FACE RECOGNITION SYSTEM USING COMPLETE GABOR FILTER WITH RANDOM FOREST

LOW JENG LAM

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor (Hons.) of Mechatronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2013

DECLARATION

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"FACE RECOGNITION SYSTEM USING COMPLETE GABOR FILTER WITH RANDOM FOREST"** was prepared by **LOW JENG LAM** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	______________________
Supervisor	:	Mr. See Yuen Chark
Date	:	______________________

Specially dedicated to

my beloved mother and father

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr, See Yuen Chark for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent who had given me encouragement throughout the project. Last but not least, I would like to thank all my friends who had given me guidance, support and knowledge in succeeding this project.

FACE RECOGNITION SYSTEM USING COMPLETE GABOR FILTER WITH RANDOM FOREST

ABSTRACT

This project attempts to resolve challenges like illumination changes, occlusions, and head orientation pose in face recognition by developing a technique called Complete Gabor Classifier with Random Forest. Complete Gabor Classifier is the hybrid method of Gabor Filter and oriented Gabor phase congruency image (OGPCI) to form a robust face recognition system. Gabor filter provides the magnitude information of Gabor responses, where OGPCI contains phase information of Gabor response. Random Forest is used as the learning framework to classify the images based on features extracted from Gabor Filter and OGPCI. This study uses three face databases, Faces96, Faces94 from University of Essex to evaluate the performance of algorithm and the Georgia Tech Face Database from Centre for Signal and Image Processing at Georgia Institute of Technology. All these databases vary according to different head scales, different head positions, different head orientations and different lighting illumination. The database Faces94 is modified to have occluded images to test the efficiency of the proposed hybrid algorithm. The proposed method achieved 100.00% recognition rate for the images with different head scales and different head positions in Faces96 database; 89.60% recognition rate for the images with different head orientations in Georgia Tech Face Database and 98.50% recognition rate for the partially occluded face images in Faces94 database.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

CGC	Complete Gabor Classifier
GFC	Gabor Filter Classifier
LDA	Linear Discriminant Analysis
OGPCI	Oriented Gabor Phase Congruency Image
OOB	Out-of-bag
PBGFC	Phase Based Gabor-Fisher Classifier
PCA	Principle Component Analysis
RGB	Red Green Blue
RF	Random Forest
SFS	Sequential Forward Selection

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Humans are very good at recognizing faces and complex patterns. Study shows that human starts to recognize faces as early as three-month-old (Barrera and Maurer, 1981). Even a passage of time doesn't affect this capability. It would be useful if computers can perform face recognition as robust as human. However, it's a true challenge to build an automated system which models human ability to recognize faces.

Therefore, face recognition had becomes one of the most desirable applications of image processing and analysis for the past few decades. The first automated face recognition can trace back to the 1960's, where Woodrow W. Bledshoe worked on using computers to perform faces recognition (Bledsoe, 1966). Since then, the technologies using face recognition techniques have been evolving throughout the years. Nowadays, there are many different areas applying face recognition technique, such as information security, access management, biometrics, law enforcements, personal security and entertainment.

Even though there are many journals and research paper on different face recognition approaches have been published throughout the years, most of the methods focus only on detecting frontal images with proper illumination. These methods can't achieve high recognition rate when the image used has flaws like occlusion, insufficient lighting condition and slight orientation. These are the major challenges in face recognition which need to be considered. Thus, face recognition has incorporates artificial intelligence elements like neural network to make recognition more robust.

1.2 Problem Statement and Motivation

According to (PetaPixel, 2012) Figure 1.1, the most popular online social network, Facebook with over 845 million active users has 250 million photos uploaded per day. It also claimed that Facebook have store more than 100 petabytes of photo and videos (PopPhoto, 2012). Most of the photos uploaded contain people with different faces, so Facebook has implemented auto-tagging feature which applies face recognition technology to differentiate unique human images (TheFacebookBlog, 2011). This highlights the importance of face recognition in image-sharing community.

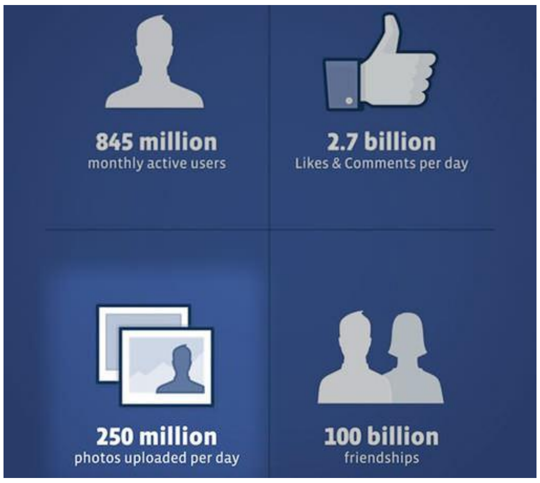


Figure 1.1: Facebook Statistics Data (PetaPixel, 2012)

Furthermore, face recognition helps to improve biometric security features by encoding the human facial appearance as a password of unlocking personal property like credit card, door access, mobile phone and laptop. By using this technology, it is virtually impossible for the hackers to steal one's "password".

Face recognition can be applied in indexing or retrieving video data based on the appearances of particular persons, which will be useful for reporters, moviegoers and forensic scientist. It will speed up the process of locating the particular persons in a video, instead of going through the video manually which are lengthy and tiring job.

All the area of fields stated has shown the importance of face recognition in different area of fields. However, the real-life challenges in face recognition are yet to be solved. For example, the problem of detecting faces wearing glasses faced by (Han et al., 2000) and limitation of lighting condition (under-exposed or over-exposed faces) faced by (Kong and Zhu, 2007). Apart for that, the challenges like face orientation variation, occlusion and extra face features like glasses and cap also affects the recognition rate.

All the applications and problem stated have become a motivation to study and evaluate the face recognition techniques. Besides, it also encourages the search and implementation of suitable face recognition method to overcome the limitations of problems discussed above.

1.3 Aims and Objectives

The aim of this project is to propose a face recognition technique, Complete Gabor Classifier with Random Forest. The proposed method is capable to recognize facial images with different head scales and different head positions, different lighting illumination, different head orientations and partially occluded images. The performance of the proposed method is measured in term of recognition rate on Faces96, Georgia Tech Face Database and Faces94 database. The performance of the proposed method is compared with the existing state-of-the-art algorithms such as PCA, LDA and Gabor-PCA.

1.4 Project Scope

This project will be focusing on developing a method in recognizing faces of different identities from databases. The databases are downloaded from internet for standard benchmarking. The databases contain frontal facial images with different head scales and different head positions, different head orientations, different illumination and partially occluded image.

The performance of proposed algorithm is measured in term of recognition rate. The project is focus on comparing the recognition rate of the proposed method with the available state-of-art algorithms. The speed of recognition is not discussed in this project.

MATLAB R2012b is used as development tool for this project, where it provides necessary toolbox, libraries and functions for image processing module and learning framework.

1.5 Report Outline

Chapter 1 highlighted the background of the topic, the problem statement and motivation behind the project, the aims and objectives, project scope and report outline of each chapter.

Chapter 2 presents the literature review of the fields related to the project. The topics include overview of face recognition system, feature extraction, feature selection and face recognition approaches.

Chapter 3 covers the methodology of proposed face recognition algorithm. The theory of the proposed algorithm is explained at this chapter.

Chapter 4 summarizes the results obtained from this project. The results obtained from different facial images database are shown and discussed here.

Besides, the comparisons of proposed algorithm with state-of-the-art algorithms are shown.

Chapter 5 concludes the entire project and recommendations for future improvement are stated.

CHAPTER 2

LITERATURE REVIEW

2.1 Face Recognition System

In a complete face processing system, the acquired image will undergo several stages as shown in Figure 2.1. Face detection is the first stage of a face recognition system. In the face detection stage, the video captured in frame or still images is taken as an input. The input images need to be pre-processed to reduce computational effort. For example, the colour image (3-dimensional data) is converted to grayscale format (2-dimensional data) or resize the high resolution picture to low resolution. This will improve the speed of processing and reduce the memory space.

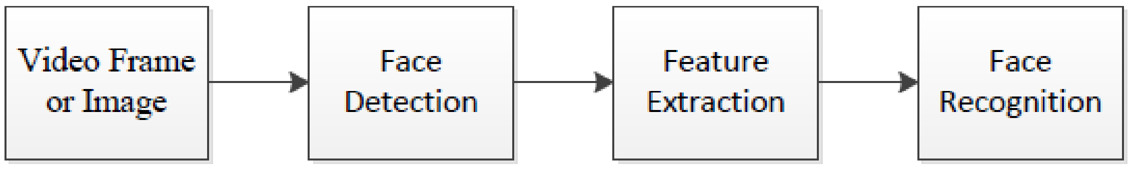


Figure 2.1: The Three Stage of a Face Recognition System

In the face detection stage, the image is scanned to extract the face region from the background. By doing so, unnecessary information is removed and only the important data are retained for further processing. Removing the unnecessary information helps to lower the dimension of image and thus reduces computation times. Unnecessary regions such as car, tree and building will be removed. The retained data known as the region-of-interest should cover most of the important features of the face like eyes, nose and mouth. These features are essential for later recognition process.

The extracted face region is then feed into the feature extraction stage to select the face characteristics that resemble as a face. At this stage, original face image which is high-dimensional is reduced into a set of data known as feature to examine the traits of the data. After the features are generated, the best subset of these features is selected and passes to face recognition stage. The chosen features are used to train the classifier in recognizing faces.

In face recognition stage, the extracted face regions are classified according to the classes assigned. The distinctive features selected from previous stage are used to differentiate the characteristics of each class members. The system is trained using learning method to classify the face regions correctly. At the end of the stage, the face region is assigned to a label based on the training result. The rate of recognition is evaluated to measure the performance of face classification algorithm.

2.2 Feature Extraction

Human has no difficulty in recognizing faces, which seems to be an automated process in brains. For example, humans are able to recognize people they know even with glasses or caps. Human find no problem in recognizing people when they have added facial feature like moustache, beard, make up and expressions. These processes seem trivial to human, but they present a challenge for the computer to mimic this ability. The main problem of face recognition is to extract information from picture. The process of extracting face information is known as feature extraction.

By referring to Figure 2.1, after the face detection stage is the feature extraction stage. The goal of feature extraction is to reduce the training facial image data which are high-dimensional to a set of valuable information known as features for investigation. The extracted features are normally represented in subspace after transformation from the original image. After that, the distinctive features are selected to match with the input feature set. Redundant or non-relevant features are rejected during the process. Figure 2.2 illustrates the block diagram of feature extraction process (Belle, 2008).

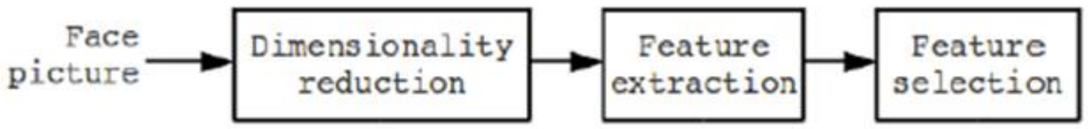


Figure 2.2: Feature Extraction Process

From the Figure 2.2, the feature extraction process involves dimensionality reduction, feature extraction and feature selection. Dimensionality reduction is the process of reducing the high dimensional training data (original image) to a set of low dimensional data known as features.

In the feature extraction stage, a filter is applied on the original image, the output of the filter is known as feature. The best subset from these created features is selected in the feature selection stage. In this stage, the important features are kept while the redundant features are discarded.

2.2.1 Feature Extraction Method

Viola and Jones (2001) used Haar basic functions as face feature extraction. Haar wavelet representation was first adopted in object detection by Papageorgiou et al. (1998). Basically, Haar filter calculates feature regions by adding up all the pixel value defined by the rectangular region. Viola and Jones used three kinds of rectangle features as shown in Figure 2.3 (Viola and Jones, 2004).

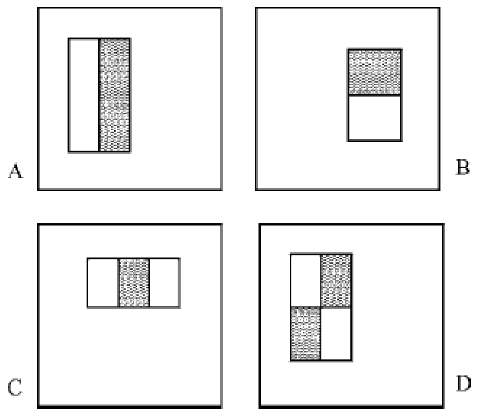


Figure 2.3: Example of Rectangle Features in a Detection Sub-window

In Figure 2.3, the value of two-rectangle features (A) and (B) is the difference between sum of pixel of grey and white rectangle. For three-rectangle features (C), it computes the difference between sum of pixels of two white rectangles and grey rectangle. Four-rectangle feature (D) computes difference between diagonal pairs of white and grey rectangles. The feature sets computed are large, consisting 160000 features for 24 × 24 pixels. The formulas of computing Haar filter output for different rectangles feature are as follow:

$$Two\ Rectangles: W_1 - B_1 \tag{2.1}$$

$$Three\ Rectangles: W_1 + W_2 - B \tag{2.2}$$

$$Four\ Rectangles: W_1 + W_2 - (B_1 + B_2) \tag{2.3}$$

Where W_N is the sum of pixels within white region is box and B_N is the sum of pixels within black region box. N is the number of white/black region box within the rectangles feature.

Gabor filter is another feature extraction method. Gabor feature representation consists of wavelet coefficients for different scales and orientations, making these features robust to rotation, translation, distortion and scaling. It was first used by Lades et al. (1993) for face recognition based on Dynamic Link Architecture which exploit high grey-level information and shape information. Vitomir and Nikola (2010) applies Gabor filter to represent facial image into 5 different scales and 8 different orientations which gives 40 different output images as shown in Figure 2.4.

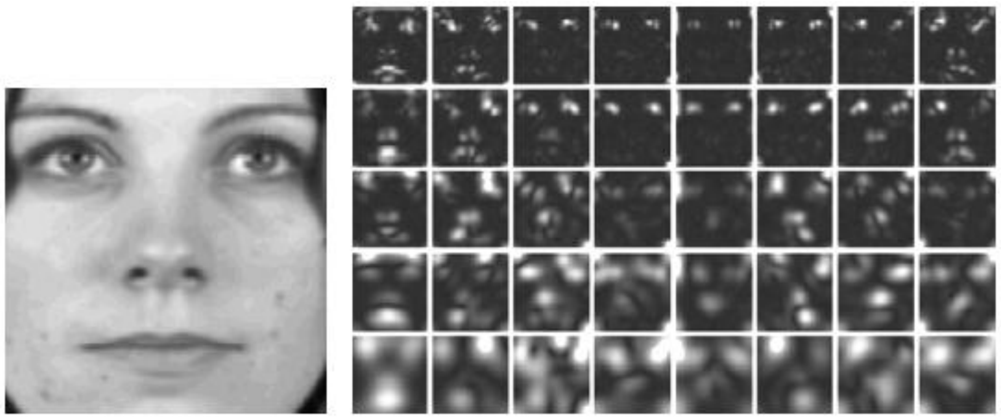


Figure 2.4: Gabor Representation of Face of 8 × 5

Existing face recognition technique used only Gabor magnitude information (Liu, 2004). The phase information of the Gabor filter is ignored due to its difficulty in extracting stable and discriminative features from the phase responses (Zhang et al., 2007). To overcome this instability, Baochang et al. (2008) formed the histogram of Gabor phase patterns using local binary method. This histogram is used as face descriptors and the results proven that Gabor phase responses do exhibit some face information. Baochang et al. method does not represent the face directly as it only represents the phase information in histogram and select the best face descriptors. Struc et al. (2008) implemented Oriented Gabor Phase Congruency Image (OGPCI) to extract stable phase information features. In Struc et al. paper, the face is represented with phase based Gabor-Fisher classifier (PBGFC), the results shows that PBGFC has lesser computational burden (lesser features) when compared to Gabor filter. Vitomir and Nikola (2010) combined both Gabor magnitude and phase information with LDA as the feature selection method to form a complete Gabor-Fisher classifier. The result shows that it outperformed several face recognition techniques like PCA and LDA. Figure 2.5 shows the facial images represented in OGPCI form of different scales.

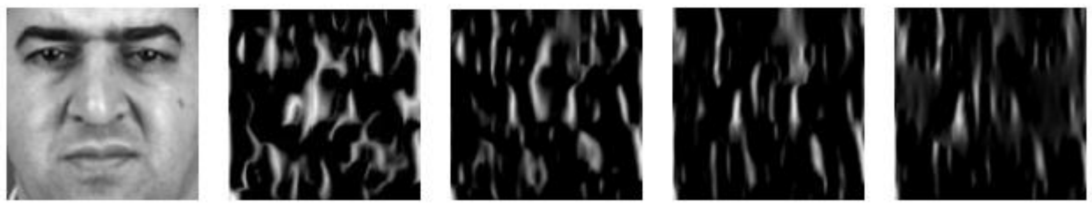


Figure 2.5 : Example of OGPCIs images

2.2.2 Feature Selection Method

After the features are computed, it is necessary to select the best subset of features that gives smallest classification error. The method of classification will determines the classification error. Smaller classification error will yields better classification result. The direct method of feature selection is by examining every features subset and chooses the one that fulfils the defined criteria. However, this method is computational ineffective when there are huge number of subsets present. Recent paper has highlighted a list of effective method in feature selection (Marqués, 2010). The list includes exhaustive search, Sequential Forward Selection (SFS). Apart from that, Genetic algorithm can be applied in feature selection (Yang and Honavar, 1998).

Feature selection can be embedded in learning algorithm of face recognition. For example, Viola and Jones (2004) used Adaboost to select best features among 160000 Haar extracted features and forms a strong classifier for face detection.

2.3 Face Recognition

Face recognition stage is closely related to face classification. It is a process to classify the face region into correct class label. Since face recognition has more than 2 classes, it is name as multi-class problem. Normally, it will be mentioned as *P* class problem if there are *P* numbers of subjects. Face recognition is a challenging task as images of same person can be in different illuminations and pose. In addition, added structures like cap, beard and moustache will complicate the recognition rate. Several approaches have been implemented to tackle such challenges. These approaches can be classified into three categories: appearance-based, feature-based and learning-based approaches.

2.3.1 Appearance-based Approach

Appearance-based approaches use statistical methods to derive image (high-dimensional) into a feature space (low-dimensional). Linear appearance approaches is commonly used, it performs linear dimension reduction to project the image into face vectors. The feature of the face can be represented using the projection weights. Examples of these methods are principal component analysis (PCA) and linear discriminant analysis (LDA)

PCA is one of the earlier approaches in facial recognition by using eigenfaces. Turk and Pentland (1991) applied Euclidean distance to measure and classify the input face images. First, the input image is projected into eigenspace and represented in a feature vector. Then, the input feature vector (weight vector) is compared with the training image feature vector to find the difference between them. The distance of these two vectors are calculated using Euclidean distance measure. The distance measures the similarity of input image to the training images. The lower the distance measure, the higher the matching score between the input image and the training image. Principle component analysis performs well in frontal images, but gives bad results when the images used have pose and illumination variation.

To alleviate these issues, Belhumeur et al. (1997) apply linear discriminant analysis (LDA) in face recognition. Unlike PCA, LDA models the complete set of face training in scatter matrix. The scatter matrix includes the difference between-class (interpersonal) and within-class (intrapersonal). The same classes are group tightly together, while different classes are scatter as far away as possible from each other, maximizing the ratio of between-class to within class scatter. Similar to PCA similarity measurement, once the eigenvectors are obtained, any distance measurement methods like Euclidean distance can be applied. However, LDA requires a large training sample for good classification which is not suitable for face recognition. A combination of PCA and LDA is adopted because its ability to reduce the training size (Zhao et al., 1998). In this method, PCA is applied first to reduce feature dimension before LDA projection. Figure 2.6 shows different face projections of LDA, PCA+LDA and PCA (Zhao et al., 2003).

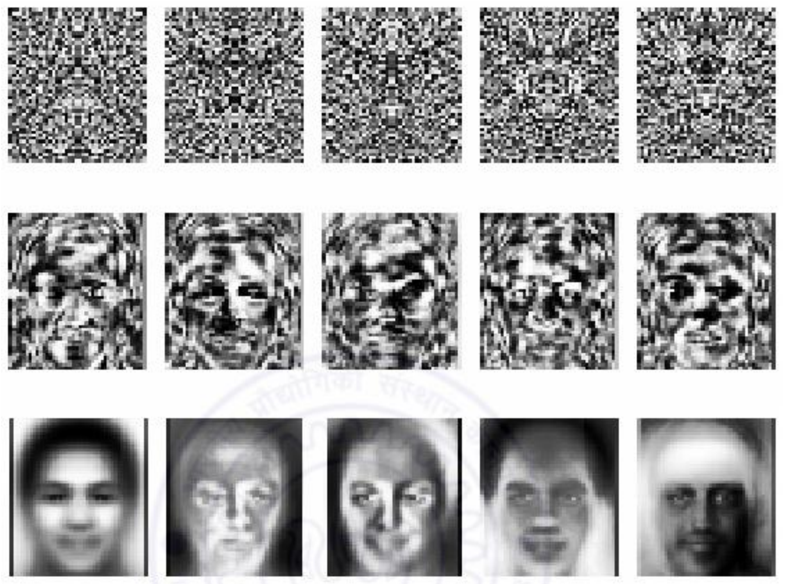


Figure 2.6: Different Linear Projection (from top): LDA, PCA+LDA and PCA

2.3.2 Feature-based Approach

Feature-based approaches extract local facial features like eyes, nose and mouth, then obtain their location and statistical information such as geometrical shape, intensity values, distance and angle between each features points. These face feature values (width, length, shape and position) are then feed into structural classifier for identification. Figure 2.7 illustrates how the face features are measured. The chin shape measurement is based on mouth and chin edge location (Brunelli and Poggio, 1993). Feature-based approaches allow flexible deformation at the feature points, which is good for face image with pose variation.

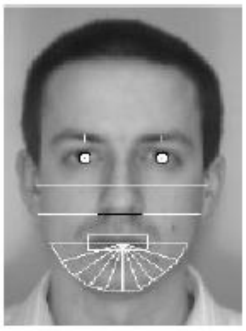


Figure 2.7: Geometric Features-based Approach in Face Recognition

Sharif et al. (2011) applied Gabor wavelet to locate the features of face. Gabor filter was described as feature extraction method at Chapter 2.2.1. In Sharif et al method, Gabor filter is applied on face image and the peak intensity point of the filtered image is obtained. The extracted features points are then used to compare with the database. The database contains feature points information of the training images. The distance of feature points between the input image and training image are computed to measure the similarity. Figure 2.8 shows the results of the feature points are extracted from Gabor image.

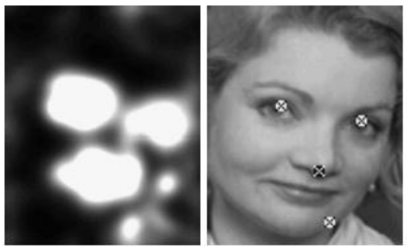


Figure 2.8: Left – Gabor Image, Right – Feature Points Extracted from the Peak of Gabor Image

2.3.3 Learning-based Approaches

Learning-based approach face recognition applies machine learning algorithms in classifying faces. Artificial neural network is one of the popular learning algorithms used in face recognition. Neural network was first demonstrated by Kohonen (1988) to recognize aligned and normalized faces. Since then, many methods based on neural network have been proposed, like combining Gabor filter with neural network (Bhuiyan and Liu, 2007).

Besides neural network method, ensemble learning recently emerged as a prominent candidate in classification technique by combining all the classifiers to give a final output (Bauer and Kohavi, 1999). Random forest is one of the ensemble learning methods developed by Breiman (2001). Breiman used bagging method to reduce variance and avoid overfitting (Breiman, 1996). Ho (1998) combined Breiman's bagging idea and random feature selection to construct a set of decisions trees to take advantages of overfitting solution. Random forest is a forest built from many classifications trees. Each tree output a class and the forest will select the class that have a maximum output votes. The term "random" means each node of trained tree is split using random selection input feature set.

Random forest has been used in many image classifications. For example, Kouzani et al. use random forest to classify image by using image pixel value (Kouzani et al., 2007). However, Kouzani et al. method is sensitive to variations in lighting and expressions. Gabor filter was implement by Ghosal et al. (2009) as it was robust to illumination and expression. By combining Gabor filter and random forest, it formed a robust face classification (Ghosal et al., 2009).

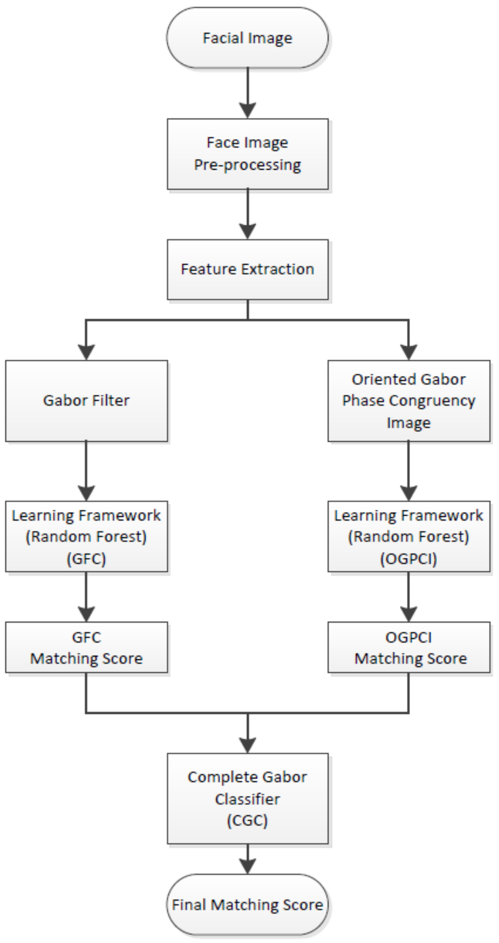
Breiman (2001) has highlighted several advantages of using random forest as classification tool. Random forest does not need any template or model as reference, thus making the learning method straightforward. In addition, this method analyses the image region without extracting geometric property like edges. Random forest also allows classification with occlusion as the region is not sampled in an orderly manner but randomly. Through voting, it eliminates few falsely classified regions. Since random forest consists of many independent trees, parallel processing can be implemented to grow each tree separately.

CHAPTER 3

METHODOLOGY

3.1 Overall Framework

Figure 3.1 illustrates the flowchart of proposed algorithm, Complete Gabor Classifier.



Facial Image
Face Image Pre-processing
Feature Extraction
Gabor Filter
Oriented Gabor Phase Congruency Image
Learning Framework (Random Forest) (GFC)
Learning Framework (Random Forest) (OGPCI)
GFC Matching Score
OGPCI Matching Score
Complete Gabor Classifier (CGC)
Final Matching Score

Figure 3.1: Proposed Algorithm Flowchart

In Figure 3.1, the facial image is pre-processed (Section 3.3.1) before extracting the features. There are two methods of extracting features which are Gabor Filter (Section 3.4.1) and Oriented Gabor Phase Congruency Image (Section 3.4.2). The extracted features from each method are used to obtain the final output class through random forest (Section 3.5). The matching score of each class for Gabor and OGPCI are combined with a fusion parameter to form the final matching score for Complete Gabor Classifier (Section 3.6). The class that has the highest matching score is selected as the final class output. The overall program flow for this project is shown in Figure 3.2.

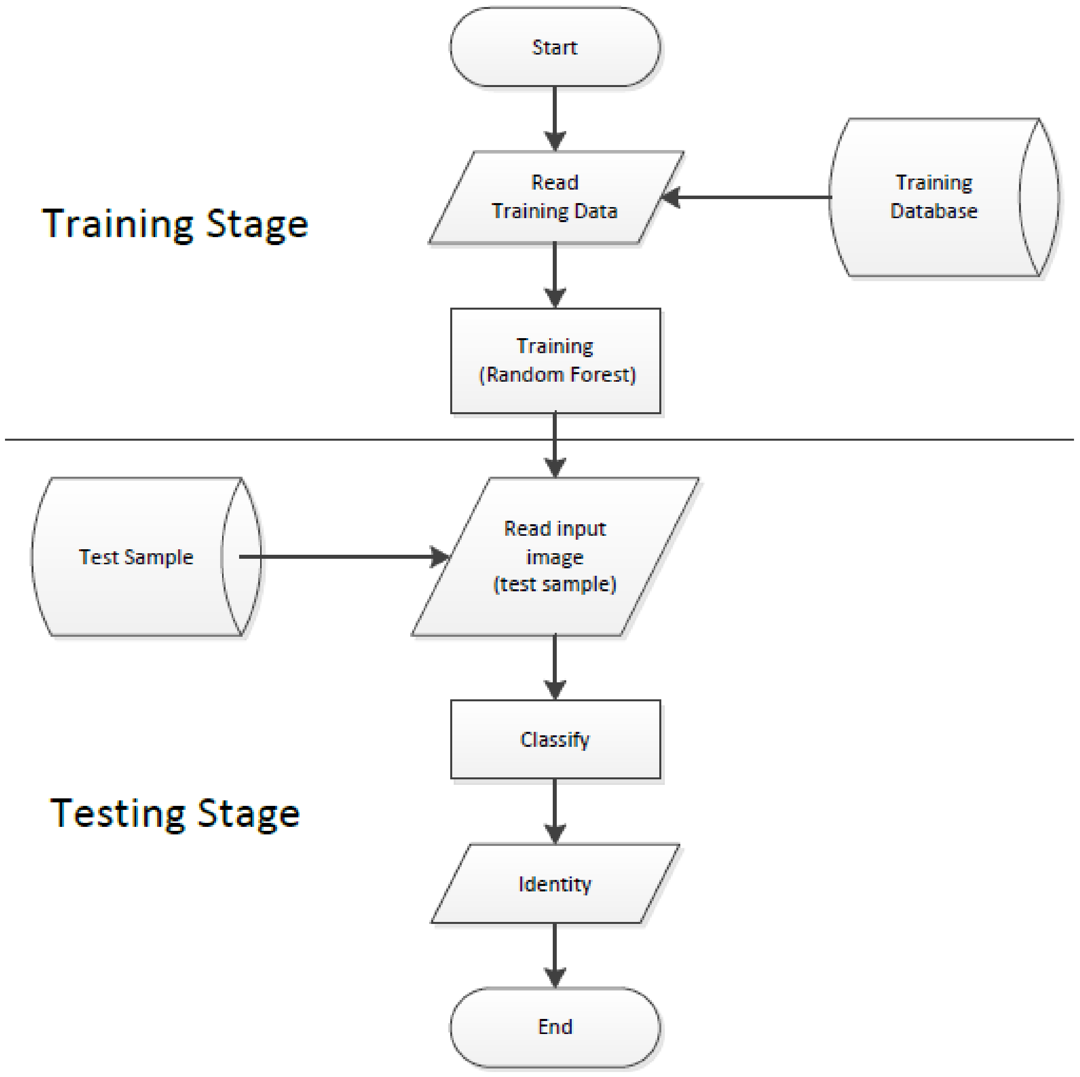


Figure 3.2: Overall Program Framework

The flowchart is divided into two stages, which are training and testing stage. The training stage is a process of training the framework to classify the face sample based on the training data. The testing stage is used to classify the test sample by applying the trained framework.

In the training stage, the important features are extracted from the training sample images (face region). The extracted features are used to grow a random forest. At the end of training stage, the forest is built and the performance of the trained forest is evaluated.

In the testing stage, the test sample (face region) is read by using the same feature extraction method as the trained forest. The extracted features are used to obtain the final output class through the trained random forest. The class of the test image is the maximum class output vote of the random forest.

3.2 Database

In this project, the performance of the algorithms is tested on different set of database. Different database are used to test the recognition performance of feature extraction algorithm and trained random forest. There are three sets of database used which are Faces94, Georgia Tech Face Database and Faces96. Faces94 contains facial images with different head scales and different head positions (Spacek, 2008). Georgia Tech Face Database contains frontal and tilted face images with different expression, orientation and features like wearing glasses (Nefian, 1999). Faces96 contains still frontal images which are used as the training set (Spacek, 2008). Some of the facial images of Faces96 are modified by adding black rectangular box as occlusion. These occluded images are used as the testing set. The details of each database are discussed at Section 4.3, 4.4 and 4.5.

3.3 Face Image Processing

The face image in the database is processed before proceeding to the training process. Figure 3.3 shows the flowchart on how the face image is processed.

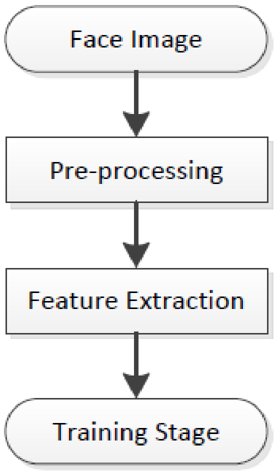


Figure 3.3: Flow Chart of Face Image Processing

The face image is through the pre-processing where RGB colour image is converted to grayscale image. In the training stage, extracted features will be used to train the framework by growing the random forest. The details of each stage will be discussed in the following section.

3.3.1 Pre-processing

Pre-processing stage is a process of optimizing the image quality. Filter operations applied to the image to reduce image details for faster computational speed. Figure 3.4 shows the flowchart of the pre-processing stage.

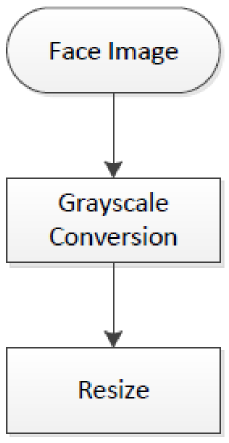


Figure 3.4: Flow Chart of Pre-processing Stage

In the pre-processing stage, RGB colour image is converted to grayscale image. An RGB image has three channels which are red, green and blue, where each channel has 8 bits, making a total of 24 bits; whereas a grayscale image contains only 1 channel, which displays the intensity level in 8-bits. Grayscale conversion helps to reduce the dimensional size of image for faster computation. Figure 3.5 shows a RGB color image converts to a grayscale image.

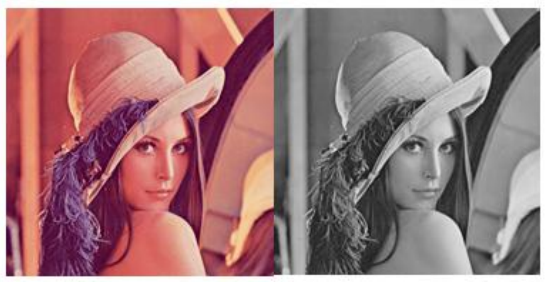


Figure 3.5: RGB to Grayscale Conversion

After the conversion process, the original dimension of the image, 180 × 200 is resized to 64 × 64 pixels. This will reduce the computational speed

3.4 Feature Extraction

In this project, the feature extraction methods used are Gabor filter and oriented Gabor phase congruency image (OGPCI).

3.4.1 Gabor Filter

As discussed in Chapter 2.2.1, Gabor face representation is robust against illumination and facial expression. A Gabor wavelet, $\psi_{u,v}$ is defined as (Vitomir and Nikola, 2010):

$$\psi_{u,v}(x,y) = \frac{{f_u}^2}{\pi\kappa\eta} e^{-(({f_u}^2/\kappa^2){x'}^2 + ({f_u}^2/\eta^2){y'}^2)} e^{j2\pi f_u x'} \tag{3.1}$$

Where,

$x' = x\cos\theta_v + y\sin\theta_v$

$y' = -x\sin\theta_v + y\cos\theta_v,$

$f_u = Gaussian\ center\ frequency = f_{max}/2^{(u/2)}$

$\theta_v = Gaussian\ orientation = v\pi/8.$

$\kappa\ and\ \eta = Ratio\ between\ center\ frequency\ and\ size\ of\ Gaussian\ envelope$

$f_{max} = Maximum\ frequency\ of\ the\ filter$

In this framework, the parameters $\kappa = \eta = \sqrt{2}$ and $f_{max} = 0.2$. These are suggested by Struc et al. (2008). To extract facial features, a filter bank is constructed featuring five scales and eight orientations, where $u \in \{0, \dots, 4\}$ and $v \in \{0, \dots, 7\}$. The filter bank contains real and imaginary part of the Gabor filter. The real part of the filter bank is needed as it is commonly used for facial feature extraction. Figure 3.6 shows the real parts of Gabor filters (40 filters).

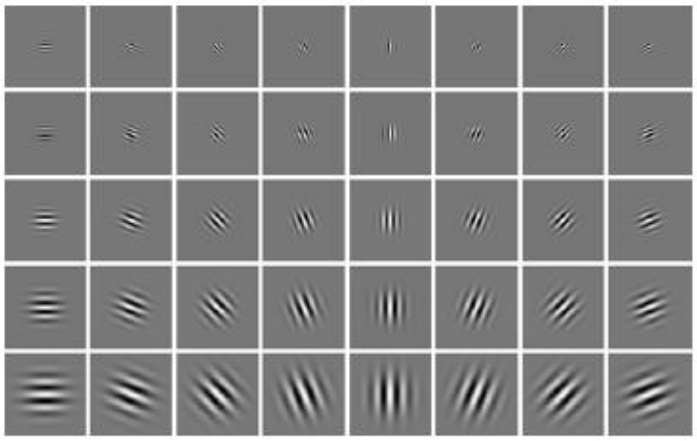


Figure 3.6: The Real Parts of Gabor Filter Bank Constructed in 8 Orientations and 5 Scales

To extract the features from an image, let $I(x, y) \in \mathbb{R}^{a\times b}$ as a grayscale face image with the size of $a \times b$ pixels. Let $\psi_{u,v}(x, y)$ as a Gabor filter at the center frequency f_u and orientation θ_v. The filtering operation is defined as the convolution of image $I(x, y)$ with Gabor filter $\psi_{u,v}(x, y)$:

$$G_{u,v}(x, y) = I(x, y) * \psi_{u,v}(x, y) \tag{3.2}$$

$G_{u,v}(x, y)$ is a complex filtering output that decomposed into real ($E_{u,v}(x, y)$) and imaginary parts ($O_{u,v}(x, y)$):

$$E_{u,v}(x, y) = Re\left[G_{u,v}(x, y)\right] \tag{3.3}$$

$$O_{u,v}(x, y) = Im\left[G_{u,v}(x, y)\right] \tag{3.4}$$

From here, the magnitude information of filtering output is computed:

$$A_{u,v}(x, y) = \sqrt{{E_{u,v}}^2(x, y) + {O_{u,v}}^2(x, y)} \tag{3.5}$$

The image of applying 40 different Gabor magnitude filters are shown at Figure 2.4.

The image features generated is huge in number, as 40 Gabor filters are applied on single image, resulting increase of dimension size by 40 times. After the process of filter, an image of 64 × 64 pixels will becomes 163840 (64 × 64 × 40) diemensional size, which is too computational expensive. To resolve this problem, downsampling using rectangular grid method is implemented as shown in Figure 3.7 (Vitomir and Nikola, 2010). In this method, only the pixels within the rectangular grid are retained, while the remaining pixels are discared, similar with resizing concept. For this project, downsampling factor is set to 128. The following calculation shows the steps of reducing 163840 features to 1000 features:

$$Image = w \times h = 64 \times 64\, pixels$$

$$Downsampling\ factor, \rho = 128$$

$$Width\ downsampling\ factor\ , \rho_w = Height\ downsampling\ factor, \rho_h$$
$$= \sqrt{\rho} = \sqrt{128} = 11.31 \approx 11$$

$$Downsampled\ width, w' = Downsample\ height, h'$$
$$= \frac{w}{\rho_w} = \frac{64}{11} = 5.81 \approx 5$$

$$Downsampled\ Image = w' \times h' = 5 \times 5\, pixels$$

$$Total\ features\ for\ Downsampled\ Image = w' \times h' \times filter$$
$$= 5 \times 5 \times 40 = 1000\, features.$$

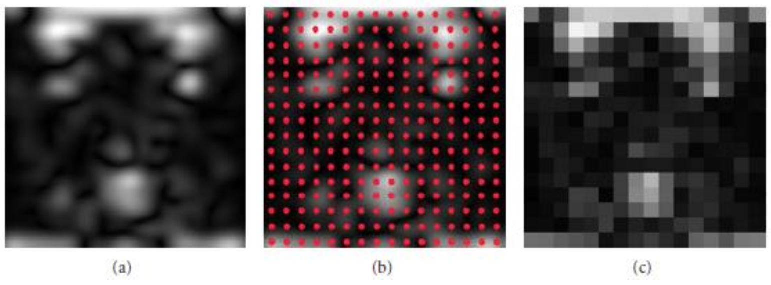


Figure 3.7: Downsampling Process (From Left to Right): (a) Magnitude Response of an Image, (b) Magnitude Response Image with Rectangular Sampling Grid, (c) Image After Downsampled.

3.4.2 Oriented Gabor Phase Congruency Image

The face can be represented in Gabor phase information as discussed in Section 2.2.1. In this project, the features of the image are extracted using oriented Gabor phase congruency image (OGPCI). It is defined as follow (Struc et al., 2008):

$$OGPCI_v(x,y) = \frac{\sum_{u=0}^{p-1} A_{u,v}(x,y)\Delta\Phi_{u,v}(x,y)}{\sum_{u=0}^{p-1}(A_{u,v}(x,y)+\epsilon)} \qquad (3.6)$$

Where $A_{u,v}(x,y)$ is the magnitude response of Gabor filter from Equation 3.5. ϵ is set to 0.0001 as a small constant that prevents divisions with zero. $\Delta\Phi_{u,v}(x,y)$ denotes as a phase deviation measure defined as:

$$\Delta\Phi_{u,v}(x,y) = \cos\left(\phi_{u,v}(x,y) - \bar{\phi}_v(x,y)\right) \\ -\left|\sin\left(\phi_{u,v}(x,y) - \bar{\phi}_v(x,y)\right)\right| \qquad (3.7)$$

Where $\bar{\phi}_v(x,y)$ denotes the mean phase angle at v-th orientation and $\phi_{u,v}(x,y)$ represents the phase angle of Gabor filter can be calculated using the value of real part and imaginary part of Gabor filter (from Equation 3.3 and 3.4):

$$\phi_{u,v}(x,y) = \tan^{-1}\left(\frac{O_{u,v}(x,y)}{E_{u,v}(x,y)}\right) \qquad (3.8)$$

OGPCI computes the phase congruency by summing of p filter scale for each orientation, v. Thus, the feature generated is smaller compared to Gabor magnitude response. For example, taking an input image size of 64 × 64 pixels, using filter bank of 8 orientations × 5 scales, the total features generated are 32768 (64 × 64 × 8) dimensional size. The filter scales, p are added together for each orientation, v, the dimensional size for OGPCI is 5 times lesser than Gabor magnitude information. The number of features generated is still too large for computational, so a downsampling is required to reduce the dimensional size. For Gabor magnitude filter, a downsampled image with factor of 128 can generates 1000 features. OGPCI information is 5 times lesser than Gabor magnitude information. Therefore, the features size generated for OGPCI is 200. The steps of computing OGPCI are as follow (Vitomir and Nikola, 2010):

1. The OGPCI are computed for v orientations of a facial image. (Example of OGPCI images generated for $v = 8$ are present in Figure 3.8).
2. The computed OGPCIs are downsampled by factor of ρ and normalized
3. The downsampled and normalized OGPCIs are concatenated to form augmented Gabor phase congruency feature vector.

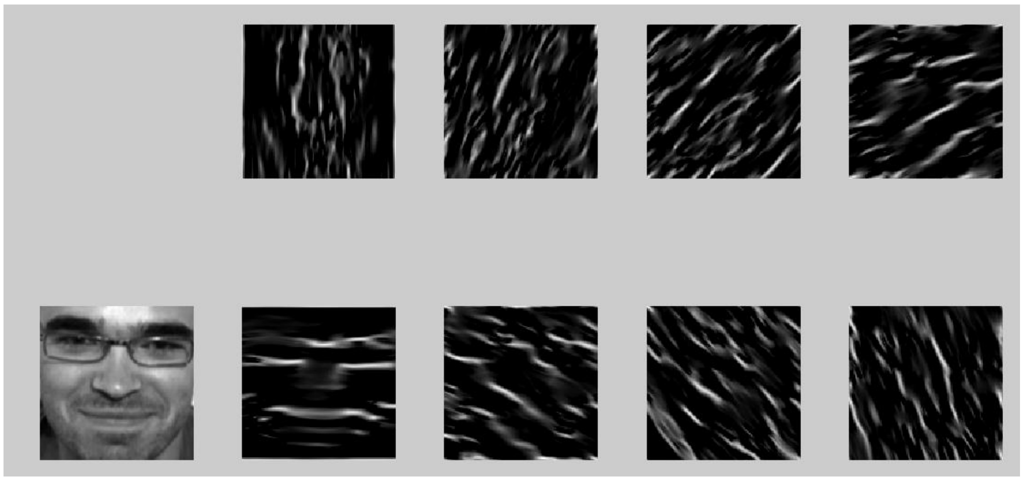


Figure 3.8: Example of all OGPCIs generated for $(v = 8)$ from original images

3.5 Learning Framework

There will be a number of redundant features among the extracted features, thus it is necessary to choose the important features. Random forest is used to evaluate the features importance and select the top important features for face recognition.

3.5.1 Random Forest

Random forest is chosen as it offers many advantages as highlighted at Chapter 2.3.3. Unlike standard decision trees, each node in random forest is split using randomly selected features instead of best features. The selection of a random subset of features has solved the data overfitting problem as proposed by Tim Ho (Ho, 1998). Figure 3.9 shows the example of a random forest.

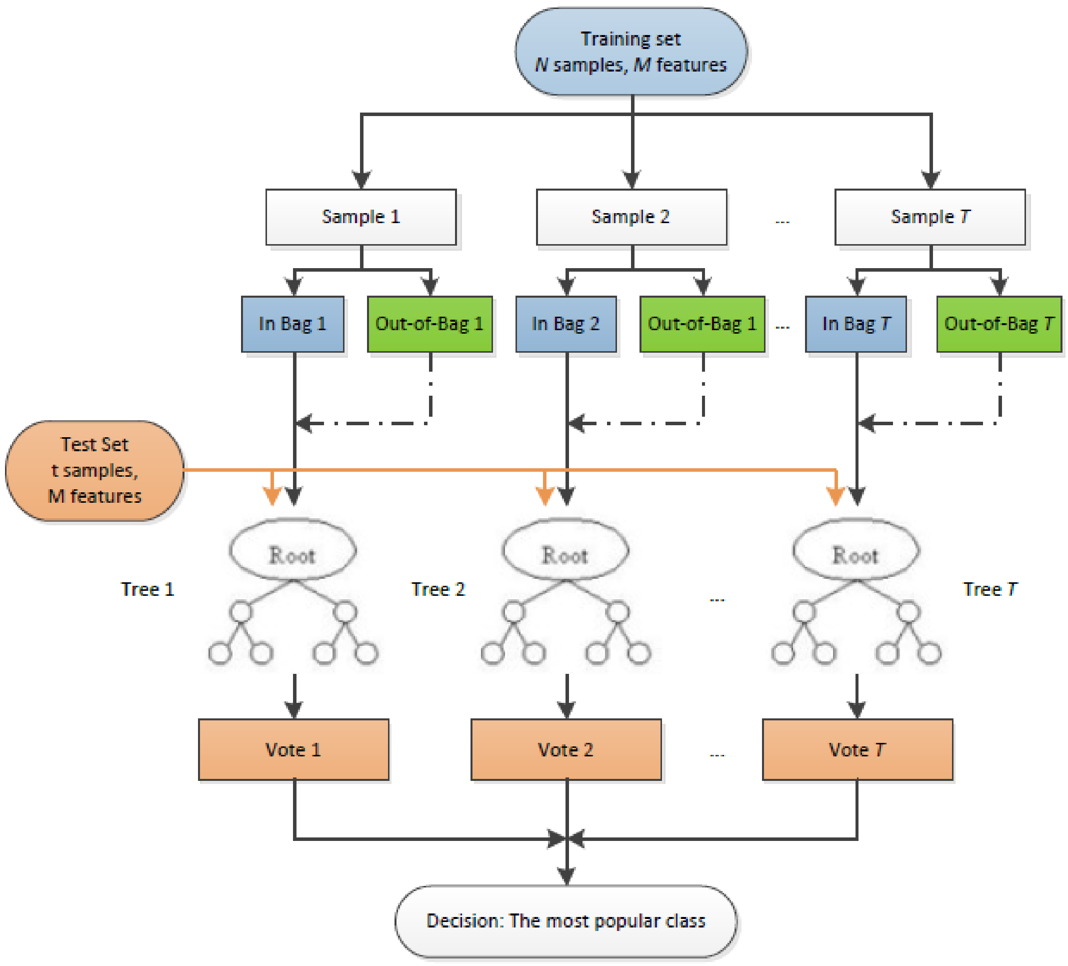


Figure 3.9: Random Forest

Random forest is constructed by *T* classification trees, where *T* is the total number of tree. In order to classify a test sample, the input vectors of test image are evaluated on each tree in the forest. Each tree gives a classification result which represent as a “vote”. The forest chooses the class having the most votes as the final classification output.

In this project, each tree is grown as follows:

1. Let the number of training sample be N, and the number of features be M.
2. Choose n times with replacement from N training cases (two-third), in-bag sample. The remaining (one-third), out-of-bag sample to estimate the error of tree.
3. At each tree decision node, a number m of features is chosen randomly from M and calculate the best split decision among the m variables. The number of m should be lesser than M.
4. Each tree is grown without pruning, to largest size possible.

The tree is grown using two-third of training sample (n), while the remaining one-third ($N - n$) is left out. This remaining sample are known as OOB (out-of-bag) samples are to obtain the value of m. Breiman (2001) stated that the forest error rate depends on two things:

1. The correlation between any two trees in the forest. Increasing the correlation will increases the forest error rate
2. The strength of each tree. Increasing the strength will decrease the forest error rate.

Increasing the value of potential predictors (m) increases the correlation and strength, vice versa. Hence, it is important to find the optimum value of m to balance these opposing effects. To do this, OOB error rate is used to tune m to achieve optimum value. The OOB sample is run through the finished tree to get the classification output. The OOB error is calculated by using the number of misclassified samples, averaged over all cases. The value of m can be adjusted (increasing or decreasing) to minimize the OOB error. It is suggested to begin the m value with square root of the total numbers of predictors ($m = \sqrt{M}$), and search for optimal value with respect to OOB error (Breiman, 2001). Figure 3.10 shows an example of decision tree grown.

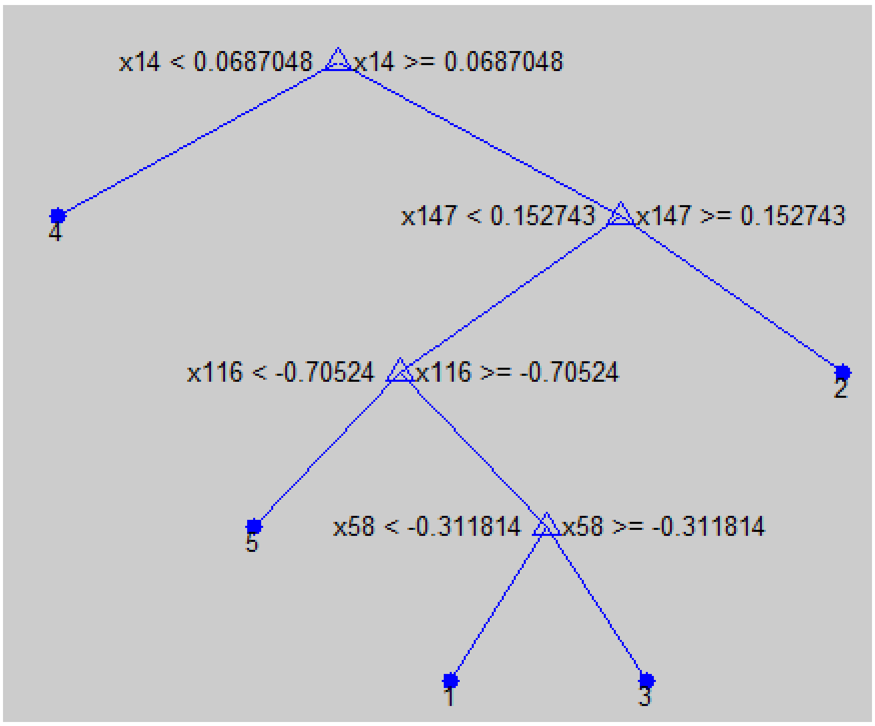


Figure 3.10: Example of a Decision Tree Grown

In Figure 3.10, the random decision tree is used to classify the data into 5 classes: "1", "2", "3", "4" and "5". In this example, there are 160 possible features to select in the pool. At each decision node, random selection of 13 features ($\sqrt{160} \approx 13$) is selected from the pool, and chooses the best split decision. The decision tree starts with a topmost node called root node with variable name "x14", the best variable selected among the 13 features. The root node is split into left and right node by comparing the input feature value with a threshold value, 0.0687048. The threshold value for each split node is obtained using Gini's diversity index. If it is smaller than the threshold value, it will proceed to left node; else it will proceed to right node. Note that at the left node, it stops further splitting as this node only contains 1 class (class "4"). The stop growing node is called leaf node. On the other side, the right node is containing more than 1 class ("1", "2", "3" and "5"). Thus, further splitting is required until it reaches leaf node, the node that contains 1 class output. This decision trees contains four levels of node depth, the first node (x14) has successfully classify class "4", the second level node (x147) classify class "2", then third level node (x116) classify class "5" and finally the fourth level node classify class "1" and class "3". This is an example of a simple random decision tree used to classify 5 classes. A collection of this random decision tree will form a forest which called random forest.

The tree is grown in maximum size (no pruning) to keep the bias low and prevent overfitting. As the number of trees increases, the generalization error converges to a limit (Breiman, 2001). Breiman stated that growing up to 500 trees has presented a promising result.

3.5.2 Feature Importance Selection

After the trees in the forest are grown, the most discriminant features subset is to remove the redundant features. The selected features are used to regrow the trees. Figure 3.11 shows how the variable importance of each feature is computed.

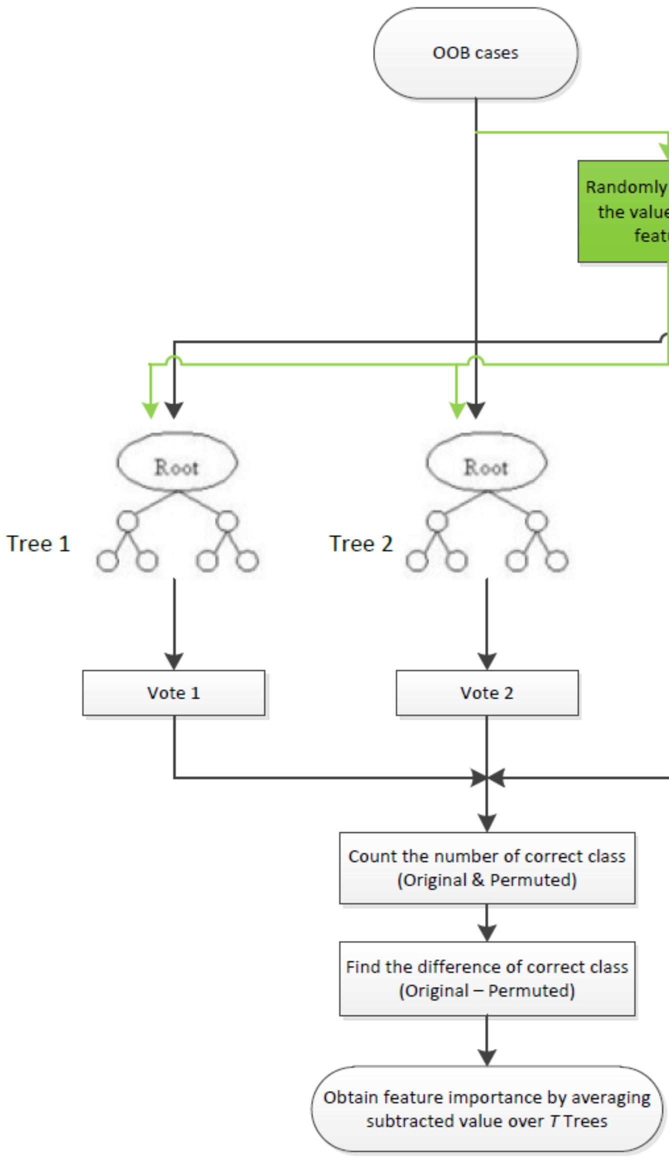

OOB cases
Randomly permutes the value of *m*-th features
Root
Tree 1
Tree 2
...
Tree *T*
Vote 1
Vote 2
Vote *T*
Count the number of correct class (Original & Permuted)
Find the difference of correct class (Original – Permuted)
Obtain feature importance by averaging subtracted value over *T* Trees

Figure 3.11: Flow Chart of Computing Feature Importance

To estimate the importance of *m*-th features (Liaw and Wiener, 2002):

1. For each grown tree, take the OOB cases and go through the tree. Count the number of votes for correct class.
2. Randomly permutes the values of *m*-th features in OOB cases.
3. Apply the tree again to the OOB cases with the permuted values and count the number of correct class.
4. Subtract the number of correct class votes for permuted OOB cases from the correctly classified class of unaltered OOB cases.

The feature importance is defined as the average of this subtracted value over all the trees in the forest. Figure 3.12 illustrates a bar chart showing the feature importance value for 30 features. The higher the feature importance value, the higher the discriminant scores of the feature.

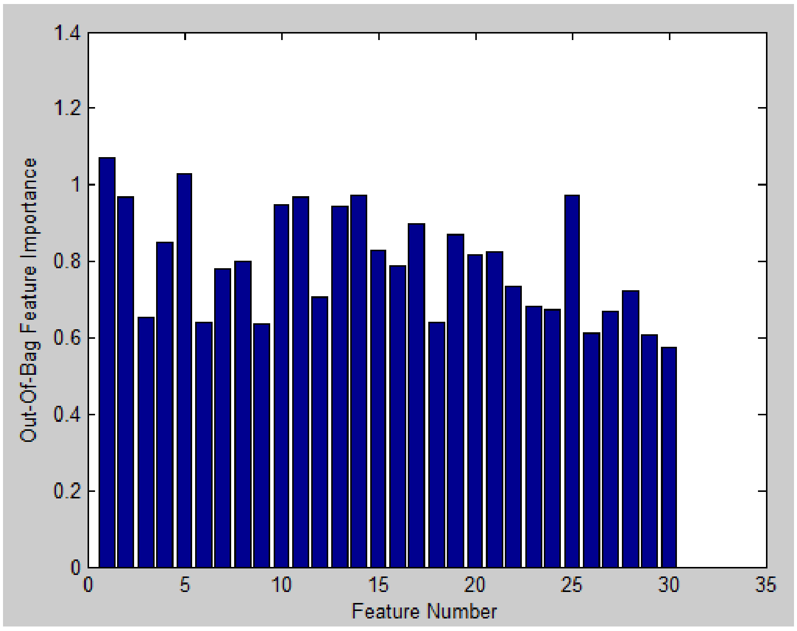


Figure 3.12: Variables Importance for 30 Features

The features are sorted in descending order, from most importance to least importance. Then, regrow the random forest using the best features and compare with the random forest built using the full features.

3.6 Complete Gabor Classifier

Gabor filter computes magnitude and phase information. Most of the existing techniques of face recognition are using Gabor magnitude information. The phase-based Gabor filter can be used in extracting face features (Struc et al., 2008). Combining between magnitude and phase information of Gabor filter have proved to create a robust face classification as discussed by Vitomir and Nikola (2010). The authors use linear discriminant analysis (LDA) as feature selection technique to form

Gabor-Fisher and Phase Based Gabor-Fisher classifier. The fusion of both classifier shows a promising result.

In this project, the feature selection is computed using random forest. After the forest of Gabor magnitude and phase information are grown, the matching score for both techniques are obtained. By combining matching score of Gabor filter classier, δ_{GFC} and Oriented Gabor Phase Congruency Image, δ_{OGPCI}, a final matching score, complete Gabor Classifier δ_{CGC} is computed as follow:

$$\delta_{CGC} = (1-\gamma)\delta_{GFC} + \gamma\delta_{OGPCI} \tag{3.9}$$

Where $\gamma \in [0,1]$, denotes the fusion parameter that control the relative importance of the two matching score. When $\gamma = 0$, the CGC technique turns into GFC technique; When $\gamma = 1$, CGC turns into OGPCI method. Note that γ value should be optimized to achieve the best recognition performance. To select the best performance, the γ value tested in the range of 0 to 1, and increases with the step size of 0.1 and selects the γ value that gives the best matching score. The overall block diagram for Gabor Complete Classifier is illustrated in Figure 3.13.

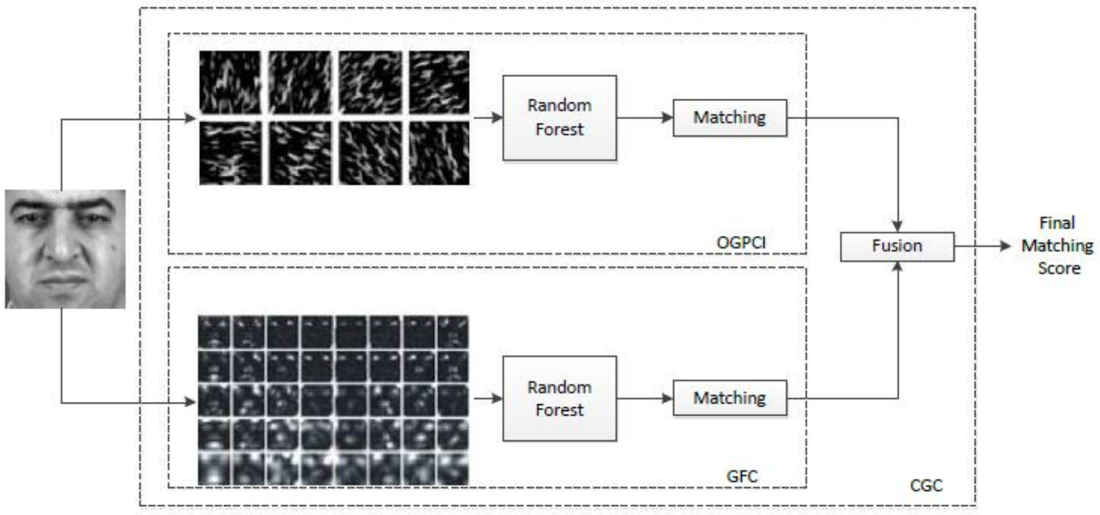


Figure 3.13: Block Diagram of the Complete Gabor Classifier

3.7 Software Architecture

The entire code is written in MATLAB, running using MATLAB R2012b. Matlab is chosen as it provides wide range of engineering tools like Image Processing Toolbox and Statistics Toolbox which are useful in this project. A well-developed library enables fast algorithm prototyping and reduces the time for debugging.

For Gabor and phase feature extraction, external library developed by Vitomir Struc is used (Vitomir, 2012). The library named PhD Tool contains most of the image processing algorithms like PCA, LDA and Gabor. This library is used as the feature extraction of this project is related to Vitomir's Method.

For random forest, Matlab has provided a random forest library named *TreeBagger,* under Statistics Toolbox. *TreeBagger* is a function to construct random forest and the information is stored in *TreeBagger* class. Inside *TreeBagger* class, there are many independent decision trees constructed. The information of each trees is stored as *classregtree* class.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Recognition Rate

The recognition performance is evaluated in term of recognition rate. Recognition rate is the percentage of the number of correct label over the total number of testing image. It is known as rank-1 recognition. The formula of recognition rate is as follow:

$$Recognition\,rate\ (\%) = \frac{The\ Number\ of\ Correct\ Label}{The\ Total\ Number\ of\ Testing\ Image} \times 100\% \qquad (4.1)$$

4.2 Results

This section covers the result of each process for the proposed method, Complete Gabor Classifier with Random Forest. The process starts from the pre-processing stage.

In the pre-processing stage, the test image undergoes grayscale conversion and resizing as shown in **Figure 4.1**. For example, the test image with true class label of 11 is converted from RGB colour format to grayscale format as shown in **Figure 4.1**(b). Then, the test image is resized from 226 × 146 pixels to 64 × 64 pixels as shown in **Figure 4.1**(c) before proceeding to the feature extraction stage.

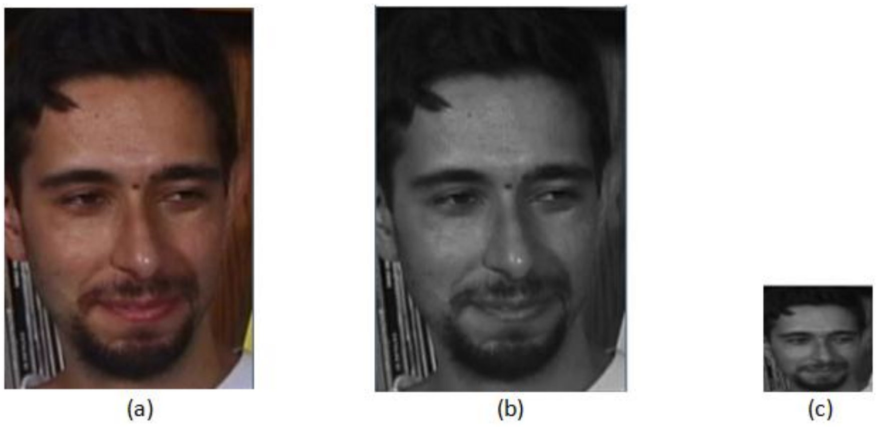


Figure 4.1: (a) Original Test Image. (b) Image After Grayscale Conversion (c) Image Resized to 64 × 64 Pixels.

In the feature extraction stage, Gabor and oriented Gabor phase congruency image filters (OGPCI) are applied on the 64 × 64 pixels grayscale test image. For Gabor Filter, there are total of 40 filtered images output (8 orientations × 5 scales) as shown in Figure 4.2.

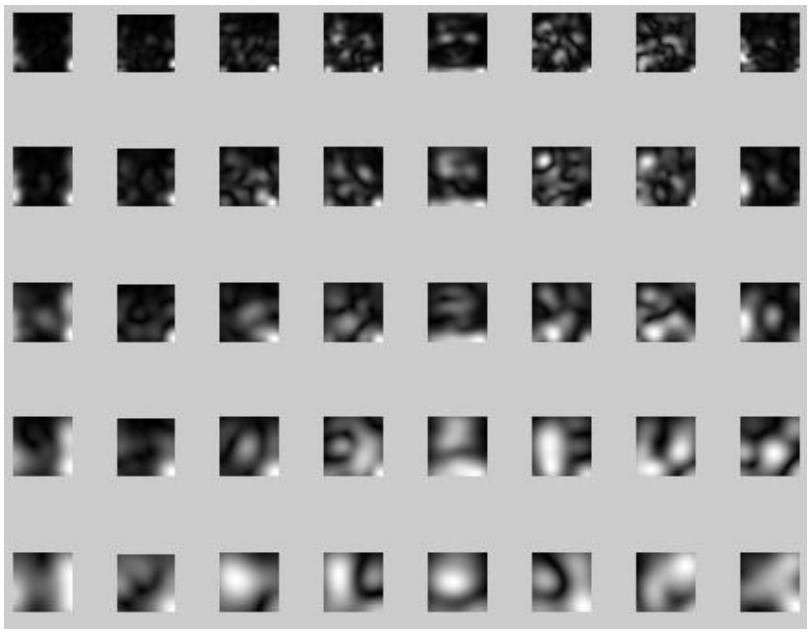


Figure 4.2: Gabor Magnitude Face Representation of Test Image

The features extracted from 40 Gabor magnitudes is huge, a total of 163840 features for 64 × 64 pixels image. Downsampling factor of 128 is used to reduce the filtered images from 64 × 64 pixels to 5 × 5 pixels, with a total of 1000 features. The downsampled Gabor magnitude output images are shown in Figure 4.3

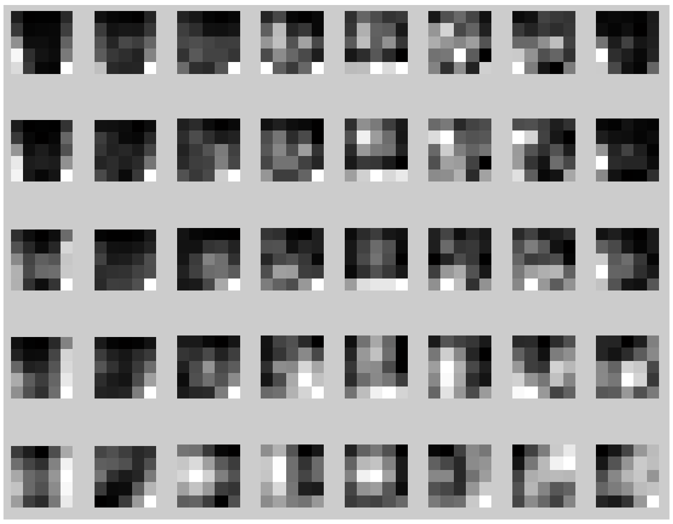


Figure 4.3 Gabor Magnitude Face Representation of Downsampled Test Image

For OGPCI, there are 8 orientations of filtered images. These 8 filtered images generate 32768 features for 64 × 64 pixels image. The 8 OGPCI filtered output images are shown in Figure 4.4.

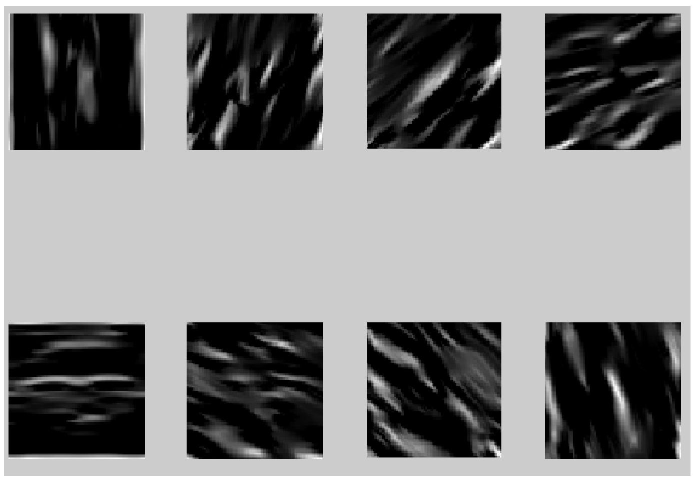


Figure 4.4: OGPCI Face Representation of Test Image

The 8 OGPCI filtered images are downsampled with factor of 128, reducing the number of features from 32768 to 200. The downsampled OGPCI filtered images are shown in Figure 4.5.

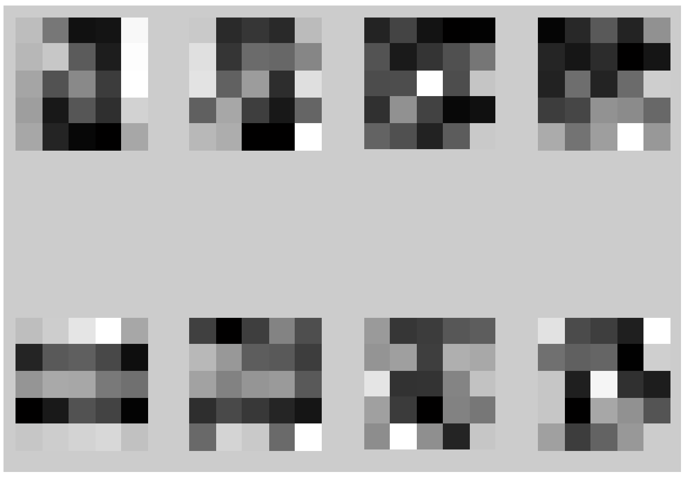


Figure 4.5: OGPCI Face Representation of Downsampled Test Image

The extracted features from Gabor filter and OGPCI are used as the input feature set of trained random forest. The random forest will predict the class of test image based on the input features set. The matching score of each class for Gabor Filter with Random Forest and OGPCI with Random Forest are shown in Figure 4.6 and Figure 4.7.

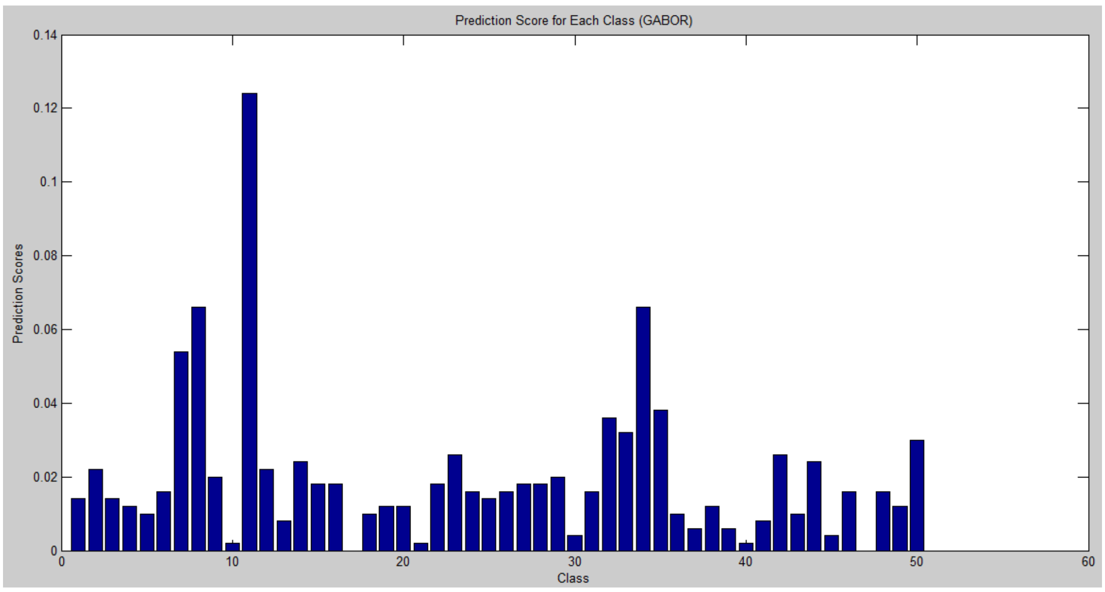


Figure 4.6: The Matching Score for Gabor Filter with Random Forest

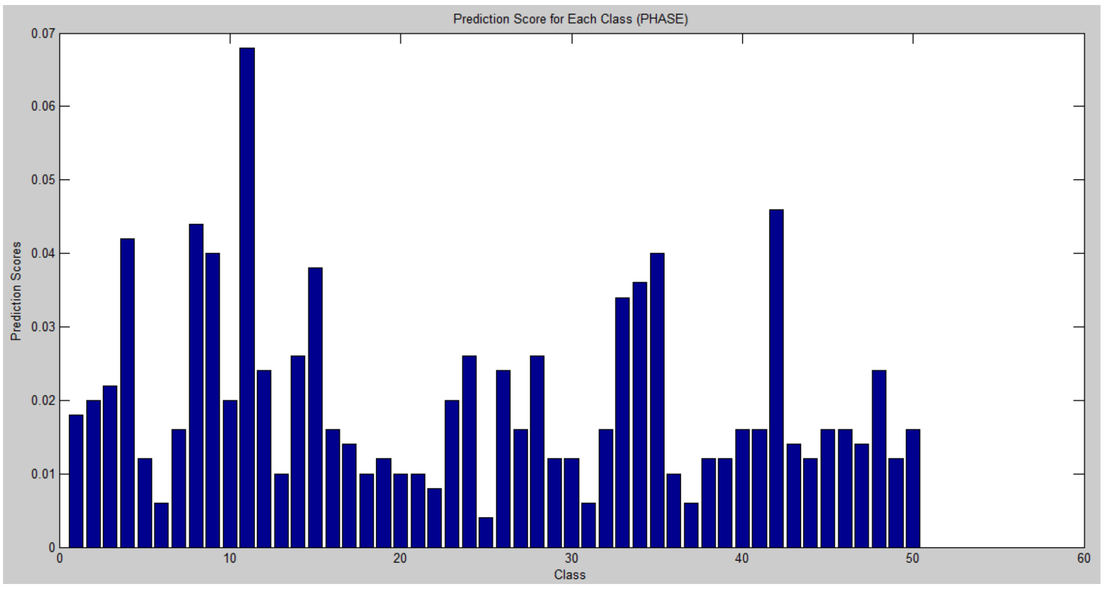


Figure 4.7: The Matching Score for OGPCI with Random Forest

Both Gabor and OGPCI have the highest matching score for class 11 (out of 50 classes), which give correct prediction for the test image. The final matching score of Complete Gabor Classifer (CGC) is obtained by combining both matching score from Gabor and OGPCI with a fusion parameter, $\gamma = 0.3$. The final matching score of CGC is obtained as shown in Figure 4.8.

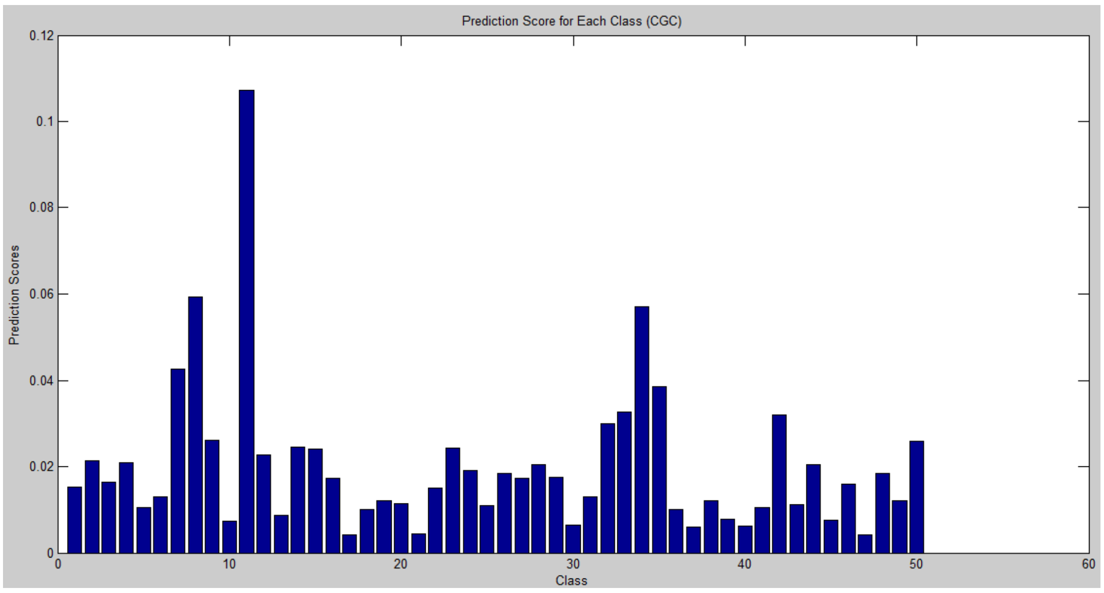


Figure 4.8: The Final Matching Score for CGC with Random Forest

The test image is classified as class 11 because the class 11 has the highest matching score for CGC as shown in the Figure 4.8. The matched image for class 11 named “s11” is displayed as shown in Figure 4.9.

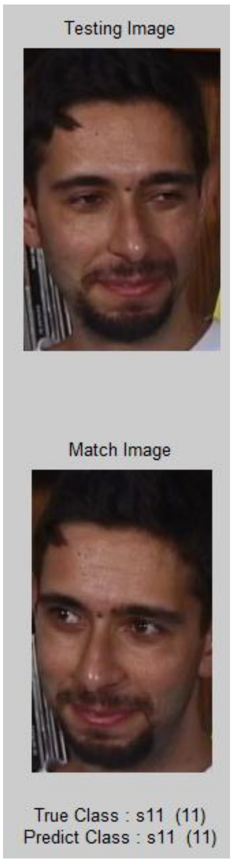


Figure 4.9: Matched Image for Class 11, “s11”

4.3 Faces96 Database

Faces96 database was created by Dr Libor Spacek from University of Essex (Spacek, 2008). Table 4.1 shows the information of Faces96 database.

Table 4.1: Faces96 Database Information

Characteristics	**Faces96**
Number of Individuals	152
Image per Individual	20
Resolution (pixels)	180 × 200
Background	Complex (Glossy Poster)
Head Scale	Large Variation
Head Turn, Tilt and Slant	Minor Variation
Position of Face	Some Translation
Image Lighting Variation	Significant Changes
Expression Variation	Some
Format	24-bit colour JPEG

In this database, 40 individuals are selected randomly to train random forest. There are 20 images for each individual, 15 images are chosen as training set and the remaining 5 images are used as test set. Figure 4.10 shows the samples of Faces96 database images which containing subjects with different head scales, different face positions and head tilting orientations.

In this project, Faces96 is used to evaluate the performance of the Gabor Filter, oriented Gabor phase congruency image (OGPCI) and Complete Gabor Classifier on face image with different head scales and different head positions.

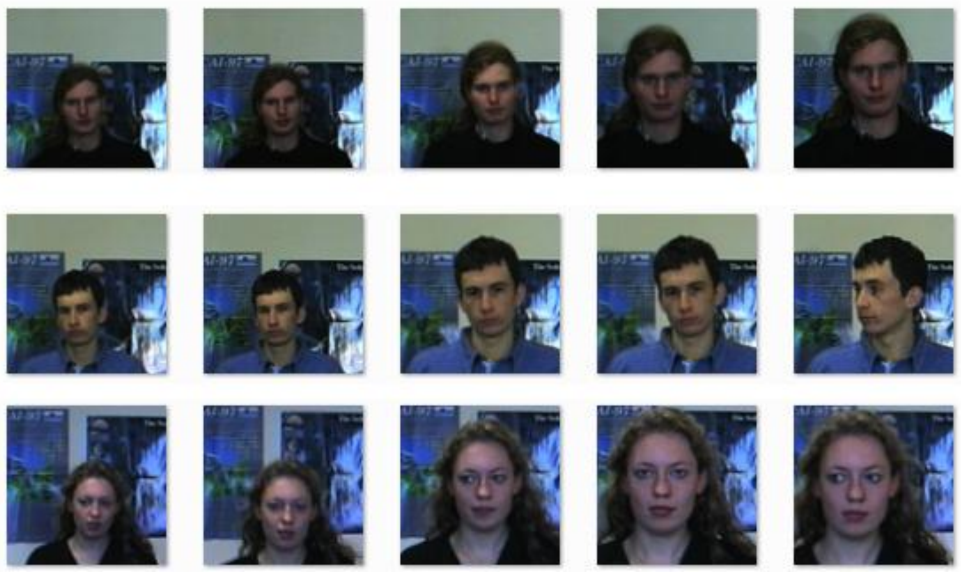


Figure 4.10: Sample Faces96 Database Image

A 64 × 64 pixels image with downsampling factor of 128 will generate 1000 features by using Gabor Filter as discussed in Section 3.4.1. On the other hand, 200 features are generated through oriented Gabor phase congruency image (OGPCI) method as discussed in Section 3.4.2. These features are used to grow trees for random forest in the range of 10 - 50 trees with the interval of 10. Table 4.2 shows the recognition rate for 10 - 50 trees using Gabor and OGPCI features on Faces96 database.

Table 4.2: Gabor (1000 Features) and OGPCI (200 Features) on Faces96 Database

Number of Trees	**Recognition Rate (%)**	
	Gabor	**OGPCI**
10	97.50	94.00
20	99.50	99.00
30	99.50	98.50
40	100.00	98.50
50	100.00	97.50
100	100.00	98.50
200	100.00	98.50

The result shows that both Gabor and OGPCI feature extraction methods achieve above 95% recognition rate with less than 50 trees. Both of the Gabor and OGPCI feature extraction methods can recognize faces with different head scales and different head positions. OGPCI has lower recognition rate compared to Gabor because the features computed for OGPCI (200 features) is lower than Gabor (1000 features). For OGPCI, the recognition rate is fluctuating when the forest is grown using small number of trees (10-50 trees). The recognition rate reached a stable level of 98.50% when more trees are used (100-200 trees). The result is unstable if the number of trees used in random forest is less. Increasing the number of trees will give stable result as more trees give more classification vote.

Using the top 50 to 250 features of Gabor Filter with increment of 50 and the top 10 to 50 features of OGPCI with increment of 10, new forests are built using the only these top features. The results of the new classification performance are shown in Table 4.3 for Gabor Filter features and shown in Table 4.4 for OGPCI features.

Table 4.3: Recognition Rate (%) for Gabor Features on Faces96 Database

Number of Trees	**Number of Features**				
	50	**100**	**150**	**200**	**250**
10	97.50	97.50	98.50	98.50	97.00
20	98.00	99.50	99.50	99.00	99.50
30	98.50	99.00	99.00	99.50	99.00
40	98.50	99.50	99.00	99.50	98.50
50	99.00	99.00	99.00	100.00	99.50

Table 4.4: Recognition Rate (%) for OGPCI Features on Faces96 Database

Number of Trees	**Number of Features**				
	10	**20**	**30**	**40**	**50**
10	88.50	93.50	94.00	94.50	93.50
20	91.50	94.50	97.50	96.50	96.00
30	90.00	96.00	96.00	96.50	97.00
40	90.50	96.50	97.50	98.00	97.00
50	91.00	95.50	97.50	98.00	97.50

The results show that the random forest computed using the best feature has almost the same performance as using all the features generated. For Gabor filter, the best classification result of 100.00% is obtained using 50 trees with 200 features. For OGPCI, the best recognition rate is 98% with 40 trees and 40 features. The recognition rate shows improvement when the number of trees and features increases.

Using the best recognition rate of random forest for both full features (Gabor with 50 trees and OGPCI with 20 trees) and top features (Gabor with 50 trees and 200 features and OGPCI with 40 trees and 40 features), Complete Gabor Classifier is formed by combining the matching score of these two methods with a fusion parameter, γ. The fusion parameter increases with step size of 0.1 from 0.0 to 1.0. The recognition rate for each fusion parameter is tabulate in Table 4.5

Table 4.5: Complete Gabor Classifier on Faces96 Database

Fusion Parameter, γ	**Recognition Rate (%)**	
	Using Best Features	**Using Full Features**
0.0	100.00	100.00
0.1	100.00	100.00
0.2	100.00	100.00
0.3	100.00	100.00
0.4	100.00	100.00
0.5	100.00	100.00
0.6	100.00	100.00
0.7	100.00	100.00
0.8	99.00	100.00
0.9	98.50	99.50
1.0	98.00	99.00

The best result of using Complete Gabor Classifier is 100.00% for both random forest, same as Gabor Filter (γ = 0.0). In Table 4.5, random forest using the best features maintain 100.00% recognition rate for γ = 0.0 to 0.7, while for full features, 100.00% recognition rate is achieved for γ = 0.0 to 0.8. The result shows that the random forest built using the top features has the same performance as using the full features. In Faces96 database, Gabor Filter is sufficient to achieve 100.00% recognition rate for all the test images. Figure 4.11 shows the output recognition results of applying Complete Gabor Classifier on Faces96 test sets.

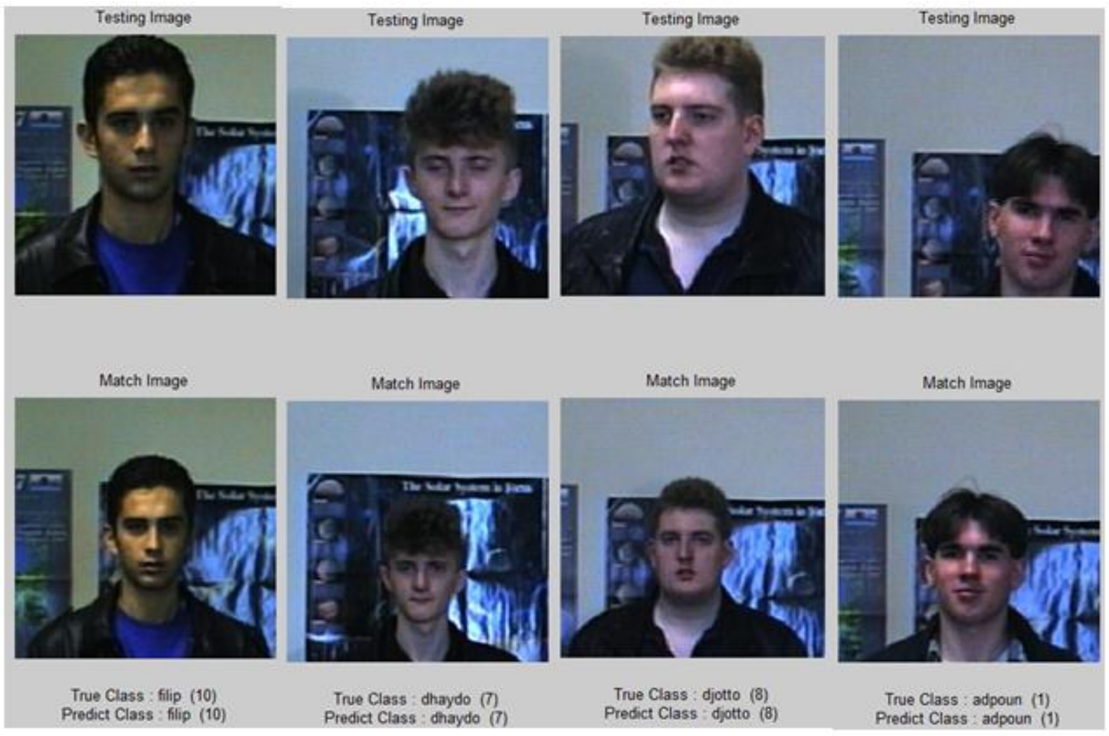


Figure 4.11: Faces96 Facial Image Testing Result

Complete Gabor Classifier can recognize image with different head scales and different head positions as shown in Figure 4.11. Taking the second column of Figure 4.11 (from the left), CGC classifies the testing image correctly even when the image has larger head scales and is positioning toward right. Therefore, Complete Gabor Classifier performs well for the facial image with different head scales and different head positions.

4.4 Georgia Tech Face Database

Georgia Tech Face Database contains frontal face image taken in between 06/01/1999 to 11/15/1999 at Centre for Signal and Image Processing at Georgia Institute of Technology (Nefian, 1999). The information of Georgia Tech Face Database is shown in Table 4.6.

Table 4.6: Georgia Tech Face Database Information

Characteristics	**Faces96**
Number of Individuals	50
Image per Individual	15
Resolution (pixels)	150 × 150
Head Turn, Tilt and Slant	Major Variation
Image Lighting Variation	Significant Changes
Expression Variation	Some
Format	24-bit colour JPEG

In this experiment, all the individual (50 subjects) are chosen to train the random forest. Each individual contains 15 images, 10 of them are chosen as the training set, while the remaining 5 are selected as the testing set. Figure 4.12 shows the samples of Georgia Tech face database. This database contains frontal and tilted faces with different face expressions, different lighting conditions and different face features. This experiment is to evaluate the performance of the Gabor Filter, oriented Gabor phase congruency image (OGPCI) and Complete Gabor Classifier on different orientation, expression and lighting illumination of frontal face.

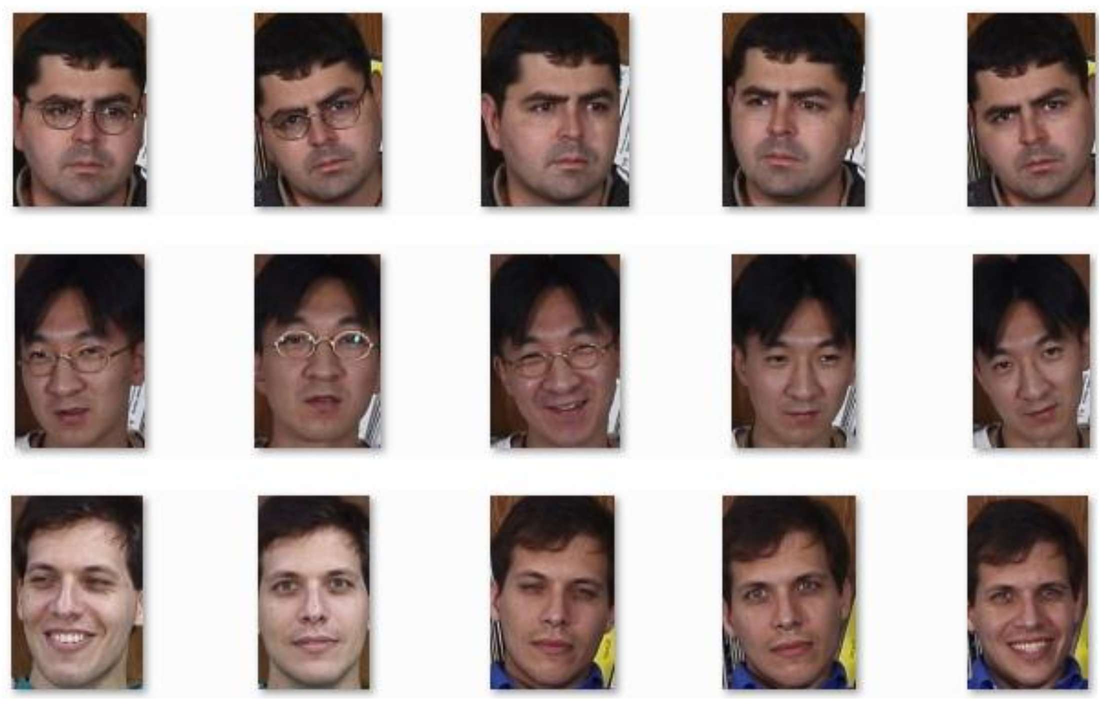


Figure 4.12: Sample of Georgia Tech Face Database Images

Random forest is grown from 100 to 500 trees with increment of 100. The recognition rate of using Gabor filter and OGPCI as feature extraction methods on Georgia Tech face database is shown in Table 4.7.

Table 4.7: Gabor (1000 Features) and OGPCI (200 Features) on Georgia Tech Face Database

Number of Trees	Recognition Rate (%)	
	Gabor	OGPCI
100	80.80	63.60
200	86.80	67.20
300	84.80	70.40
400	86.40	70.40
500	86.80	71.60
1000	87.20	72.00

In Table 4.7, the difference between the recognition rate for the chosen trees of 500 and the chosen trees of 1000 is 0.40%. Comparatively, the difference between the chosen trees of 400 and the chosen trees of 500 is 0.40%. Therefore, in order to reduce the consumption of the memory and to have an efficient recognition rate, trees of 500 are chosen.

The result shows that Gabor feature extraction method has better recognition rate than OGPCI because Gabor contains more features than OGPCI. For Georgia Tech face database, it requires more trees (around 500 trees) to achieve good recognition rate (> 80%) compared to Faces96 database. The reason is that the Georgia Tech face database has less samples (10 samples) compared to Faces96 (15 samples). Thus, there are not enough training samples to form train the random forest. Besides, the number of features used to train the random forest is small, 1000 features for Gabor filter and 200 features for OGPCI. Increasing the facial image resolution or reducing downsampling factor will increases the number of extracted features used in training .

The forest is regrow using the top 50 to 250 features for Gabor filter and top 10 to 50 features for OGPCI. The classification results are shown in Table 4.8 and Table 4.9.

Table 4.8: Recognition Rate (%) for Gabor Features on Georgia Tech Face Database

Number of Trees	**Number of Features**				
	50	**100**	**150**	**200**	**250**
100	75.60	76.40	80.00	81.60	84.00
200	77.60	79.20	82.40	82.40	84.80
300	78.80	82.00	82.80	84.40	85.20
400	79.20	83.60	83.60	84.40	84.80
500	78.40	83.20	84.40	84.00	86.40

Table 4.9: Recognition Rate (%) for OGPCI Features on Georgia Tech Face Database

Number of Trees	**Number of Features**				
	10	**20**	**30**	**40**	**50**
100	59.60	56.40	61.60	61.20	61.20
200	65.60	63.60	68.40	66.40	66.00
300	64.80	66.00	70.00	67.20	68.40
400	66.80	68.40	68.00	68.40	70.00
500	66.80	68.40	67.60	69.60	70.40

The results show that the random forest computed using the best feature has slightly lower performance than using all the features on Georgia Tech Face Database. For Gabor filter, the best classification result is 86.40% is obtained using 500 trees with 250 features compared to 86.80% (Table 4.7) when all the features are used. For OGPCI, the best recognition rate is 70.40% when computed using 500 trees with 50 features, slightly lower than 71.60% which used all OGPCI features as in Table 4.7. This concludes that the random forest built using top features has almost the same performance as using full features with only 0.40% lesser in recognition rate. The features used are reduced to 25% of full features but can produces almost similar performance using full features.

The best matching score from both random forest computed using best features, Gabor Filter (500 trees with 250 features) and OGPCI (500 trees with 50 features) are combined with fusion parameter, γ to form Complete Gabor Classifier (CGC). For random forest grown using full features, the best recognition rate of Gabor Filter (500 trees) and OGPCI (500 trees) are combined to form CGC. The rate of recognition of CGC for each fusion parameter, γ is shown in Table 4.10.

Table 4.10: Complete Gabor Classifier on Georgia Tech Face Database

Fusion Parameter, γ	**Recognition Rate (%)**	
	Using Best Features	**Using Full Features**
0.0	86.40	86.80
0.1	86.80	87.60
0.2	88.40	88.00
0.3	89.20	89.60
0.4	88.40	89.20
0.5	88.40	86.40
0.6	86.80	84.80
0.7	84.00	83.60
0.8	82.40	81.20
0.9	76.00	75.20
1.0	70.40	71.60

As shown in Table 4.10, the best recognition rate of using Complete Gabor Classifier is when γ = 0.3. The recognition rate for best features is 89.20% and the recognition rate for full features is 89.60%. CGC exploits the information of Gabor magnitude and phase information by combining the Gabor filter and OGPCI matching score, so the rate of recognition is increased. This concludes that the hybrid of Gabor and OGPCI has better performance than using single method. Figure 4.13 shows the classification result of using Gabor, OGPCI and Complete Gabor Classifier on Georgia Tech Face Database.

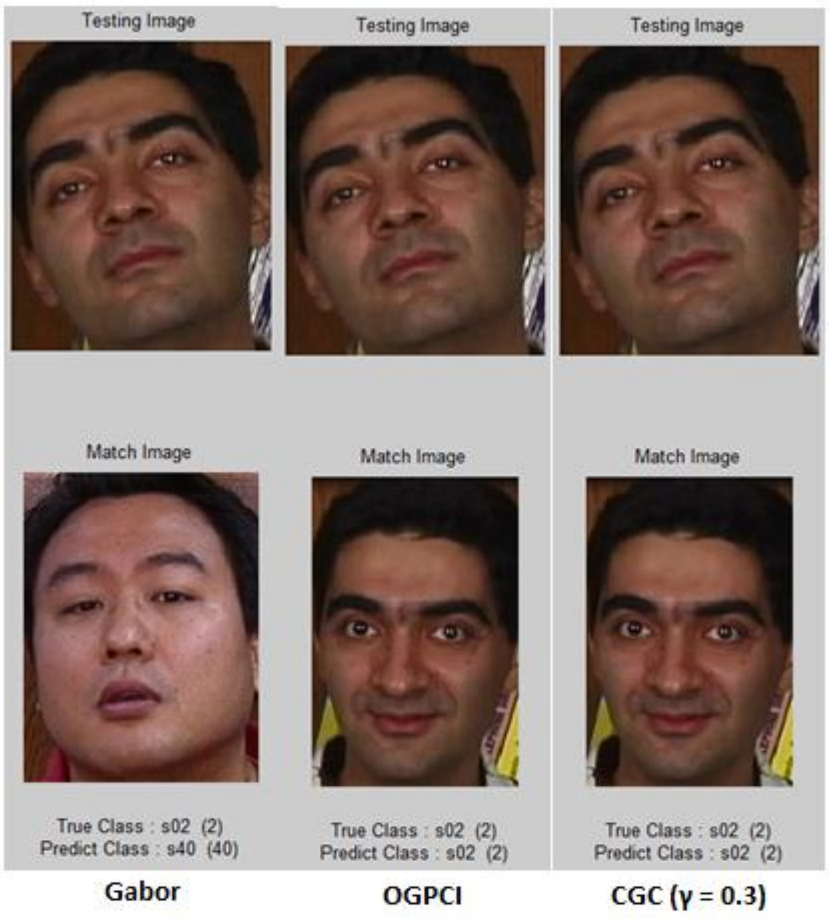


Figure 4.13: The Classification Result from Gabor, OGPCI and CGC on Georgia Tech Face Database (Gabor Misclassify)

Gabor Filter failed to classify the result correctly in the Figure 4.13, but OGPCI gives correct class output. Complete Gabor Classifier (CGC) classifies the test image correctly because CGC is the hybrid of Gabor and OGPCI. CGC exploits OGPCI phase information to give correct classification results. Thus, combining both Gabor and OGPCI improves the face recognition rate. Similarly, CGC also exploits Gabor magnitude to correct OGPCI misclassification. The result is shown at Figure 4.14.

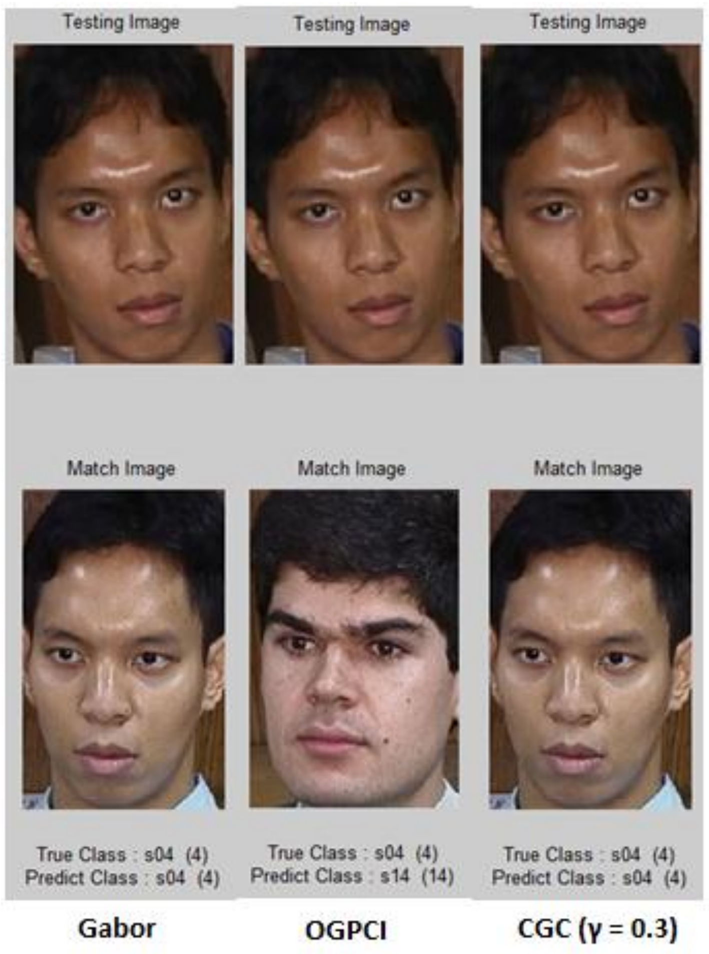


Figure 4.14: The Classification Result from Gabor, OGPCI and CGC on Georgia Tech Face Database (OGPCI Misclassify)

Gabor Filter classifies the test image correctly, whereas OGPCI classify the image wrongly. By combining both Gabor and OGPCI, CGC identifies the test image correctly by exploiting the information of Gabor magnitude to improve the performance. Figure 4.15 shows some of the correct classification results using Complete Gabor Classifier with Random Forest.

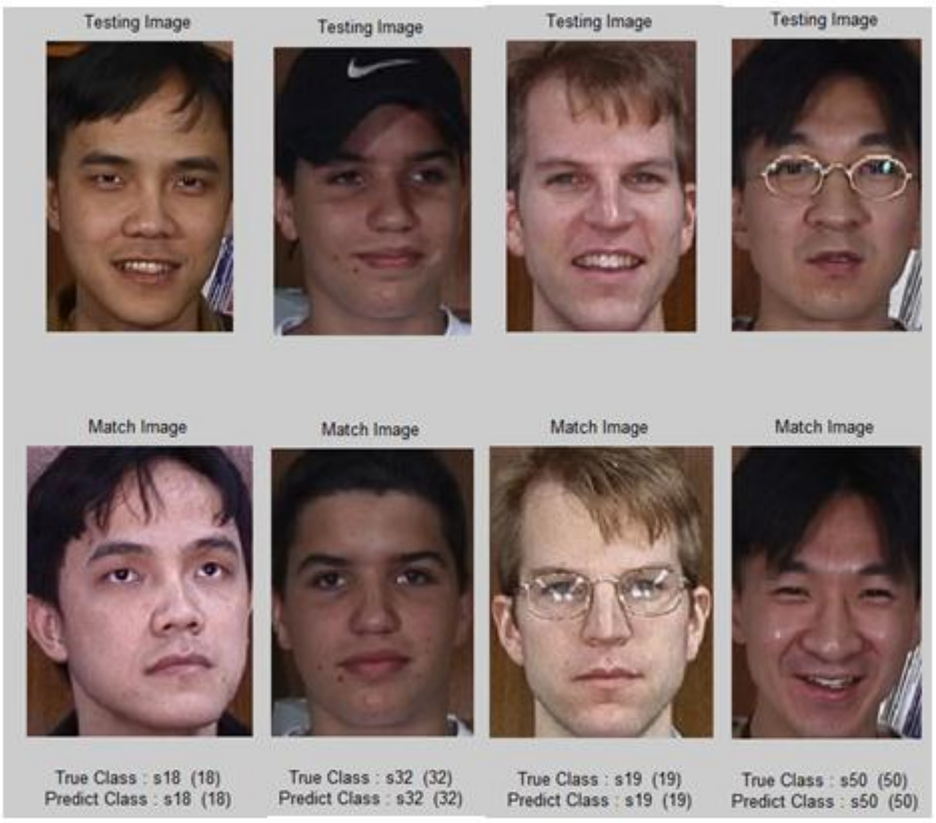


Figure 4.15: Complete Gabor Classifier on Georgia Tech Face Database (Correct Classification)

Complete Gabor Classifier recognizes facial image with different illumination lighting and facial orientation as shown in the first column from the left in Figure 4.15. Besides, CGC performs correctly for the test subjects with extra features like wearing caps and glasses in the second column and fourth column from the left in Figure 4.15. Furthermore, CGC recognizes facial image with different expressions as shown in the third and fourth column.

Therefore, Complete Gabor Classifier with Random Forest can performs well for facial images with different head orientation, different facial expression, different lighting illumination and addition features like glasses. CGC had achieved 89.60% recognition rate when using 500 trees with 1000 Gabor Filter features and 200 OGPCI features. The result can be further improved by the addition of the number of features used in growing random forest.

4.5 Faces94 Database (Occluded)

Faces94 was created by Dr Libor Spacek (2008) from University of Essex, similar with Faces96 database. Table 4.11 shows the information of Faces94 database.

Table 4.11: Faces94 Database Information

Characteristics	**Faces94**
Number of Individuals	153
Image per Individual	20
Resolution (pixels)	180 × 200
Background	Plain Green
Head Scale	None
Head Turn, Tilt and Slant	Very Minor Variation
Position of Face	Minor Translation
Image Lighting Variation	None
Expression Variation	Minor changes
Format	24-bit colour JPEG

For Faces94 database, 40 subjects are chosen randomly for growing random forest. Fifteen out of twenty images per individual are chosen as the training set, while the remaining 5 images as the testing set. For testing set, the images are modified by adding different size of black box as occlusion. The details of each occlusion box used on Faces94 are shown in Table 4.12.

Table 4.12: Details of Occlusion Boxes on Faces94 Database

Properties	**Occluded Box No.**				
	1	**2**	**3**	**4**	**5**
Size (w × h), pixels	100 × 30	100 × 50	100 × 30	100 × 70	40 × 100
Occlusion Percentage (%)	8.33	13.89	8.33	19.44	11.11
Occluded Region	Eyes	Nose + Mouth	Mouth	Forehead	Half face

Faces94 is used for the occlusion test because this database do not have much variation, such as complex background, various light illumination, head tilting which will affects the recognition rate. This experiment is to test the effectiveness of face recognition algorithm on occluded images. Figure 4.16 shows the example of the training set of Faces94 database and Figure 4.17 shows the samples of the occluded testing image set.

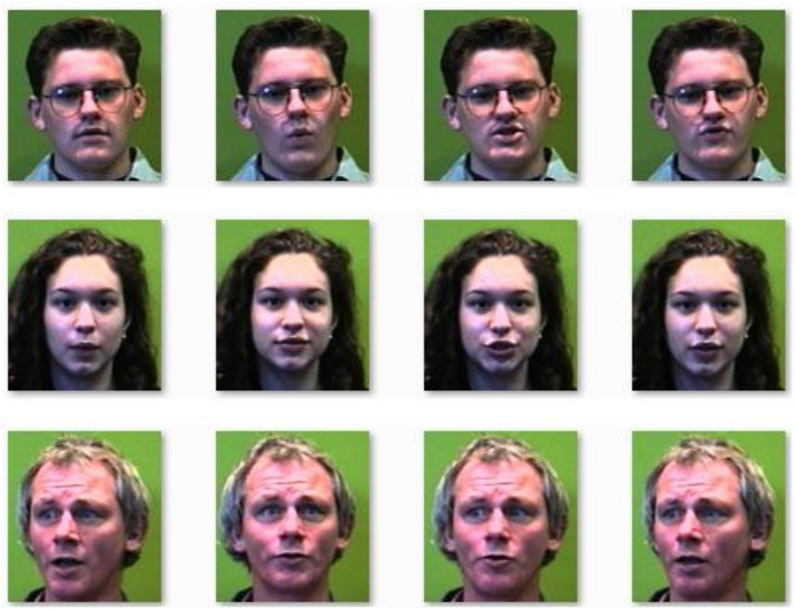


Figure 4.16: Sample Training Set for Faces94 Database Image

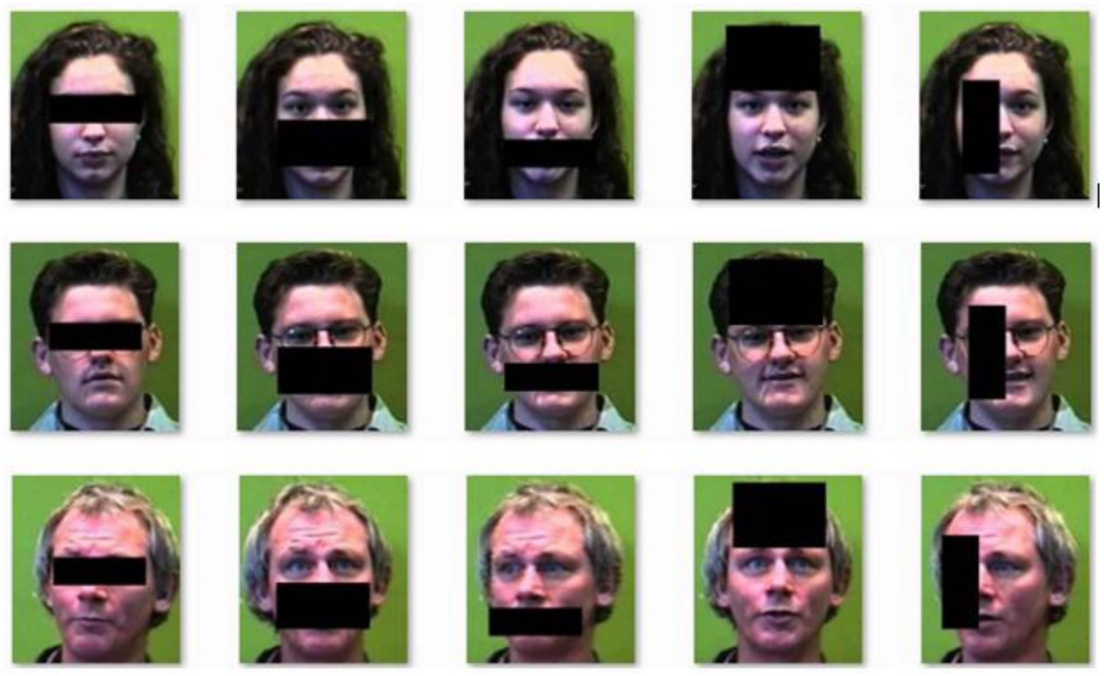


Figure 4.17: Sample Testing Set for Faces94 Database Image (Occluded)

Random forest is grown using Gabor filter and oriented Gabor phase congruency image (OGPCI) feature extraction methods on the training set of Faces94 database. The number of trees to grow is set in the range of 60 to 100 with increment of 10. After the random forest is grown, the testing set (occluded image) is evaluated using the random forest built to obtain the classification performance. The result is shown in Table 4.13.

Table 4.13: Gabor (1000 Features) and OGPCI (200 Features) on Faces94 Database

Number of Trees	**Recognition Rate (%)**	
	Gabor	**OGPCI**
60	96.50	84.00
70	96.00	86.50
80	96.50	88.50
90	96.50	87.50
100	95.00	87.50
500	97.00	89.00

As shown in the Table 4.13, when the trees are grown using 500 trees, the recognition rate does not improves significantly, with 0.50% improvement from the best results obtained from 60 - 100 trees.

Even with occlusion, the best recognition rate of 96.50% is achieved by growing 80 trees for Gabor Filter. On the other hand, OGPCI achieve 88.50% recognition rate with 80 trees. The extracted features for OGPCI is lesser than Gabor filter, thus it has lower recognition rate. The random forest is rebuilt again using the top features to test whether the newly built forest can replicate the performance of random forest using all features. The results are tabulated in Table 4.14 for Gabor Filter and Table 4.15 for OGPCI.

Table 4.14: Recognition Rate for Gabor Features on Faces94 Database

Number of Trees	Number of Features				
	50	100	150	200	250
60	78.50	82.50	87.00	91.00	93.50
70	77.50	83.00	88.00	91.00	93.00
80	79.50	83.00	89.00	90.50	92.50
90	79.50	83.00	89.00	91.50	93.00
100	80.50	84.00	88.50	91.00	93.50

Table 4.15: Recognition Rate for OGPCI Features on Faces94 Database

Number of Trees	Number of Features				
	10	20	30	40	50
60	56.00	67.00	65.50	66.00	72.50
70	56.00	68.00	67.00	66.00	74.00
80	56.50	67.50	67.50	67.00	74.00
90	56.50	67.50	69.00	68.50	74.00
100	55.00	66.50	68.00	69.00	75.50

The result shows that when the random forest is regrow using the top features. The recognition rate is much lower than using all features. The best performance for Gabor filter is 93.50% using 100 trees with 250 features, while 75.50% for OGPCI using 100 trees with 50 features. This is due to the features like eyes, nose or mouth region are occluded, which cause the feature information degraded. Hence, features which are lower importance are needed to classify the facial image correctly. These less important features are not chosen for the newly built forest, so random forest built using best features will has lower recognition rate when compared to random forest built using all features.

The best matching score from both random forest are computed using the top features, Gabor Filter (100 trees with 250 features) and OGPCI (100 trees with 50 features). These combined with fusion parameter, γ to form Complete Gabor Classifier (CGC). For random forest using the full features, Gabor filter with 80 trees and OGPCI with 80 trees are combined to form CGC. The rate of recognition is shown in Table 4.16 for each fusion parameter, γ.

Table 4.16: Complete Gabor Classifier on Faces94 Database

Fusion Parameter, γ	**Recognition Rate (%)**	
	Using Best Features	**Using Full Features**
0.0	93.50	96.50
0.1	94.00	96.50
0.2	94.50	97.00
0.3	94.50	98.50
0.4	94.00	98.50
0.5	92.50	97.50
0.6	88.50	97.50
0.7	86.50	95.50
0.8	81.00	93.00
0.9	78.00	90.50
1.0	75.50	88.50

In the Table 4.16, the best recognition rate of using Complete Gabor Classifier is 94.50% with fusion parameter, γ = 0.3. Complete Gabor Filter used the information of the two matching score which is Gabor filter and oriented Gabor phase congruency image (OGPCI) to form an improved face recognition system. By choosing optimum fusion parameter (γ = 0.3), the face recognition rate is improved. Figure 4.18 shows the matching results of applying CGC on Faces94 database.

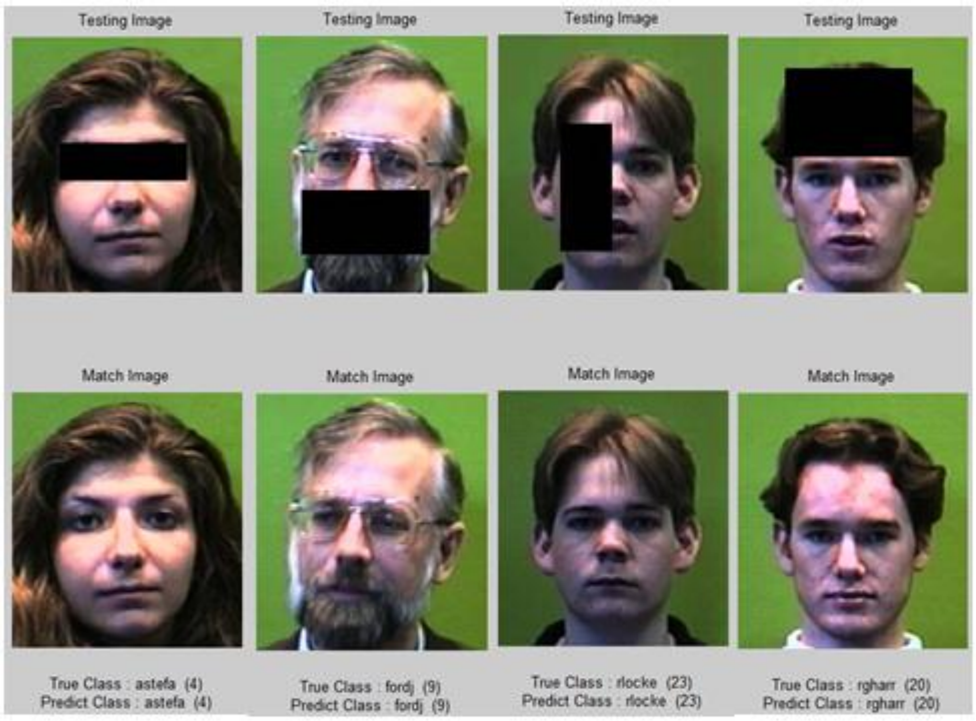


Figure 4.18: Complete Gabor Classifier on Faces94 Database (Correct Classification)

In Figure 4.18, Complete Gabor Classifier successfully recognize facial image which is partially occluded. Even though the important features like eyes, mouth and nose are occluded, the proposed technique still able to classify the occluded test subject correctly.

4.6 Comparisons with State-of-the-art Algorithms

The proposed method, Complete Gabor Classifier with Random Forest (CGC-RF) is compared with the existing state-of-the-art algorithm like Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Gabor-PCA. Table 4.17 presents the recognition rate for each algorithm on different database.

Table 4.17: Comparison of Recognition Rate for Different Algorithm on Different Database

Method	Faces96	Georgia Tech Face Database	Faces94
CGC-RF (all)*	100.00	89.60	98.50
CGC-RF (best)**	100.00	89.20	94.50
PCA	90.50	50.40	81.50
LDA	100.00	65.60	95.00
Gabor-PCA	89.00	55.60	100.00

* CGC-RF (all) is Completer Gabor Classifier with Random Forest using all the extracted features

** CGC-RF (best) is Completer Gabor Classifier with Random Forest using the top extracted features

Table 4.17 includes the comparisons of random forest computed using all features and best features for Gabor, oriented Gabor phase congruency image (OGPCI) and Complete Gabor Classifier (CGC). CGC-RF for all and best features had achieved 100.0% of recognition rates on Faces96, outperforming PCA and Gabor-PCA. For Georgia Tech Face Database, CGC-RF for full features has recognition rates of 89.60%, outperforming PCA, LDA and Gabor-PCA. CGC-RF for full features results in competitive recognition rates of 98.50% on Faces94 database, higher than PCA and LDA methods, but lower than Gabor-PCA. Gabor-PCA achieved 100.0% recognition rates for Faces94, performing well for occluded faces. CGC-RF has lower performance than Gabor-PCA because the number of tree used for CGC-RF grown is less (60 – 100 trees). To have competitive performance, Breiman (2001) suggested to grow the forest using 500 trees. By growing more trees, CGC-RF (the proposed algorithm) will match with the performance of Gabor-PCA. In summary, CGC-RF performs well on all databases which contain images with different head scales, different head positions, different illumination changes and occlusion, making it a robust face recognition system.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, this project proposed a face recognition algorithm called Complete Gabor Classifier with Random Forest. Complete Gabor Classifier exploits the features from magnitude and phase response of Gabor Filter. Random forest is used as the learning framework of proposed method. The performance of the proposed technique is evaluated on three available face databases, namely Faces94, Faces96 and Georgia Tech Face Database. At the end of the results, the proposed techniques shown promising performance on all the dataset and outperform several available face recognition method like PCA, LDA and Gabor-PCA. This concludes that the proposed face recognition techniques has shown robust performance on facial images with different head scales, different head positions, different lighting illumination, facial expression and partially occluded area.

5.2 Recommendations

Training the random forest takes substantial amount of time. The training time increases when the number of features used and tree grow increases. Thus the random forest is grown using only 1000 features for Gabor filter and 200 features for OGPCI. To have better performance result, the number of input features should increases to around 80000 features as suggested by Ghosal et al. (2009).

The proposed algorithm is tested on facial image with different head scales, different head positions, illumination variation and occlusion image. The performance of the Complete Gabor Classifier can be further evaluated by testing on lower resolution images.

Face recognition approaches can be further studies by finding approaches on classifying gender based on facial images, estimating the age of subject, or even categorizing the race of person.

REFERENCES

BAOCHANG, Z., ZONGLI, W. & BINENG, Z. 2008. Kernel learning of histogram of local Gabor phase patterns for face recognition. *EURASIP Journal on Advances in Signal Processing,* 2008.

BARRERA, M. E. & MAURER, D. 1981. Recognition of mother's photographed face by the three-month-old infant. *Child Development***,** 714-716.

BAUER, E. & KOHAVI, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning,* 36**,** 105-139.

BELHUMEUR, P. N., HESPANHA, J. P. & KRIEGMAN, D. J. 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 19**,** 711-720.

BELLE, V. 2008. Detection and Recognition of Human Faces using Random Forests for a Mobile Robot. Online.

BHUIYAN, A. A. & LIU, C. H. On face recognition using gabor filters. Proceedings of world academy of science, engineering and technology, 2007. 51-56.

BLEDSOE, W. W. 1966. The model method in facial recognition. *Panoramic Research Inc., Palo Alto, CA, Rep. PR1,* 15.

BREIMAN, L. 1996. Bagging predictors. *Machine learning,* 24**,** 123-140.

BREIMAN, L. 2001. Random forests. *Machine learning,* 45**,** 5-32.

BRUNELLI, R. & POGGIO, T. 1993. Face recognition: Features versus templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 15**,** 1042-1052.

GHOSAL, V., TIKMANI, P. & GUPTA, P. Face classification using gabor wavelets and random forest. Computer and Robot Vision, 2009. CRV'09. Canadian Conference on, 2009. IEEE, 68-73.

HAN, C. C., MARK LIAO, H. Y., YU, G. J. & CHEN, L. H. 2000. Fast face detection via morphology-based pre-processing. *Pattern Recognition,* 33**,** 1701-1712.

HO, T. K. 1998. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 20**,** 832-844.

KOHONEN, T. 1988. Self-organization and associative memory. *Self-Organization and Associative Memory, 100 figs. XV, 312 pages.. Springer-Verlag Berlin Heidelberg New York. Also Springer Series in Information Sciences, volume 8,* 1.

KONG, W. & ZHU, S. 2007. Multi-face detection based on downsampling and modified subtractive clustering for color images. *Journal of Zhejiang University-Science A,* 8**,** 72-78.

KOUZANI, A., NAHAVANDI, S. & KHOSHMANESH, K. Face classification by a random forest. TENCON 2007-2007 IEEE Region 10 Conference, 2007. IEEE, 1-4.

LADES, M., VORBRUGGEN, J. C., BUHMANN, J., LANGE, J., VON DER MALSBURG, C., WURTZ, R. P. & KONEN, W. 1993. Distortion invariant object recognition in the dynamic link architecture. *Computers, IEEE Transactions on,* 42**,** 300-311.

LIAW, A. & WIENER, M. 2002. Classification and Regression by randomForest. *R news,* 2**,** 18-22.

LIU, C. 2004. Gabor-based kernel PCA with fractional power polynomial models for face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 26**,** 572-581.

MARQUÉS, I. 2010. Face recognition Algorithms. *Universidad Euskal Herriko*.

NEFIAN, A. V. 1999. *Georgia Tech Face Database* [Online]. Available: http://www.anefian.com/research/face_reco.htm [Accessed 1 April 2013].

PAPAGEORGIOU, C. P., OREN, M. & POGGIO, T. A general framework for object detection. Computer Vision, 1998. Sixth International Conference on, 1998. IEEE, 555-562.

PETAPIXEL. 2012. *3000 Photos Are Uploaded Every Second to Facebook* [Online]. Available: http://www.petapixel.com/2012/02/01/3000-photos-are-uploaded-every-second-to-facebook/ [Accessed 22 Nov 2012].

POPPHOTO. 2012. *People Upload an Average of 250 Million Photos Per Day to Facebook* [Online]. Available: http://www.popphoto.com/news/2012/02/people-upload-average-250-million-photos-day-to-facebook [Accessed 22 Nov 2012].

SHARIF, M., KHALID, A., MUDASSAR, R. & MOHSIN, S. 2011. Face Recognition using Gabor Filters. *Journal of Applied Computer Science,* 5.

SPACEK, L. 2008. *Computer Vision Science Research Projects* [Online]. Available: http://cswww.essex.ac.uk/mv/allfaces/index.html [Accessed 21 Jan 2013].

STRUC, V., VESNICER, B. & PAVESIC, N. The phase-based gabor fisher classifier and its application to face recognition under varying illumination conditions. Signal Processing and Communication Systems, 2008. ICSPCS 2008. 2nd International Conference on, 2008. IEEE, 1-6.

THEFACEBOOKBLOG. 2011. *Making Photo Tagging Easier* [Online]. Available: http://www.facebook.com/blog/blog.php?post=467145887130 [Accessed 22 Nov 2012].

TURK, M. & PENTLAND, A. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience,* 3**,** 71-86.

VIOLA, P. & JONES, M. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001. IEEE, I-511-I-518 vol. 1.

VIOLA, P. & JONES, M. J. 2004. Robust real-time face detection. *International journal of computer vision,* 57**,** 137-154.

VITOMIR, Š. 2012. *The PhD Toolbox* [Online]. Available: http://luks.fe.uni-lj.si/sl/osebje/vitomir/face_tools/PhDface/index.html [Accessed 29 Jan 2013].

VITOMIR, Š. & NIKOLA, P. 2010. The complete gabor-fisher classifier for robust face recognition. *EURASIP Journal on Advances in Signal Processing,* 2010.

YANG, J. & HONAVAR, V. 1998. Feature subset selection using a genetic algorithm. *Intelligent Systems and Their Applications, IEEE,* 13**,** 44-49.

ZHANG, B., SHAN, S., CHEN, X. & GAO, W. 2007. Histogram of Gabor phase patterns (HGPP): a novel object representation approach for face recognition. *Image Processing, IEEE Transactions on,* 16**,** 57-68.

ZHAO, W., CHELLAPPA, R. & KRISHNASWAMY, A. Discriminant analysis of principal components for face recognition. Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, 1998. IEEE, 336-341.

ZHAO, W., CHELLAPPA, R., PHILLIPS, P. J. & ROSENFELD, A. 2003. Face recognition: A literature survey. *Acm Computing Surveys (CSUR),* 35**,** 399-458.

APPENDICES

APPENDIX A: Matlab Coding

main.m

```
close all;
% clear all;   % comment it if need debug, clear all will clear
breakpoint


%% Mode Setting

% TRAIN, TRAINBEST, LOAD
mode = 'LOAD';

% Load Mode Setting
% EVAL, TEST
loadmode = 'TEST';

% 1 - Manual pick photo
% 2 - Test all test sample
% 3 - Face Detect
testmode = 1;

%% Main Body

% Feature to-be-added : Feature, Trained Time
disp('========================');
switch mode
    case 'TRAIN'
        disp('Training Mode (Full)');

        methodname = 'GABOR';          % GABOR, PHASE
        psize = [64 64];
        databasename = 'faces96';      % faces94, faces96
        selecttype = 'all';

        enablesave = 1;
        showplot = 1;

        opt.method = struct('name',methodname,'psize', psize);
        opt.param.gabor = struct('Downsampling', 128);
        opt.RF =
struct('NTrees',500,'parallel',1,'RandStream',0,'SelectBest',0);
```

```
        opt.database = struct('name',
databasename,'fileselect',selecttype);

        opt.database = InitDatabase(opt.database);
        rf = TrainRF(opt);

    case 'TRAINBEST'
        disp('Training Mode (Best)');

        rf = loadRF('ggg_500');
        enablesave = 1;
        showplot = 1;

        opt.best = struct('NFeatures', 10);
        rf.RF.NTrees = 60;

        rf = TrainBestRF(rf,opt);

    case 'LOAD'
        disp('Loading Mode');
        loadfile1 = 'ggg_500';           %Gabor
        loadfile2 = 'pgg_500';            %Phase

        rf = loadRF(loadfile1);

        complete = 1;
        if(complete)
            rfg = rf;
            rf = loadRF(loadfile2);
            rfp = rf;
        end

        switch loadmode
            case 'EVAL'
                disp('Evaluation Load Mode');
                showplot = 1;
            case 'TEST'
                disp('Testing Mode');

                showplot = 0;
                if (complete)
                    result = testRF(rfg,rfp,testmode);
                else
                    result = testRF(rf,testmode);
                end
            otherwise
        end

    otherwise
end

%% Plot Information

if(showplot)
    display.show =
struct('ooberror',0,'featureimp',1,'fracinbag',0,...

'coordinate',0,'outlier',0,'erroreachtree',0,'eigen',0,'classmargin'
,0,'recognitionrate',0);
```

```
    display.param = struct('OutlierThreshold', 10);

    DisplayRF(rf,display);
end

%% Save Option

if(strcmp('TRAIN',mode) || strcmp('TRAINBEST',mode))
    if(enablesave)
        saveRF;
    end
end

%% END

disp('DONE');
disp('========================');
```

InitDatabase.m

```
function [ database ] = InitDatabase( database )
%INITDATABASE Summary of this function goes here
%   Detailed explanation goes here


traindir = strcat('database\train\',database.name);

switch database.fileselect
    case 'all'
        temp = dir(traindir);
        NSample = length(temp);
        NSample = NSample-2;

        SampleInfo = zeros(NSample,2);
        SampleList = cell(NSample,1);

        for i=1:NSample
            SampleInfo(i,1) = i;
            SampleName = temp(i+2).name;
            SampleList{i} = SampleName;
            TrainFiles = dir([traindir,'\',SampleName,'\*.jpg']);
            SampleInfo(i,2) =
length(TrainFiles(not([TrainFiles.isdir])));
        end


    otherwise
end


database.dir = traindir;
database.NSample = NSample;
database.SampleInfo = SampleInfo;
database.SampleList = SampleList;

end
```

loadRF.m

```
function [ rfData ] = loadRF(RFname )

if nargin == 0
    mode = 'UI';
else
    mode = 'DEFINED';
end

switch mode

    case 'UI'
        loadfile = uigetfile('*.mat', 'Select load file' );

    case 'DEFINED'
        loadfile = RFname;
end

loadcommand = ['load ' loadfile];
eval(loadcommand);

fprintf('Load %s file completed \n',loadfile);
rfData = rf;
end
```

TrainRF.m

```
function [ rf ] = TrainRF( opt )
%INIT Summary of this function goes here
%   Detailed explanation goes here

%% Initialization of rf

rf.method = opt.method;
rf.param = opt.param;
rf.database = opt.database;
rf.RF = opt.RF;


%% Parallel Processing Initialization

EnableParallel = opt.RF.parallel;

if(EnableParallel)
    Options = statset('UseParallel','always',
'UseSubstreams','always');
    if(~(matlabpool('size') > 0))      % Check if matlabpool is
running
        matlabpool local 2;      % Run matlabpool if it is not
started
    end
else
    Options = statset('UseParallel','never',
'UseSubstreams','always');
    if(matlabpool('size') > 0)      % Check if matlabpool is running
```

```
        matlabpool close;
    end
end


%% RandStream Initialization

s = RandStream('mlfg6331_64' ,'Seed',0);
RandStream.setGlobalStream(s);
stream = RandStream.getGlobalStream;
reset(stream);

rf.RF.paralleloption = Options;

%% Feature Extraction Method

Method = rf.method.name;
psize = rf.method.psize;

if (strcmp(Method,'LBP'))
    mapping=getmapping(8,'u2');
    rf.method.mapping = mapping;
end

switch Method
    case 'LBP'
        disp('LBP');
    case 'GABOR'
        filter_bank = construct_Gabor_filters_PhD(8, 5, psize);
        rf.method.filter_bank = filter_bank;
        disp('GABOR');
    case 'PHASE'
        filter_bank = construct_Gabor_filters_PhD(8, 5, psize);
        rf.method.filter_bank = filter_bank;
        disp('PHASE');
    otherwise
end


%% Load Database & Feature Extraction

fprintf('Train Database : %s \n',opt.database.name);


NSample = opt.database.NSample;
traindir = opt.database.dir;
SampleList = opt.database.SampleList;

data_matrix = [];
ids = [];
totalsample = 0;

for i=1:NSample
    NSampleEach = opt.database.SampleInfo(i,2);
    TrainFiles = dir([traindir,'\',SampleList{i},'\*.jpg']);
    for j=1:NSampleEach
        str = 
strcat(traindir,'\',SampleList{i},'\',TrainFiles(j).name);
```

```
        X = imread(str);
        if (length(size(X)) == 3)
            X = rgb2gray(X);
        end

        feature_vector = FeatureExtract(Method,X,rf);

        data_matrix = [data_matrix,feature_vector];
        ids = [ids;i];
        totalsample = totalsample+1;
    end
end

data_matrix = double(data_matrix');


rf.database.data_matrix = data_matrix;
rf.database.ids = ids;
rf.database.totalsample = totalsample;

disp('Finished Database Loading');

%% Random Forest Training

disp('Training Random Forest');

NTrees = opt.RF.NTrees;
% NFeatures = opt.RF.NFeatures;

tic
b = TreeBagger(NTrees,data_matrix,ids,...
    'surrogate','off','oobvarimp', 'on','Options', Options );
toc

rf.b = b;

end
```

TrainBestRF.m

```
function [ rf ] = TrainBestRF( rf,opt )
%TRAINBESTRF Summary of this function goes here
%   Detailed explanation goes here

%% Parallel Processing Initialization

EnableParallel = rf.RF.parallel;

if(EnableParallel)
    Options = statset('UseParallel','always',
'UseSubstreams','always');
    if(~(matlabpool('size') > 0))      % Check if matlabpool is
running
```

```
        matlabpool local 2;     % Run matlabpool if it is not
started
    end
else
    Options = statset('UseParallel','never',
'UseSubstreams','always');
    if(matlabpool('size') > 0)     % Check if matlabpool is running
        matlabpool close;
    end
end

rf.RF.paralleloption = Options;

%% RandStream Initialization

s = RandStream('mlfg6331_64' ,'Seed',0);
RandStream.setGlobalStream(s);
stream = RandStream.getGlobalStream;
reset(stream);

rf.RF.paralleloption = Options;

%% Initialization

NTrees = rf.RF.NTrees;
NFeatures = opt.best.NFeatures;
data_matrix = rf.database.data_matrix;
ids = rf.database.ids;
bf = rf.b;

%% Training


[~, sort_idx] = sort(bf.OOBPermutedVarDeltaError,'descend');
best_idx = sort_idx(1:NFeatures);

tic
bs = TreeBagger(NTrees,data_matrix(:,best_idx),ids,...
'surrogate','off','oobvarimp','on','Options', Options);
toc

rf.RF.best_idx = best_idx;
rf.RF.SelectBest = 1;
rf.b = bs;

end
```

FeatureExtract.m

```
function [ feature_vector ] = FeatureExtract( method,im,rf)
%FEATUREEXTRACT Summary of this function goes here
%   Detailed explanation goes here

psize = rf.method.psize;

switch method
    case 'LBP'
        X = imresize(im, psize);
        mapping = rf.method.mapping;
        feature_vector = (lbp(X,1,8,mapping,'h'))';
    case 'GABOR'
        DownSamplingFactor = rf.param.gabor.Downsampling;
        X = double(im);
        X = imresize(X, psize,'bilinear');
        filter_bank = rf.method.filter_bank;
        feature_vector = filter_image_with_Gabor_bank_PhD...
            (X,filter_bank,DownSamplingFactor);
    case 'PHASE'
        DownSamplingFactor = rf.param.gabor.Downsampling;
        X = double(im);
        X = imresize(X, psize,'bilinear');
        filter_bank = rf.method.filter_bank;
        [pc,EO] = produce_phase_congruency_PhD(X,filter_bank);
        feature_vector = resize_pc_comps_PhD(pc,
DownSamplingFactor);
    otherwise
end

end
```

TestRF.m

```
function [ result ] = testRF(rf1, rf2, mode )
%TESTRF Summary of this function goes here

complete = 0;
rf = rf1;

if nargin == 2
    testmode = rf2;

elseif nargin == 3
    rfp = rf2;
    testmode = mode;
    complete = 1;
    para=0.3;
end


%% Display database trained

DatabaseName = rf.database.name;
fprintf('Database Name : %s\n',DatabaseName);
```

```
%% Best Feature Selection
b = rf.b;

if (complete)
    b1 = rfp.b;
end

%% Initialization

Method = rf.method.name;

if (complete)
    Method1 = rfp.method.name;
end
RootDir = pwd;

%% Test Operation

% 1 - Manual pick photo
% 2 - Test all test sample
% 3 - Face Detect Mode
switch testmode
    case 1
        %% Manual Mode

        graph = 0;

        if(graph)
            if (complete)
                Row = 4;
            else
                Row = 3;
            end
        else
            Row = 2;
        end

        disp('Manual');

        TrainPath = strcat(RootDir, '\database\train\',
DatabaseName,'\');
        TestPath = strcat(RootDir, '\database\test\',
DatabaseName,'\');
        [TestFile, PathName] = uigetfile(strcat(TestPath,'\*.jpg'),
'Select test file' );
        TestDir = strcat(PathName, TestFile);
        TrueClass = PathName((length(TestPath)+1):(length(PathName)-
1));

        for i=1:rf.database.NSample
            if strcmp(TrueClass,rf.database.SampleList{i})
                TrueLabel = i;
            end
        end

        ori = imread(TestDir);
        if (length(size(ori)) == 3)
            X = rgb2gray(ori);
        end

        feature_vector = FeatureExtract(Method,X,rf);
        feature_vector = feature_vector';

        if (complete)
            feature_vector1 = FeatureExtract(Method1,X,rfp);
            feature_vector1 = feature_vector1';
        end

        if (rf.RF.SelectBest)
            feature_vector = feature_vector(:,rf.RF.best_idx);
            if (complete)
                feature_vector1 =
feature_vector1(:,rfp.RF.best_idx);
            end
        end

        tic
        [Y,scores,stdevs] = predict(b,feature_vector);

        if (complete)
            [Y1,scores1,stdevs] = predict(b1,feature_vector1);
        end
        toc

        figure;
        if (complete)
            subplot(Row,1,1);
        else
            subplot(Row,1,1);
        end
        imshow(ori);
        title('Testing Image');

        label = rf.database.SampleList(str2num(cell2mat(Y)));
        Line1 = ['True Class : ' TrueClass '  (' num2str(TrueLabel)
')' ];

        if (complete)
            finalscores = (1-para)*scores+para*(scores1);
            [FinalSc, FinalIdx] = max(finalscores)
            label = rf.database.SampleList(FinalIdx);
            Line2 = ['Predict Class : ' char(label) '  ('
num2str(FinalIdx) ')' ];

            if (graph)
                subplot(Row,1,3);
                bar(scores);
                title('Prediction Score for Each Class (GABOR)');
                xlabel('Class');
                ylabel('Prediction Scores');

                subplot(Row,1,4);
                bar(scores1);
                title('Prediction Score for Each Class (PHASE)');
                xlabel('Class');
                ylabel('Prediction Scores');
            end
            StoreScore = [scores;scores1];
        else
            Line2 = ['Predict Class : ' char(label) '  (' char(Y)
')' ];

            if (graph)
                subplot(Row,1,3);
                title('score');
                bar(scores);
                title('Prediction Score for Each Class');
                xlabel('Class');
                ylabel('Prediction Scores');
            end
            StoreScore = scores;
        end

        if (complete)
            subplot(Row,1,1);
        else
            subplot(Row,1,1);
        end
        MatchImgPath = strcat(TrainPath,label,'\');
        DirImg = dir([char(MatchImgPath), '\*.jpg']);
        MatchImg = strcat(MatchImgPath,DirImg(1).name);

        subplot(Row,1,2);
        imshow(imread(char(MatchImg)));
        title('Match Image');

        xmsg = {Line1 ; Line2;' '};
        xlabel(xmsg);



    case 2
        %% Auto TestMode
        para=0:0.1:1.0;

        disp('Auto Test Mode');

        NSample = rf.database.NSample;
        SampleList = rf.database.SampleList;

        TestPath = strcat(RootDir, '\database\test\',
DatabaseName,'\');
        TestDir = dir(TestPath);
        TestDir = TestDir([TestDir.isdir]);
        TestDir(strncmp({TestDir.name}, '.', 1)) = [];

        NTest = length(TestDir);
        NRemoveTest = 0;
        TestLabel = [];
%         TestName = [];

        % Compare TestDir with Train Database, remove untrained test
dir
```

```
        for i=1:NTest
            found = 0;
            for j=1:NSample
                if strcmp(TestDir(i-NRemoveTest).name,SampleList{j})
                    TestLabel = [TestLabel; j];
                    found = 1;
                    break;
                end
            end

            if (~found)
                TestDir(i) = [];
                NRemoveTest= NRemoveTest+1;
            end
        end

        NVerifiedSample = length(TestLabel);
        TotalTest = 0;
        StorePredict = [];
        StoreScore = [];
        StoreScore1 = [];
        Error = 0;

        % Compute the Total Number
        tic
        for i=1:NVerifiedSample
            SampleDir = strcat(TestPath, TestDir(i).name);
            InfoTestDir = dir([SampleDir , '\*.jpg']);
            NumTestEach =
length(InfoTestDir(not([InfoTestDir.isdir])));
            for j=1:NumTestEach
                str = strcat(SampleDir,'\',InfoTestDir(j).name);

                X = imread(str);
                if (length(size(X)) == 3)
                    X = rgb2gray(X);
                end

                feature_vector = FeatureExtract(Method,X,rf);
                feature_vector = feature_vector';

                if (complete)
                    feature_vector1 = FeatureExtract(Method1,X,rfp);
                    feature_vector1 = feature_vector1';
                end

                if (rf.RF.SelectBest)
                    feature_vector =
feature_vector(:,rf.RF.best_idx);
                    if (complete)
                        feature_vector1 =
feature_vector1(:,rfp.RF.best_idx);
                     end
                end

                [Y,scores] = predict(b,feature_vector);

                PredictClass = str2num(cell2mat(Y));
                if (complete)
                    [Y1,scores1] = predict(b1,feature_vector1);
                end

                if  (PredictClass ~= TestLabel(i))
                    Error=Error+1;
                end

                StoreResult= [TestLabel(i),PredictClass];
                StorePredict = [StorePredict; StoreResult];
                StoreScore = [StoreScore; scores];
                TotalTest = TotalTest +1;

                if (complete)
                    StoreScore1 = [StoreScore1; scores1];
                end
            end
        end
        toc

        if (complete)
            StoreRec = [];
            for i=1:length(para)
                y = para(i);
                finalscores = (1-y)*StoreScore+y*(StoreScore1);
                [FinalSc, FinalIdx] = max(finalscores,[],2);
                temp = StorePredict(:,1) - FinalIdx;
                correctIdx =find(temp==0);
                Recognition_Rate =
length(correctIdx)/length(StorePredict(:,1));
                StoreRec = [StoreRec; Recognition_Rate];
                AllPredict{i} = [StorePredict(:,1) FinalIdx];
            end
            result.StoreRec = StoreRec;
            result.scores1 = StoreScore1;
            result.predicttable = AllPredict;
        else

            Recognition_Rate = (TotalTest-Error)/TotalTest

            result.predicttable = StorePredict;
        end

    case 3
        %% FaceDetect Mode

        disp('FaceDetect Mode');

        addpath('C:\OpenCV2.4\mexopencv-master');
        classifier =
cv.CascadeClassifier('C:\OpenCV2.4\data\haarcascades\haarcascade_fro
ntalface_alt2.xml');

        TestPath = strcat(RootDir,'\*.jpg');
        [TestFile, PathName] = uigetfile(TestPath, 'Select test
file' );
        TestDir = strcat(PathName, TestFile);

        ori = imread(TestDir);
        if (length(size(ori)) == 3)
            gray = rgb2gray(ori);
        end

        histgray = cv.equalizeHist(gray);
        boxes =
classifier.detect(histgray,'ScaleFactor',1.20,'MinNeighbors',2,'MinS
ize',[24,24]);

        figure,
        imshow(ori);

        count = 0;

        for i = 1:numel(boxes)

rectangle('Position',boxes{i},'EdgeColor','g','LineWidth',2);
            CropImg{i} = imcrop(gray,boxes{i});
            count = count+1;
        end

        for i = 1:count

            X = CropImg{i};

            feature_vector = FeatureExtract(Method,X,rf);
            feature_vector = feature_vector';

            if (rf.RF.SelectBest)
                feature_vector = feature_vector(:,rf.RF.best_idx);
            end

            [Y,scores,stdevs] = predict(b,feature_vector);

            label{i} = rf.database.SampleList(str2num(cell2mat(Y)));
            textstr = [char(label{i}) ' (' char(Y) ')'];
            text(boxes{i}(1),boxes{i}(2), textstr,
'FontSize',10,'BackgroundColor',[.7 .9 .7]);
        end
        StoreScore = [];
    otherwise
end

result.testmode = testmode;
result.scores = StoreScore;

end
```

DisplayRF.m

```
function [rf] = DisplayRF(rf,display)
%DISPLAYRF Summary of this function goes here
%   Detailed explanation goes here

% if (rf.RF.SelectBest)
%     b = rf.bs;
%     disp('Display Selected Feature RF Information');
% else
%     b = rf.bf;
%     disp('Display Full RF Information');
% end

b = rf.b;

show = display.show;
param = display.param;

b = fillProximities(b);

% OOBError

if (show.ooberror)
    figure;
    plot(oobError(b));
    xlabel('number of grown trees')
    ylabel('out-of-bag classification error')
end

% Feature Importance

if (show.featureimp)figure;
    bar(b.OOBPermutedVarDeltaError);
    xlabel('Feature Number');
    ylabel('Out-Of-Bag Feature Importance');

end

% Fraction In-Bag Observation

if (show.fracinbag)
    finbag = zeros(1,b.NTrees);
    for t=1:b.NTrees
        finbag(t) = sum(all(~b.OOBIndices(:,1:t),2));
    end
    finbag = finbag / size(b.X,1);
    figure;
    plot(finbag);
    xlabel('Number of Grown Trees');
    ylabel('Fraction of in-Bag Observations');
end

% Outlier

if (show.outlier)
    figure;
    hist(b.OutlierMeasure);
    xlabel('Outlier Measure');
    ylabel('Number of Observations');
    fprintf('List of Outlier > %d:\n',param.OutlierThreshold);
    b.Y(b.OutlierMeasure>param.OutlierThreshold)
end

% Class Margin

if (show.classmargin)
    figure;
    plot(oobMeanMargin(b));
    xlabel('Number of Grown Trees');
    ylabel('Out-of-Bag Mean Classification Margin');
end

% Error Each Tree

if (show.erroreachtree)
    figure;
    bar(error(b,b.X,b.Y,'mode','individual'));
    xlabel('Number of Grown Trees');
    ylabel('Classification Error');
end

% Recognition Rate

if (show.recognitionrate)
    err = oobError(b);
    rec_rate = (1-err(b.NTrees))/1

end
```