size = cvGetSize(img)
  cstorage = cvCreateMemStorage(0)
  #img = smoothImage(img)
  #displayWindow("Smooth Image",img)
  #circles =
cvHoughCircles(img,cstorage,CV_HOUGH_GRADIENT,2,Img_size.height/10,5,5,Img_size.height/100,Img_size.hei
ght/5)
  circles = cvHoughCircles( img, cstorage, CV_HOUGH_GRADIENT, 2, 4*Img_size.height, 40, 40 )
  print "Circles are detected is %d" %(circles.total)
  #print circles.total
  if circles:
    for i in range(0,circles.total):
```

```
        p = cvGetSeqElem(circles,i)
        print p[1]
        #cvCircle( img, cvPoint(cvRound(p[0]),cvRound(p[1])), 3, CV_RGB(0,255,0), -1, 8, 0 )
    displayWindow("HoughCircle",img)


def findLines(img,HOUGH_STANDARD):
    lines = 0
    storage = cvCreateMemStorage(0)
    dst = cvCreateImage(cvGetSize(img),8,1)
    color_dst = cvCreateImage(cvGetSize(img),8,1)
    cvCanny(img,dst,50,200,3)
    #cvCvtColor(dst,color_dst,CV_GRAY2BGR)
    if HOUGH_STANDARD:
      lines = cvHoughLines2(dst,storage,CV_HOUGH_STANDARD,1,CV_PI/180,100,0,0)
      for i in range(min(lines.total, 100)):
              line = lines[i]
              rho = line[0];
              theta = line[1];
              pt1 = CvPoint();
              pt2 = CvPoint();
              a = cos(theta);
              b = sin(theta);
              x0 = a*rho
              y0 = b*rho
              pt1.x = cvRound(x0 + 1000*(-b));
              pt1.y = cvRound(y0 + 1000*(a));
              pt2.x = cvRound(x0 - 1000*(-b));
              pt2.y = cvRound(y0 - 1000*(a));
              cvLine( color_dst, pt1, pt2, CV_RGB(255,0,0), 1, 8 );

    else:
      lines = cvHoughLines2(dst,storage,CV_HOUGH_PROBABILISTIC,1,CV_PI/180,50,50,10)
      for line in lines:
          cvLine( color_dst, line[0], line[1], CV_RGB(255,0,0), 1, 8 );
    print "%d line(s) are detected"%(lines.total)
    displayWindow("HoughLines",color_dst)
    #return color_dst
```

```
def findImageWidth(img):
  return img.width


def findImageHeight(img):
  return img.height
```

fdModule.py

```
import sys,os,Image,time
from opencv.cv import *
#from opencv.highgui import *
from UIModule import *
from fontModule import *
from imageToolModule import *
from imageProcModule import *
from fileHandleModule import *


class faceDetect:
    def __init__(self,img):
      self.debugFlag = 0
      #flag to save each faces in JPG or not
      self.saveFlag = 0
      # Global Variables
      self.imgScale(img)
      #self.image_scale = 1.5
      #self.Img_size = cvGetSize(self.img)
      ###################
      cascade = None
      self.storage = cvCreateMemStorage(0)
      cascade_name = "haarcascades/haarcascade_frontalface_alt.xml"
      # Parameters for haar detection
      # From the API:
      # The default parameters (scale_factor=1.1, min_neighbors=3, flags=0) are tuned
      # for accurate yet slow object detection. For a faster operation on real video
      # images the settings are:
      # scale_factor=1.2, min_neighbors=2, flags=CV_HAAR_DO_CANNY_PRUNING,
      # min_size=<minimum possible face size
```

```
        self.min_size = cvSize(20,20)
        self.haar_scale = 1.2
        self.min_neighbors = 2
        self.haar_flags = 0
        if not os.path.exists(cascade_name):
          print "Can't found" + cascade_name
          print "Exit the system"
          sys.exit(-1)
        self.cascade = cvLoadHaarClassifierCascade( cascade_name, cvSize(1,1) );
        if not self.cascade:
          print "ERROR: Could not load classifier cascade"
          print "Exit the system"
          sys.exit(-1)
    ##function to determine the img scale based on img_size.width
    def imgScale(self,path):
      # allocate temporary images
      self.imgF = path
      #Load Image into memonry, CV_LOAD_IMAGE_COLOR = 1
      if not os.path.exists(self.imgF):
        print "file:%s not found" % self.imgF
        print "Exit the system"
        sys.exit(-1)
      self.img = cvLoadImage(self.imgF,CV_LOAD_IMAGE_COLOR);
      #self.img = Image.open(self.imgF)
      #self.img = cvLoadImage(self.imgF,-1)
      #get the image's size
      self.Img_size = cvGetSize(self.img)
      if (self.Img_size.width > 2000):
        self.image_scale = 2
      else:
        self.image_scale = 1.5

    def detectFacePos(self):
      #print isColor(self.img)
      #print self.img.channels
      #create a gray image for further processing
      self.gray = cvCreateImage( self.Img_size, 8, 1 );
```

```
        self.small_img = cvCreateImage( cvSize( cvRound (self.Img_size.width/self.image_scale), cvRound
(self.Img_size.height/self.image_scale)), 8, 1 );
        if isColor(self.img):
          # convert color input image to grayscale
          cvCvtColor( self.img, self.gray, CV_BGR2GRAY );
        # scale input image for faster processing
        cvResize( self.gray, self.small_img, CV_INTER_LINEAR );
        cvEqualizeHist( self.small_img, self.small_img );
        #displayWindow("Equ Hist",self.small_img)
        cvClearMemStorage(self.storage);
        if(self.cascade):
          self.t=cvGetTickCount();
          #Face Detector
          self.faces = cvHaarDetectObjects( self.small_img, self.cascade, self.storage,self.haar_scale,
self.min_neighbors, self.haar_flags, self.min_size );
          self.t = cvGetTickCount() - self.t;
          print "Face(s) detection time = %gms" % (self.t/(cvGetTickFrequency()*1000.));
          if self.debugFlag:
              print "Face(s) detection time = %gms" % (self.t/(cvGetTickFrequency()*1000.));
              print "%d of Face(s) are detected" % self.faces.total
          return self.faces,self.image_scale


    def setDebugFlag(self,status):
        self.debugFlag = status


    def showFacePos(self,faces):
        return


    def saveFaceIntoJPG(self,i,pt1,pt2):
      self.rect = cvRect(pt1.x,pt1.y,pt2.x-pt1.x,pt2.y-pt1.y)
      tempArray = cvGetSubRect(self.img,self.rect)
      fileName, fileExt = os.path.splitext(self.imgF)
      cvSaveImage(fileName+"_"+str(i)+fileExt,tempArray)
      #displayWindow('Face',tempArray)


    def getTotalNumberDetectedFace(self):
      if self.faces.total:
        return self.faces.total
```

```
    def showFaceWithRect(self,faces):
      self.faces = faces
      #print how many faces in photo
      #print "%d face(s) are detected."%(self.faces.total)
      i = 1
      if self.faces:
        for face_rect in self.faces:
          # the input to cvHaarDetectObjects was resized, so scale the
          # bounding box of each face and convert it to two CvPoints
          pt1 = cvPoint( int(face_rect.x*self.image_scale), int(face_rect.y*self.image_scale))
          pt2 =
cvPoint( int((face_rect.x+face_rect.width)*self.image_scale),int((face_rect.y+face_rect.height)*self.im
age_scale) )
          if self.saveFlag:
            #save face into JPG file
            self.saveFaceIntoJPG(i,pt1,pt2)
            i = i+1
          #hightlight face here
          drawRect(self.img,pt1,pt2,0)
          #addText(self.img,str(pt1),pt1)
          #addText(self.img,str(pt2),pt2)
        #displayWindow("showFaceWithRect",self.img)


    def saveRectImage(self,path):
      if self.img:
        filename = os.path.basename(self.imgF)
        cvSaveImage(path+filename,self.img)
        #im = Image.open(str(path+filename))
        #im.save(path+filename,quality=50)


    def setSaveFlag(self,status):
        self.saveFlag = status

    def getFaceImg(self,img,f,scale):
```

```
        rect =
cvRect(int(round(f.x*scale)),int(round(f.y*scale)),int(round(f.width*scale)),int(round(f.height*scale))
)
        faceImg = cvGetSubRect(img,rect)
        return faceImg




if __name__ == '__main__':
  flag = 0
  if flag:
    f = file()
    #for BioID face database
    #filelist = f.listFile("E:/me/informatics/U08096/Face Database/BioID/list of BioID/")
    #savePath = "../Testing_Results\BioID/"
    ##for MIT-CBCL face database
    #filelist = f.listFile("E:/me/informatics/U08096/Face Database/MIT-CBCL-facerec-database/test/")
    #savePath = "../Testing_Results/MIT-CBCL/"
    ##for ORL face database
    #filelist = f.listFile("E:/me/informatics/U08096/Face Database/ORLDatabase/")
    #savePath = "E:/me/informatics/U08096/Face Detection/Testing_Results/ORL/"
    ##for AraNefian face database
    #filelist = f.listFile("E:/me/informatics/U08096/Face Database/Ara Nefian FR/cropped_faces/")
    #savePath = "E:/me/informatics/U08096/Face Detection/Testing_Results/AraNefianFR/"
    ##for YaleB face database
    #filelist = f.listFile("E:\me\informatics\U08096\Face Database\YaleB\yaleB01_B10/")
    #savePath = "../Testing_Results/YaleB/"
    ##for CMU_FD face database
    filelist = f.listFile("E:/me/informatics/U08096/Face Database/CMU_Face_Database/test-low_jpg/")
    savePath = "../Testing_Results/CMU_FD/test-low/"
    i = 0
    for file in filelist:
      fd = faceDetect(file)
      facePos,scale = fd.detectFacePos()
      fd.showFaceWithRect(facePos)
      fd.saveRectImage(savePath)
      if fd.getTotalNumberDetectedFace() is None:
        print "File %s has no face(s)" % file
```

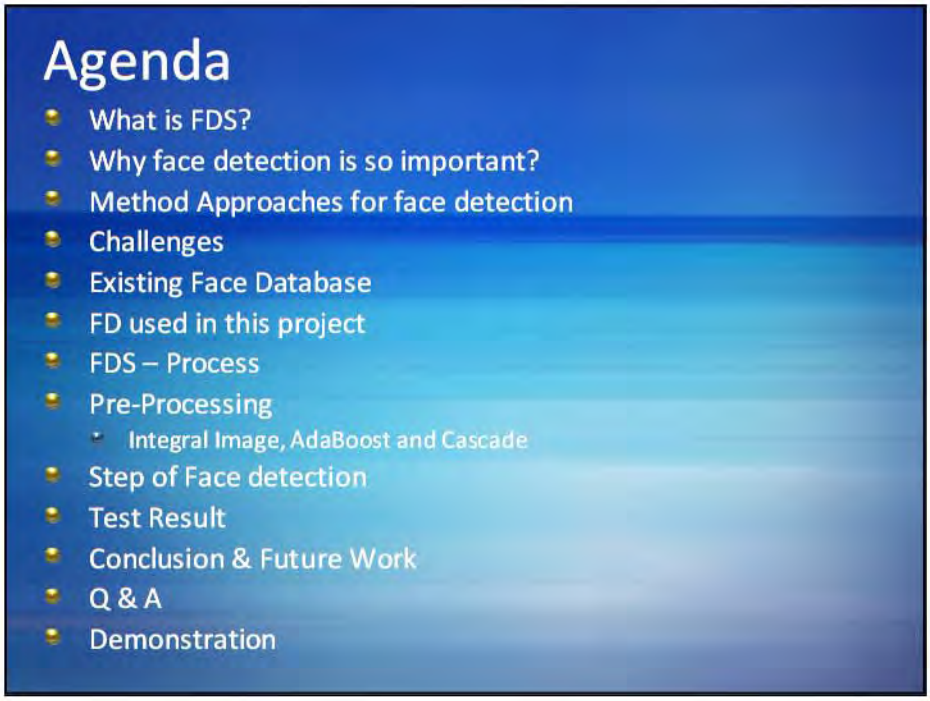
```
        i = i+ 1
    print "Total Non-face in picture is %d" % i
  else:
    #imgFile =
"E:\me\informatics\U08096\me_work\me_programming_\me_programming_\images\\061104prize263s.JPG"
    imgFile = "E:\me\informatics\U08096\Face Database\CMU_Face_Database\newtest_jpg/class57.jpg"
    print imgFile
    fd = faceDetect(imgFile)
    #fd.setSaveFlag(1)
    facePos,scale = fd.detectFacePos()
    print fd.getTotalNumberDetectedFace()
    #print facePos
    fd.showFaceWithRect(facePos)
    #fd.saveRectImage(savePath)
    cvWaitKey(0)
    destroyAllWindows()
  """
  f = file()
  filelist = f.listFile("E:/me/informatics/U08096/Face Database/BioID/BioID-FaceDatabase-V1.2/")
  #print filelist
  imgFile = "E:/me/informatics/U08096/Face Database/BioID/BioID-FaceDatabase-V1.2//BioID_0000.pgm"
  savePath = "E:\me\informatics\U08096\Face Detection\Testing_Results\BioID/"
  #print f.getFileName(imgFile)
  fd = faceDetect(imgFile)
  #fd.setSaveFlag(1)
  facePos,scale = fd.detectFacePos()
  #print facePos
  fd.showFaceWithRect(facePos)
  fd.saveRectImage(savePath)
  if not facePos:
    print "no face detection"
  cvWaitKey(0)
  #release any resource
  destroyAllWindows()
  """
```

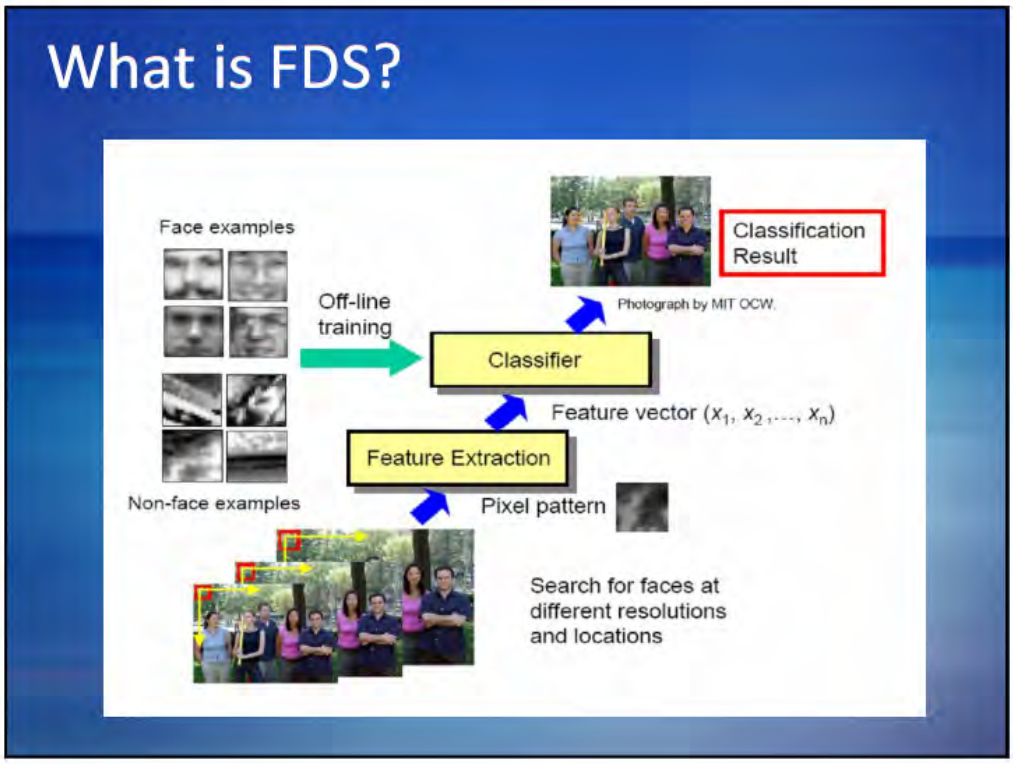
Development Day (Gantt chart)

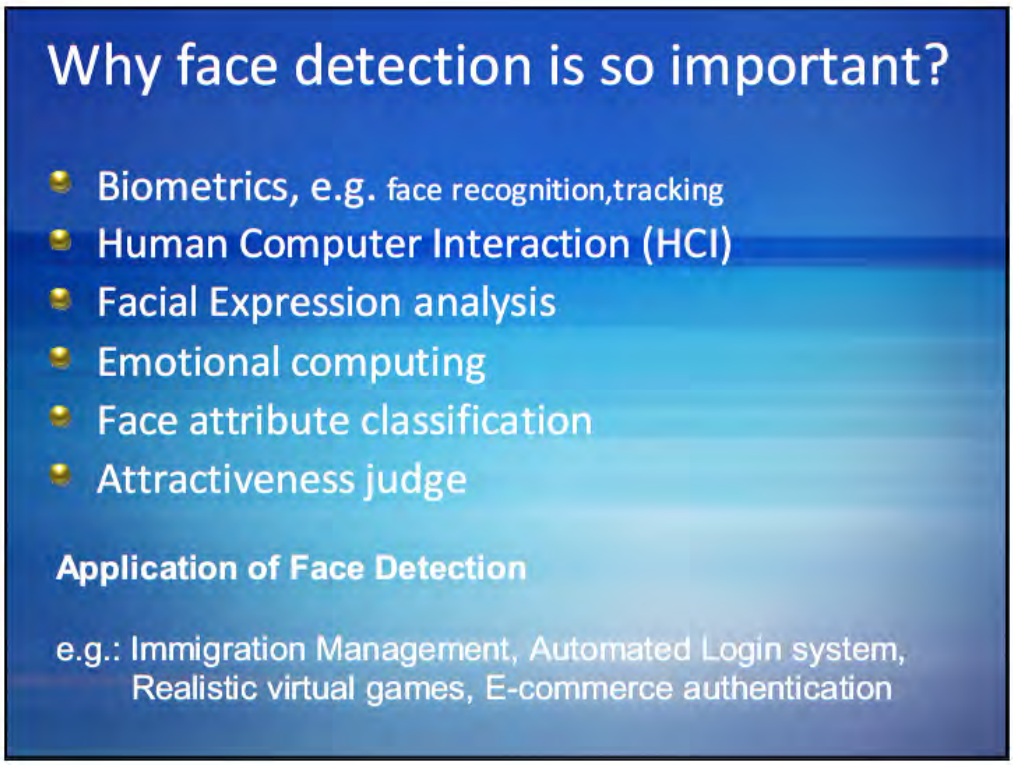
The development day is shown below:

Step	Tasks	Duration (days)
1	Preparation	32
1.1	Researching	15
1.2	Requirement Definition	7
1.3	Prepare hardware architecture	5
1.4	Prepare software architecture	5
2	Design	33
2.1	File module design	5
2.2	Face Detection module design	15
2.3	User Interface design	7
2.4	Design review and evaluation	3
2.5	Programming code planning	3
3	Development	71
3.1	Programming code construct	47
3.1.1	Programming code in file module	10
3.1.2	Programming code in Face Detection module	30
3.1.3	Programming code in User Interface	7
3.2	User documentation	3
3.3	Test code implementation	3
3.4	Implementation review and evaluation	3
3.5	Integration	10
3.6	Error checking	5
4	Testing	6
4.1	File Module testing	2
4.2	Face Detection test	2
4.3	User Interface	2
5	Production	25
5.1	Document the system	10
5.2	Production the system	10
5.3	Monitor and error fix for system after production	5
6	Maintenance	10
6.1	Maintenance Phase start	10
	Total day(s)	177

Presention Slides




What is FDS?
Face examples
Non-face examples
Off-line training
Classifier
Feature vector (x1, x2,..., xn)
Feature Extraction
Pixel pattern
Classification Result
Photograph by MIT OCW.
Search for faces at different resolutions and locations


Why face detection is so important?
Biometrics, e.g. face recognition,tracking
Human Computer Interaction (HCI)
Facial Expression analysis
Emotional computing
Face attribute classification
Attractiveness judge
Application of Face Detection
e.g.: Immigration Management, Automated Login system,
Realistic virtual games, E-commerce authentication

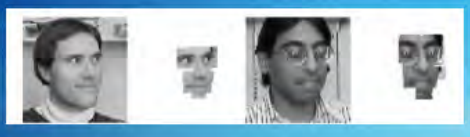
Method Approaches for FD

2C & 4M: two categories and 4 methods

- Global features
- Component features
- Knowledge based methods
  - the relationship between facial feature. E.g. two eyes, one nose and one mouth
- Template marching
  - few standard patterns stored to find same patterns in faces.
- Invariant feature methods
  - find structure features of a face. E.g. eyebrows, nose.
- Learning based methods
  - this models are learned from a set of training images.

Example of FD methods

Global feature



Component feature

Challenges

- Variant pose:
  -people not always orient to camera.
  e.g. frontal, upside Down.

Challenges (cont'd)

- Illumination condition:
  - Lighting and camera characteristics affect the appearance of a face.

Photos from Peter N. Belhumeur, Columbia University

Challenges (cont'd)

- Facial Expression:
  - Different expression in the face is presented different information to the machine.

Challenges (cont'd)

- Face Occlusion:
  - e.g. glasses, respirator, scarf

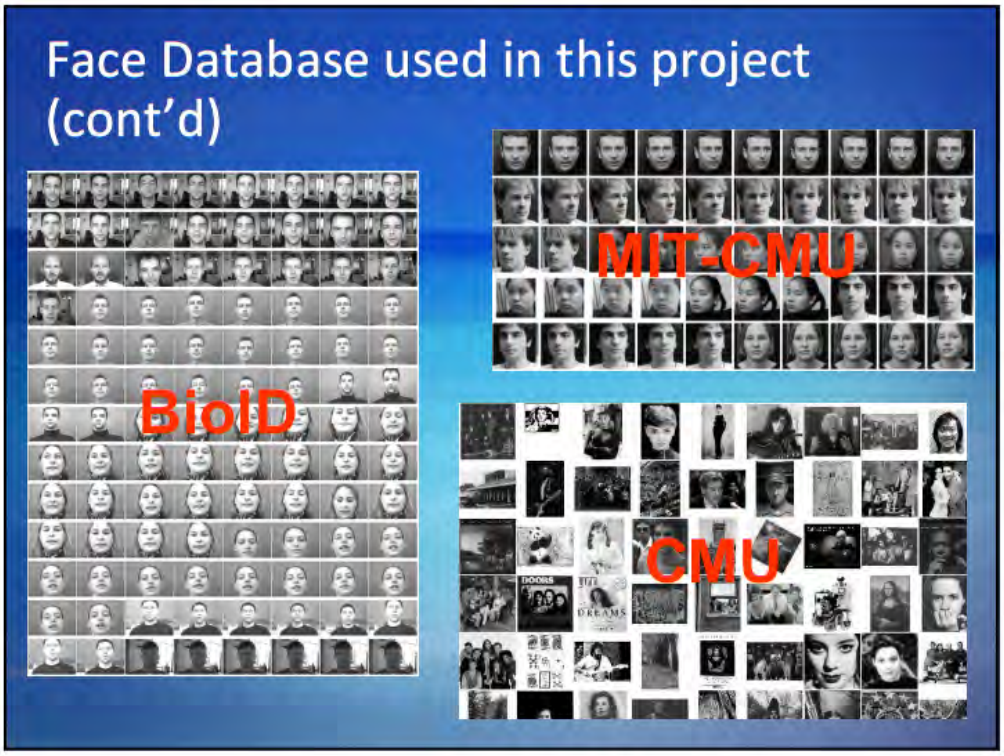
Photos from GTAV Face Database

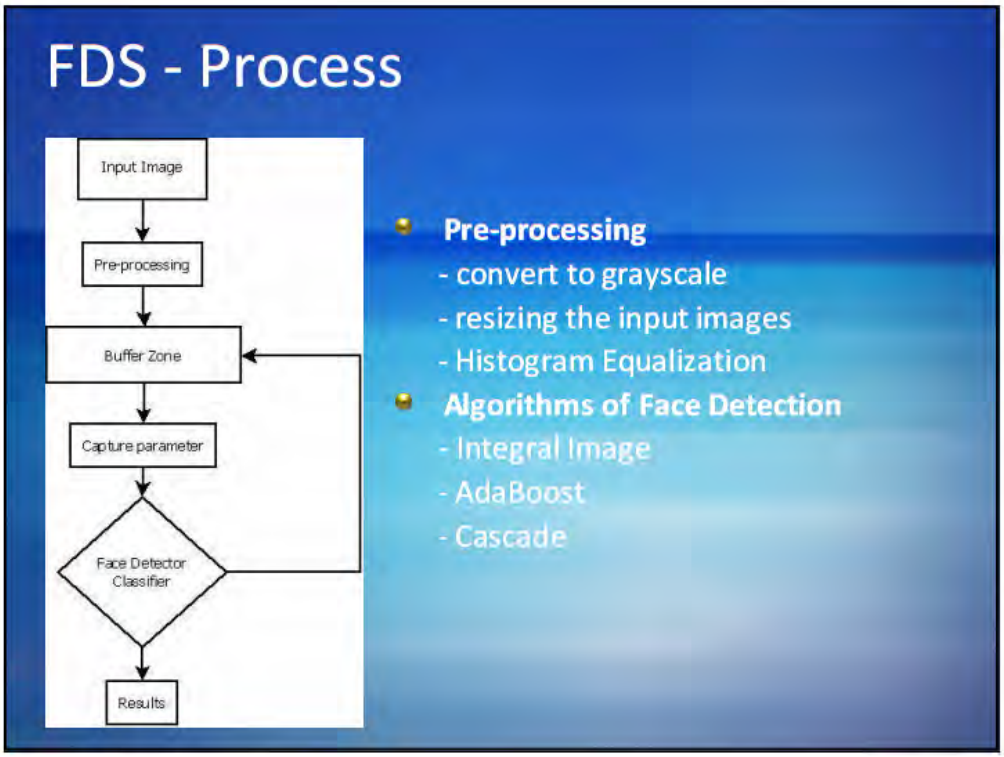
Existing Face Database

- FERET
- **BioID**
- CAS-PEAL
- Yale Face Database
- XM2VTS
- Purdue AR
- Yale Face Database B
- AT&T (Olivettti) Database (ORL)
- PIE Database
- JAFFE Database
- **MIT Face Database**
- **CMU Face Database**

Face Database used in this project

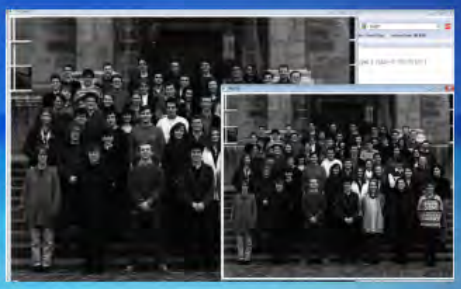
FD Name	Description
BioID (Test Set A)	1521 faces in 1521 images. Each image is grayscale with the size of image is 384 x 286. Indoor environment.
MIT-CMU (Test Set B)	2000 faces in 2000 images. Each one has different pose orientation with the size of images is 115 x 115.
CMU Face (Test Set C)	511 faces in 130 images. Each face is face vary and uncontrolled condition.


Face Database used in this project (cont'd)
BioID
MIT-CMU
CMU


FDS - Process
Input Image
Pre-processing
Buffer Zone
Capture parameter
Face Detector Classifier
Results
Pre-processing
- convert to grayscale
- resizing the input images
- Histogram Equalization
Algorithms of Face Detection
- Integral Image
- AdaBoost
- Cascade

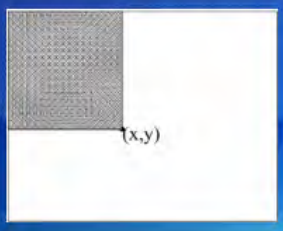
Pre-Processing

- Convert Into GrayScale
- Resize



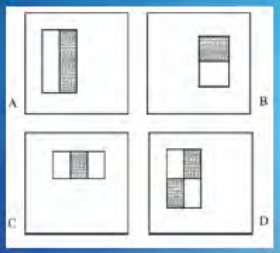
- Histogram Equalization

Integral Image



Def: The *integral image* at location (*x*,*y*), is the sum of the pixel values above and to the left of (*x*,*y*), inclusive.

Using the integral image representation one can compute the value of any rectangular sum in constant time.



AdaBoost

- AdaBoost is: (proposed by Paul Viola, Michael Jones)
  - a short form of **Adaptive Boosting**.
  - machine learning algorithms.
  - a sequence of best weak classifier with best combining weights.
  - training is required.

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier
Features vector
Weight
Weak classifier

AdaBoost (cont'd)

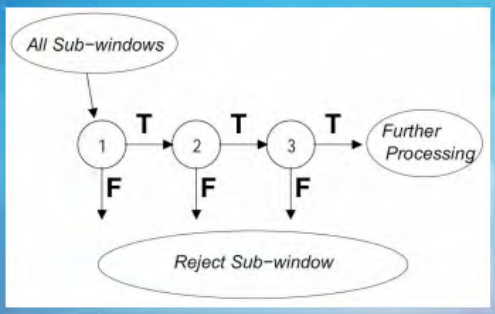
- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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 $\alpha_t = \log \frac{1}{\beta_t}$

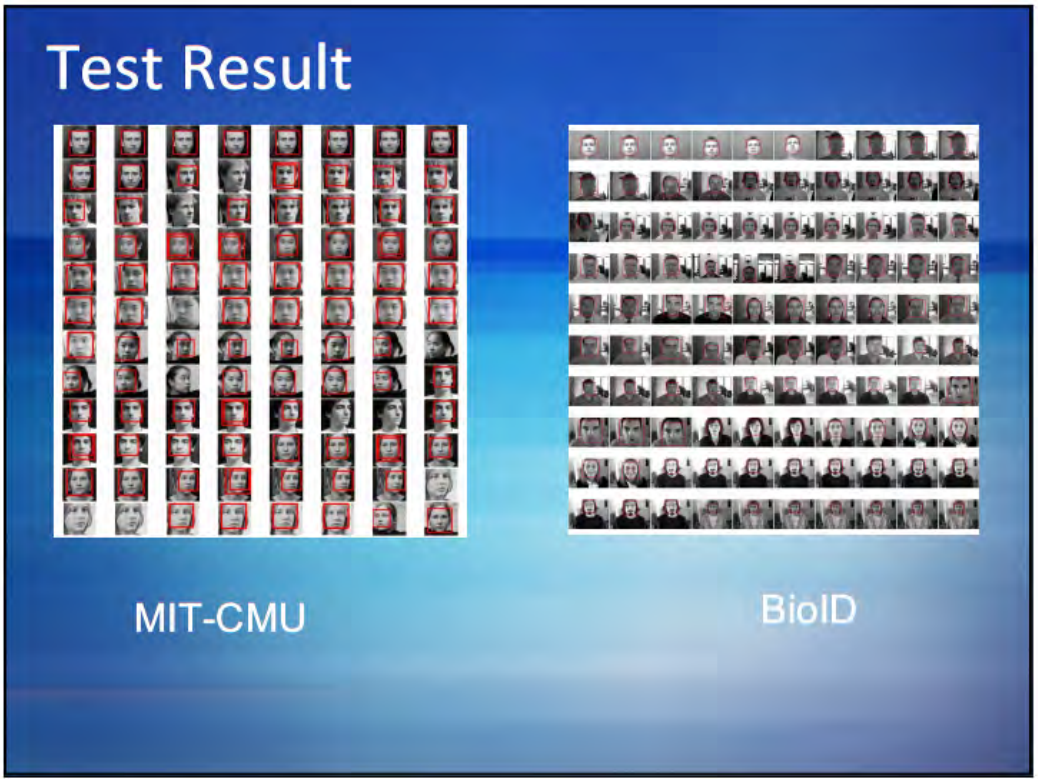
Cascade

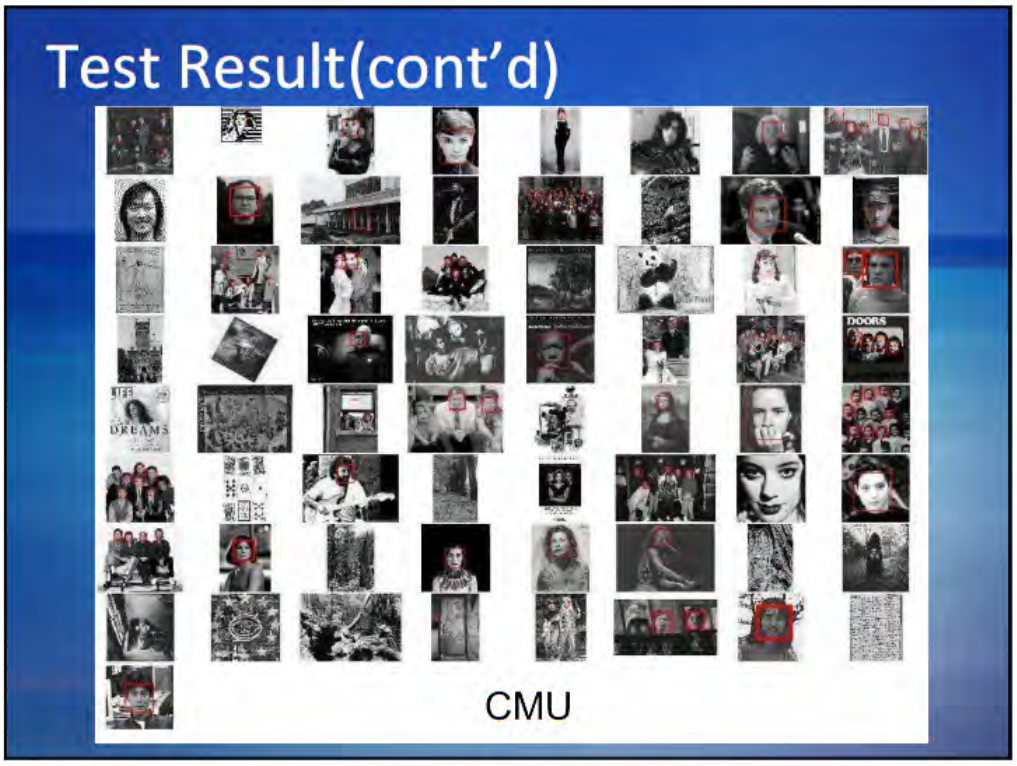
- each weak classifier only classify selected features.
- Former classifier eliminates a large portion of non-face component
- Series of weak classifier can achieve good detection

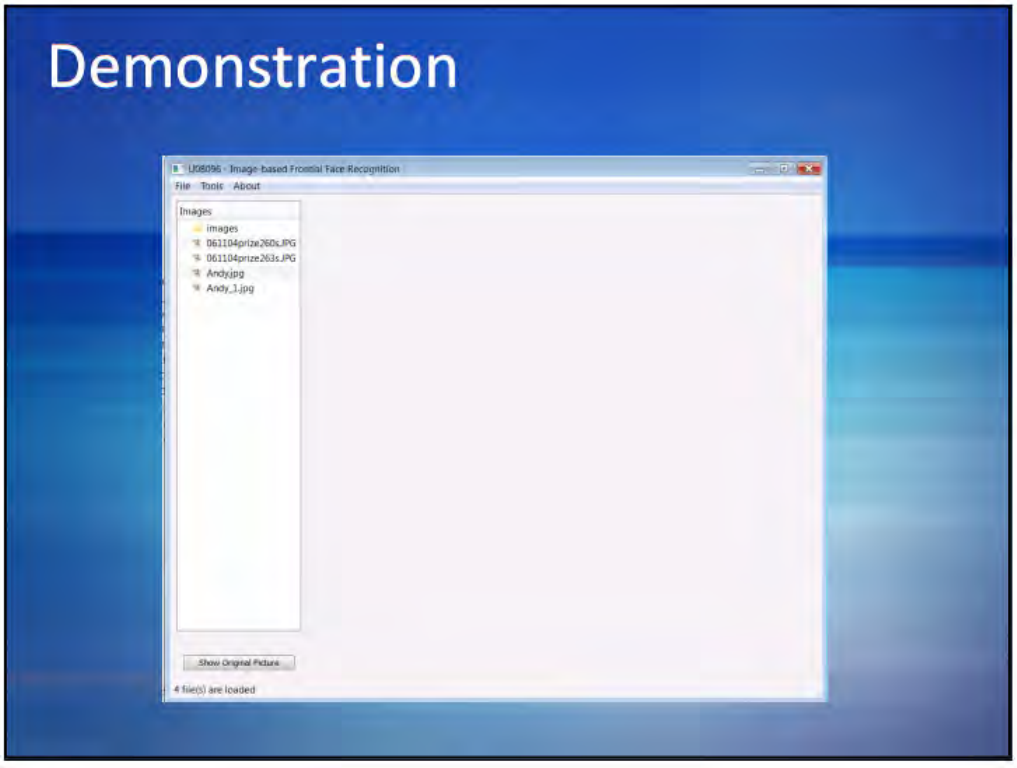


Step of Face Detection


Step of Face Detection


Test Result
MIT-CMU
BioID


Test Result(cont'd)
CMU


Demonstration
U08096 - Image-based Frontial Face Recognition
File Tools About
Images
images
061104prize260s.JPG
061104prize263s.JPG
Andy.jpg
Andy_1.jpg
Show Original Picture
4 file(s) are loaded

Conclusion

- FDS got at least 80% detection rate on uncontrolled dataset.
- FDS got at least 96% detection rate on indoor environment.
- FDS got average 100ms detection rate for 384x286 size on Netbook.

  (Dell mini 12: CPU Z530@1.6GHz,1G Ram)

Future Work

- Increase the detection rate
- Deploy eye detector or skin detector to increase detection rate.

Student Progress Form

Progress Report No : ___1_____ Date : _15 Nov 2008___
(to be completed by student)

Face Detection System (FDS) is proposed when I was looking for information from the internet, because of it is a hot topic in computer vision. I think this is an interesting topic, encouraging me to do a lot of works related to jobs.

At this stage, I am starting to setting up a deployment environment. Try to figure out what they are working.

I really appreciate some to those who have contributed to human. I think this topic is this.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___1_____(hours)

Start Time:_19:00_____ End Time:_____20:00_____

Progress Report No : ___2_____ Date : __29 Nov 2008___

(to be completed by student)

The proposal of this project submitted and reviewed.

Data searching of this topic is mainly focusing on introduction and literature review.

Programming Code just started.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___0.5_____(hours)

Start Time:_19:00_____ End Time:_____19:30_____

Progress Report No : ___3_____ Date : __20 FEB 2009___

(to be completed by student)

Literature review and introduction are completed. At this stage, I am focusing on the programming code about face detection to make sure, it is a possible to work in my project.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___0.5_____(hours)

Start Time:_19:30_____ End Time:_____20:00_____

Progress Report No : ___4_____ Date : _27 March 2009__

(to be completed by student)

Pre-processing is applied in the programming code to get highly detection rate. Also,

User interface is designing with using Pyqt4.

I am still composing the “Methodology for project” section in my project.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___1_____(hours)

Start Time:_19:00_____ End Time:_____20:00_____

Progress Report No : ___5_____ Date : _29 MAY 2009__

(to be completed by student)

User Interface is completed.

I am start to compose the “Analysis and Design – Face Detection System” section of project.

Testing will be carried out within a next months. I hope that everything keep going well properly.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___1_____(hours)

Start Time:_19:00_____ End Time:_____20:00_____

Progress Report No : ___6_____ Date : _27 NOV 2009__

(to be completed by student)

To prepare presentation slides and final report.

Progress evaluation (to be completed by the supervisor)

General Progress:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	
Research:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Tool Studying:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A
Implementation:	(a) very slow	(b) behind	(c) satisfactory	(d) impressive	(e) done (f) N/A

Comments:

Suprevisor Signature:_____ Duration ___1_____(hours)

Start Time:_19:00_____ End Time:_____20:00_____

Project Proposal

Oxford Brookes University

U08096 Proposal of Final Year Project

Image-based
Face Detection System

Student Name: Tsang Tat Man

Student ID: 0301-0506-0333

Supervisor: Lo Chi Wing, Peter

Module Number: U08096

Programme: Bsc (Hons) Computer and Information System

Date: 30 November 2009

Table of Content

Abstract

Face is one of the physiological biometrics based on stable features. Face detection is a challenging mission. It is because faces in the images are all uncontrolled.

In this project, learning based methods are deployed: AdaBoost Method. AdaBoost is a machine learning algorithms which if formed from a sequence of weak classifier. Each of weak classifier forms a cascade and contributes a strong classifier in this project as face detector.

The aim of this project is to develop a face detection system implemented by AdaBoost algorithms. The faces in image are experienced with pre-processing before doing face detection. The methods are' Image Resizing', 'Grayscale conversion' and 'Histogram Equalization' which help system to speed up the detection speed, enhance the images quality.

Introduction

In a recent decade, computer vision becomes a new area of research. It mainly solved the problem to construct intelligent system. The aim of face detection is detected human faces from any images or videos.

Face detection can be regarded as object detection.[1] In this case, face is a target object. System should be reported has or not face(s) in the input images, and located the location of each face in given images.

Face detection is the first step of face recognition. Face Recognition is widely concerned in computer vision.

Project Objectives

The objective of the project is developed and proposed a system to detect human faces effectively. Input images may be varied with face size, complex of background and illumination condition.

Cross-platform is one of the mainly point should be considered, it is because it is a software trend to develop high valuably software.

Justification of the Project

The following research areas are the summary of Face detection why this project is so important to do:

1. Biometrics
2. Human Computer Interaction
3. Facial Image coding/ compression
4. Facial Expression analysis
5. Emotional computing
6. Face attribute classification
7. Attractiveness judge

Project Requirements

The followings are the list of the hardware and software requirement.

Hardware:

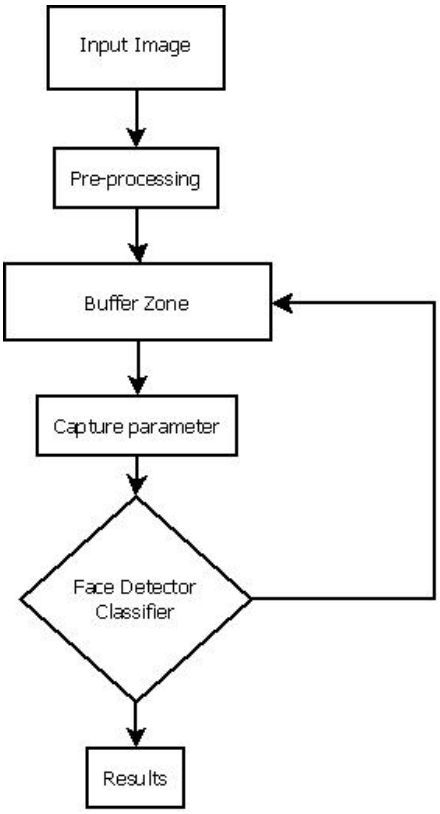
- Intel Pentium Processor 1 GHz or greater
- 128 RAM capacity or more
- 32 MB Graphic Card RAM capacities or more
- 1GB hard disk space or more.

Software:

In order to achieve this program can be executed difference platform, the followings are used.

- Python Programming Language
- PyQt4 (User Interface)
- OpenCV (A sets of library relate with computer vision.)

Approach and Methods



The above figure is the flowchart of the face detection system (FDS). This project will be focused on “Pre-processing”, “Face Detector”.

Pre-processing

Each input images are experienced with resizing and converting into grayscale. It is because most of the images are taken in uncontrolled environment. Pre-processing is necessary, because of enhance the quality of each given images, which enhance the detection rate. The pre-processing of the face detection are:

1. Grayscale conversion
2. Image Resizing
3. Histogram Equalization

Algorithms of Face Detection

The core of face detection are using following techniques as follows:

1. Integral Image
2. AdaBoost and Cascade

Impact Assessment

The performance of face detection can be measured from the detection rate. Higher detection rate means higher efficiency of the system. Beside the detection rate, we still need to consider another parameter – false positive rate. False positive rate is the rate of non-face objects are report as positive. [2]

Reference

1 Face detection - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Face_detection accessed 14 June 2009

2 Type I and type II errors - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Type_I_and_type_II_errors accessed Nov 2008

References

1 Computer vision - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Computer_vision accessed 14 June 2009

2 BBC NEWS | UK | Britain is 'surveillance society', http://news.bbc.co.uk/2/hi/uk_news/6108496.stm accessed on 14 June 2009

3 Python (programming language) - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Python_%28programming_language%29 access May 2009

4 Guido van Rossum - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Guido_van_Rossum, access May 2009

5 PyQt4 - PythonInfo Wiki, http://wiki.python.org/moin/PyQt4 access June 2009

6 Riverbank | Software | PyQt | What is PyQt?, http://www.riverbankcomputing.co.uk/software/pyqt/intro, access June 2009

7 SourceForge.net: Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/, accessed May 2009

8 OpenCV – Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/OpenCV, accessed May 2009

9 Welcome – OpenCV Wiki, http://opencv.willowgarage.com/wiki/, accessed May 2009

10 DARPA Grand Challenge – Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/DARPA_Grand_Challenge, accessed March 2009

11 Face detection – Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Face_detection, accessed December 2008

12 Face detection - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Face_detection accessed 14 June 2009

13 Yang W.Q., B.L. Goulart, K. Demchak, and Y.D. Li. 2002. Interactive effects of mycorrhizal inoculation and organic soil amendments on nitrogen acquisition and growth of highbush blueberry. Journal of the American Society for Horticultural Science. Vol. 127, No. 5. p. 742-748.

14 An Introduction to Face Recognition Technology, http://inform.nu/Articles/Vol3/v3n1p01-07.pdf accessed May 2009

15 Human–computer interaction - Wikipedia, the free encyclopedia,

http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction accessed May 2009

16 An Introduction to Face Detection and Recognition, http://www.ee.pdx.edu/~mperkows/CLASS_479/Biometrics/FaceRecognition/IntroFaceDetectRecognition.ppt accessd 16 June 2009

17 Bernd Heisele, Purdy Ho, Jane Wu, and Tomaso Poggio, “Face recognition: componenet-based versus global appraches”, Computer Vision and Image Understanding 91 (2003) 6-21

18 The FERET Database, http://www.itl.nist.gov/iad/humanid/feret/, accessed March 2009

19 University College Dublin , http://ee.ucd.ie/~prag/, accessed July 2009

20 BioID: BioID:Downloads:BioID Face Database, http://www.bioid.com/downloads/facedb/index.php, accessed March 2009

21 The PEAL Face Databse, http://www.jdl.ac.cn/peal/, accessed March 2009

22 Yale Face Database, http://cvc.yale.edu/projects/yalefaces/yalefaces.html, accessed March 2009

23 The XM2VTS Database, http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/, accessed March 2009

24 Yale Face Database, http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html, accessed March 2009

25 Robotics Institute: PIE Database, http://www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261, accessed May 2009

26 Facial Expression Database: Japanese Female Facial Expression (JAFFE) Database, http://www.kasrl.org/jaffe.html, accessed March 2009

27 Biometrics: Face recognition, http://pagesperso-orange.fr/fingerchip/biometrics/types/face.htm access Nov 2009
28 Type I and type II errors - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Type_I_and_type_II_errors accessed May 2009

29 Publications of Shuguang Shan, http://www.jdl.ac.cn/user/sgshan/publications.htm accessed Nov 2009

30 Nikon S60 advertising/design goodness - advertising and design blog, http://www.frederiksamuel.com/blog/2008/12/nikon-s60.html accessed Sep 2009

31 Viola,P.; Jones, M.,”*Rapid object detection using a boosted cascade of simple*

feature", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), ISSN: 1063-6919, Vol. 1, pp. 511-518, December 2001

32 FaceDetection – OpenCV Wiki, http://opencv.willowgarage.com/wiki/FaceDetection, accessed May 2009

33 Glossary – Grayscale Images, http://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm, accessed May 2009

34 Histogram equalization - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Histogram_equalization accessed May 2009

35 Paul Viola, Michael Jones, Robust Real-Time Face Detection, International Journal of Computer Vision,2004.

36 Integral image-based representations, www.cse.yorku.ca/~kosta/CompVis.../**integral**_representations.pdf accessed Dec 2009

37 Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139,August 1997.

38 Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *In Proceedings of the Fourteenth International Conference on Machine Learning*, 1997

39 AdaBoost - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/AdaBoost, accessed on July 2009

40 Yoav Freund and Robert and E. Schapine. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95*, pages 23-37. Springer-Verlag, 1005

41 Viola,P.; Jones, M.,"*Rapid object detection using a boosted cascade of simple feature*", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), ISSN: 1063-6919, Vol. 1, pp. 511-518, December 2001