Preprocessing for Illumination Variations and Face Recognition using Feature Location based Feature Extraction

A Project Report

submitted by

R.ABIRAM VELAN (EE11B003)

TULLURI VINAY KUMAR (EE11B045)

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

2015

CERTIFICATE

This is to certify that the project titled **Preprocessing for Illumination variations and Face Recognition using feature location based feature extraction**, submitted by **R.Abiram Velan (EE11B003)** and **Tulluri Vinay Kumar (EE11B045)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology** in **Electrical Engineering**, is a bonafide record of the project work done by them in the Department of Electrical Engineering, IIT Madras. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. R.Aravind
Project Guide
Professor,
Dept. of Electrical Engineering,
IIT Madras, Chennai - 600036

Place: Chennai

Date: 18^{th} May, 2015

ACKNOWLEDGEMENTS

We take this opportunity to express our sincere gratitude towards our project guide, Dr.R.Aravind, for the freedom and support he gave us during the course of the project. We are extremely grateful to him for agreeing to take us on as his students. His guidance and deep knowledge of the field have been indispensable.

ABSTRACT

Keywords: Preprocessing, Histogram Equalization, Color and Contrast Enhancement, High Frequency Emphasis filtering, CLAHE, MSRCR, Face Detection, Viola-Jones, Face Recognition, feature extraction.

The main aim of this work is to implement a new face recognition algorithm, which is computationally simple and very efficient. To deal with bad illumination conditions, which is a hindrance to face detection and recognition algorithms, some preprocessing algorithms like Histogram Equalization, Contrast Limited Adaptive Histogram Equalization, Multi-scale Retinex with Color Restoration, Color and Contrast Enhancement, High Frequency Emphasis Filtering are proposed and discussed. The face detection algorithm used is based on Viola-Jones algorithm with MATLAB implementation. The recognition algorithm is evaluated using Yale and Feret databases. Experimental results demonstrate the effectiveness of this approach of face recognition.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AHE	Adaptive Histogram Equalization
CLAHE	Contrast Limited Adaptive Histogram Equalization
SSR	Single Scale Retinex
MSR	Multi Scale Retinex
MSRCR	Multi Scale Retinex with Colour Restoration

CHAPTER 1

Introduction

As one of the most successful applications of image processing and analysis, face detection and recognition has received great significance in recent times and emerged as an active research area which spans numerous fields and disciplines. Not only for computer scientists, this field is also a topic of interest for many psychologists and neuro-scientists. The reason being face recognition, in addition to having numerous practical day to day applications such as access control, security monitoring, surveillance systems etc., is also a fundamental human behavior that is essential for effective human communication and interactions.

Face recognition may not be the most reliable and efficient when compared to other bio-metrics, but one major advantage is that it does not require much cooperation from the individual as others do. Current face recognition technologies struggle to perform under certain conditions such as facial expressions, pose variation, poor lighting, sunglasses, long hair, aging etc. Lots of research is being carried out to overcome these issues and to come up with more robust systems for recognition. To a certain extent, algorithms which can deal with slight pose variation, in-plane rotation of faces, poor lighting conditions etc., have come up. But due to the increasing significance and applications of face recognition technology in recent times, more efficient systems are under demand which in turn is encouraging a lot of research in this area.

The two very general applications of face recognition are Face Identification - given a face image, the system should be able to tell who he/she is or the most probable - and Face Verification - given a face image and a guess of the identification, system should be able to tell whether our guess about the given facial image is true or false.

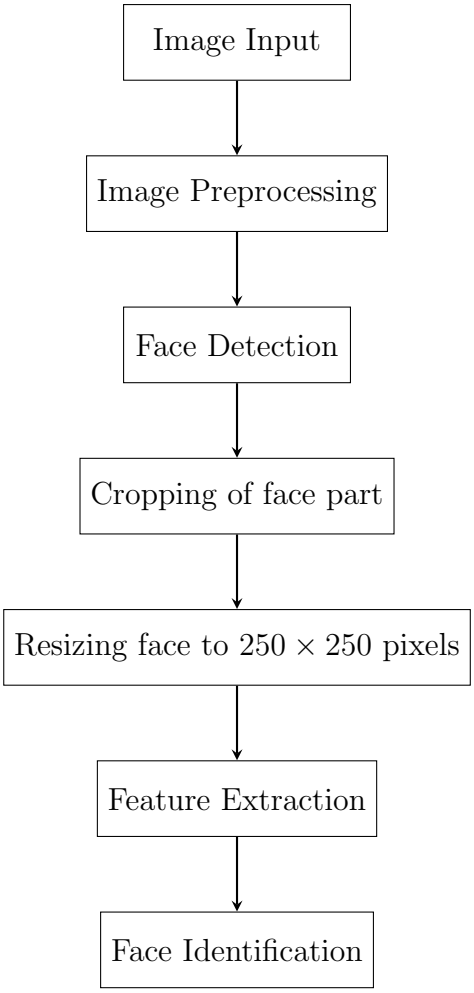

Figure 1.1: Face Recognition Process

Image pre-processing is one of the very crucial steps in the face recognition process to tackle the various issues that were mentioned earlier. This report majorly is targeted to deal with images under poor lighting or illumination conditions. In Chapter 2, some image pre-processing algorithms, namely Histogram Equalization, Contrast Limited Adaptive Histogram Equalization, Multi-scale Retinex with Color Restoration, Color and Contrast Enhancement, High Frequency Emphasis Filtering are proposed and discussed.

Chapter 3 is about the face detection algorithm used to detect faces. A brief description of Viola-Jones algorithm, on which the detection method used is based on, is given and the MATLAB implementation of the algorithm is discussed. Few improvements are made to the existing algorithm to deal with issues like non-face detections and in-plane rotation of faces.

Chapter 4 is about a new method of face recognition based on a paper by Walaa Mohamed et al.[8], which is proposed and discussed. Owing to various applications of face recognition these days in cloud technology, live stream applications, etc., where the speed of the computations matters a lot, traditional algorithms might not be very efficient in such cases. The algorithm discussed in this chapter is computationally simple and also very efficient, which could make this algorithm a very application friendly one.

CHAPTER 2

Image Preprocessing

Image Preprocessing is the very first step in Digital Image Processing. The principle objective of image preprocessing is to improve the interpretibilty and perception of information, for human viewers or an automated image processing technique, by eliminating the unwanted distortions and enhancing some of the image features essential for further processing. It is a highly subjective problem and depends mainly on the type of application.

The image preprocessing techniques can be broadly classified into the following two categories:

1. **Spatial Domain Techniques**, which directly deals with the image pixels and manipulates the intensity value of each pixel to achieve the desired enhancement.
2. **Frequency domain Techniques**, in which the image is transformed to its frequency domain, the enhancement techniques are performed on the transformed image and then the image is transformed back to the spatial domain.

Image preprocessing is an essential step for the field of Face Recognition. This is because of the fact that when an image is captured, it is affected by a lot of factors like noise, illumination condition, optical and motion blur, etc. These factors in turn affect the facial features which will be used for face detection and recognition. Thus preprocessing techniques are used to enhance the facial features and improve the recognition results. Some of the image preprocessing techniques are discussed in detail in this section.

Among all the Preprocessing algorithms mentioned below, the High Frequency Emphasis Filtering followed by Histogram Equalization was found to give the best results with the face recognition algorithm used in this thesis.

2.1 Histogram Equalization

Histogram Equalization is a commonly used contrast enhancement technique. In the histogram of a low contrast image, most of the pixels are clustered around a narrow band of intensity levels and thus leaving a huge range of intensity levels unused. Figure 2.1 shows a low contrast image along with its histogram. It can be seen that most of the pixels are concentrated towards the left of the histogram. Histogram Equalization maximizes the contrast of the given image by trying to flatten its histogram, thus making use all the gray intensity levels available in the dynamic range.

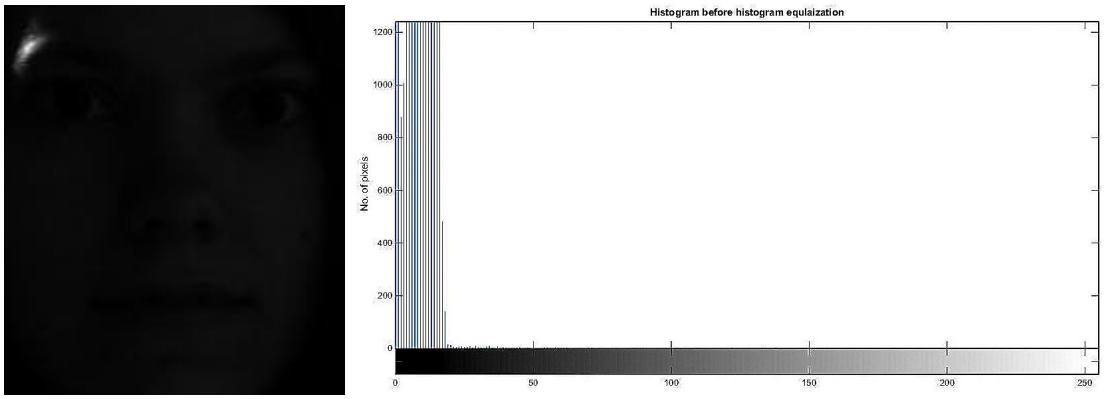

Figure 2.1: Low contrast image and its histogram. Image courtesy: Yale Face database [1]

The histogram of a digital image with n number of pixels and L intensity levels in the range $[0, G]$ is defined as a discrete function,

$$h(r_k) = n_k \tag{2.1}$$

where, r_k is the k^{th} intensity level in the interval $[0, G]$ and n_k is the number of pixels in the image with intensity level r_k. For images of class `uint8`, the value of G is 255 and $G = L - 1$. The normalized histogram is obtained by dividing all the elements of $h(r_k)$ by the total number of pixels

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n} \tag{2.2}$$

for $k = 1, 2, ..., L$. The equalization transformation is given by

$$s_k = T(r_k) = \sum_{j=1}^{k} p(r_j)$$
$$= \sum_{j=1}^{k} \frac{n_j}{n} \tag{2.3}$$

for $k = 1, 2, ..., L$, where s_k is the intensity level in the histogram equalized image corresponding to value r_k in the input image.

Figure 2.2 shows the histogram equalized image of the low contrast image in figure 2.1 and also the histogram of the histogram equalized image. It can be seen that the histogram is well spread, making use of all the intensity levels available.

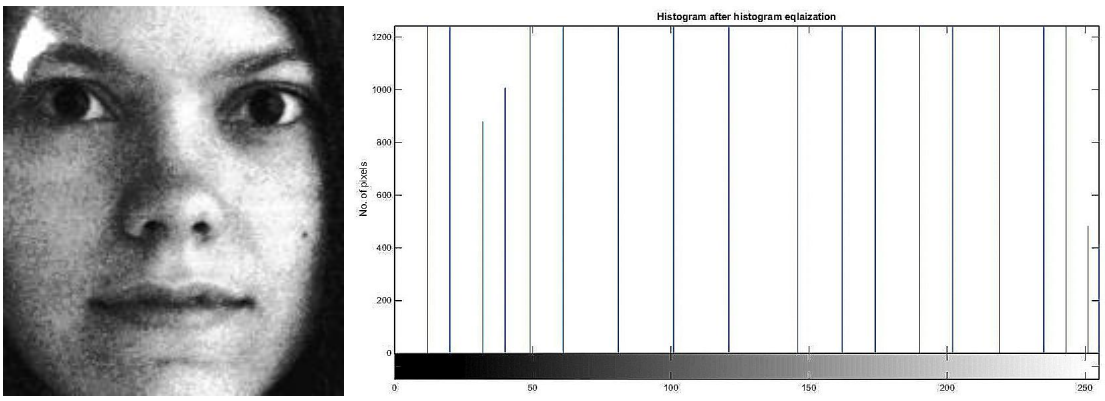

Figure 2.2: Histogram equalized image and its histogram. Image courtesy: Yale Face database [1]

2.2 Color and Contrast Enhancement

Preprocessing techniques generally operate globally on the entire image or act locally or a combination of both. This preprocessing technique by Mark De-Nies [2] is a combination of limited local scaling and limited global power-law transformation, thus providing a reasonable compromise between global and local enhancement algorithms.

This enhancement technique modifies the image intensity and image saturation. Hence, it performed on an image in HSV format, and the color enhancement is done on the S (saturation) plane and intensity enhancement is done on the V (intensity) plane. It involves separation of intensity and saturation values to enhance the features of the image. At the same time, the direction of separation is preserved, i.e., the darker (or brighter) pixels in a neighborhood remains darker (or brighter) locally, though the gradient of separation is reduced in high gradient regions to facilitate gradients to increase in the low gradient regions.

First, the input image is converted from RGB format to HSV format. The S and V planes of the image are processed separately. Rebalancing of the intensity values towards the middle of the range of values is done by applying the following power-law adjustment to the V plane of the image.

$$s = r^{\gamma} \tag{2.4}$$

where $\gamma = (I_{avg} - I_{midrange})^2$ and I_{avg} is the average intensity in the image and $I_{midrange}$ is the middle value of the range of intensity level, which is 127.5 (if range is 0 to 255) and 0.5 (if range is 0 to 1). The value of γ is typically restricted to lie between 0.67 and 1.5. This step tends to spread out the intensity values in the dark intensity range for predominantly darker images or darkens the brighter images.

The S and V planes are scanned using 128 successively larger threshold values calculated as

$$thres_i = \frac{(max\_val \times i)}{128} \tag{2.5}$$

where $i = 1, 2, ..., 128$, $thres_i$ is the i^{th} threshold value and max_val is the maximum pixel value of that particular plane of the image. For each threshold value, patches are identified in both S and V planes which contains pixel values greater than the threshold value surrounded by pixels with smaller values. These patches are then stretched by the addition of a multiple of difference between input value

and current threshold to its output value. Also, to avoid radical changes to local intensities, the stretching is limited to $1/8^{th}$ of the input value. Two passes are made for each plane - first pass stretches the pixels towards higher values and for the second pass, the plane is inverted and the same procedure is repeated, stretching the pixels towards the lower values. Finally, the resultant S and V planes are re-normalized to fit the range of 0 to 255 or 0 to 1.

Figure 2.3: Original image (left); Image after Color and Contrast Enhancement (right). Image courtesy: FERET database [5]

Figure 2.3 shows an image and its enhanced output image. It can be seen that both the color and intensity contrast have been enhanced in a natural appearing way. However this algorithm is slower compared to other enhancement techniques discussed. The MATLAB functions used for the implementation of Contrast and Color Enhancement are given in Appendix A.3.

2.3 High Frequency Emphasis Filtering

In an image, the high frequency components are associated with the edges and other abrupt changes in the gray levels. Thus, sharpening of the image can be achieved in the frequency domain by using a highpass filtering process in the Fourier Transform. Highpass filters zero out the *dc* term, thus reducing the average intensity value of the image to 0. To avoid this, an offset is added to the highpass

filter. This offset along with the multiplication of the filter by a constant greater than 1, helps in enhancing the high frequency components of the image while at the same time retaining the gray level tonality due to the low frequency components.

The transfer function for the High Frequency Emphasis filtering is given by the following equation:

$$H_{hfe}(u,v) = a + bH_{hp}(u,v) \tag{2.6}$$

where $a \geq 0$ and $b > a$, typically a lies in the range 0 to 0.5 and b lies in the range 1.5 to 2 (here $a = 0.5$ and $b = 1.5$ is used), $H_{hp}(u,v)$ is the transfer function of the highpass filter. For this method, a Butterworth highpass filter of n^{th} order is used, whose transfer function is given by:

$$H_{hp}(u,v) = \frac{1}{1 + \left(\dfrac{D_0}{D(u,v)}\right)^{2n}} \tag{2.7}$$

where D_0 is the specified non-negative quantity (here $D_0 = 1$ is used) and $D(u,v)$ is the distance of the point (u,v) from the center of the frequency rectangle. Two discrete functions are considered in the two dimensional discrete space, denoted by f and g, and let $f(x,y)$ denote gray level of the input image at the point (x,y), $g(x,y)$ denote the gray level of the output image at the point (x,y) and F and G denote the Fourier transform of the corresponding images. Then, the expression for high frequency emphasis filtering can be obtained from equations 2.6 and 2.7 as:

$$\begin{aligned} G(u,v) &= H_{hfe}(u,v)F(u,v) \\ &= \left(a + \frac{b}{1 + \left(\frac{D_0}{D(u,v)}\right)^{2n}}\right)F(u,v) \end{aligned} \tag{2.8}$$

Using the convolution theorem, the above equation can be represented in the spatial domain as:

$$g(x,y) = h_{hfe}(x,y) * f(x,y) \tag{2.9}$$

where $h_{hfe}(x, y)$ is the inverse Fourier transform of the filter transfer function $H_{hfe}(u, v)$.

Figure 2.4: Original image (left); Image after High frequency emphasis filtering (centre); Image after Histogram Equalization (right). Image courtesy: Yale face database [1]

After High frequency emphasis filtering, in order to further improve the contrast of the face and background, Histogram equalization (section 2.1) is applied on the filtered image. Figure 2.4 shows the original image, the image after high frequency emphasis filtering, and the image after further application of histogram equalization to the filtered image. The MATLAB function used for the implementation of high frequency emphasis filter is given in Appendix A.2.

2.4 CLAHE

The normal histogram equalization technique uses the same transformation derived from the image histogram to all the pixels of the image. In case of Adaptive Histogram Equalization (AHE), the transformation function is derived for each pixel in the image from its neighborhood region. The problem with this method was that it was also enhancing the noise in relatively homogeneous regions. To solve this problem, a new method called the Contrast Limited Adaptive Histogram Equalization (CLAHE) was proposed by S.M.Pizer [6].

This CLAHE method applies histogram equalization to smaller neighborhoods and the histogram is clipped. The high peaks in the histogram are normally caused due to homogeneous regions when a narrow range of intensity values are mapped to a wide range of intensity values. Hence, enforcing a maximum limit on the number of pixels for any gray level will limit the extent of contrast enhancement and thus enhancement of noise. Now the image histogram has to be renormalized. This is done by redistributing the clipped pixels uniformly over all the histogram bins. This is shown in figure 2.5.

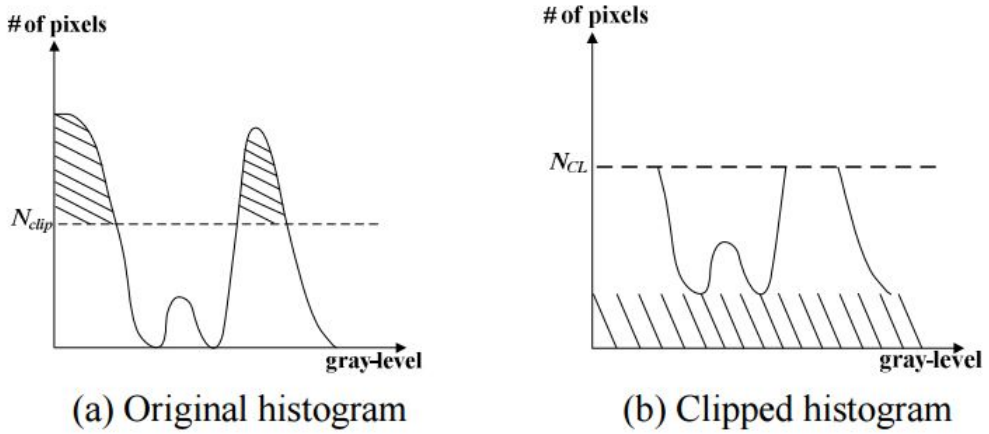

Figure 2.5: Original Histogram and the Histogram after clipping. Figure courtesy: Zhiyuan Xu [10].

The clip limit N_{CL} is specified by the user. Adding a uniform level L to the clipped histogram will increase the histogram over the clip limit. Thus, the original histogram is clipped at a lower limit N_{clip}, so that after the addition of L (which depends on N_{clip}) the maximum of histogram is equal to the clip limit N_{CL}.

$$v = \begin{cases} v_{orig} + L & \text{if } v_{orig} < N_{clip} \\ N_{CL} & \text{if } v_{orig} \geq N_{clip} \end{cases} \tag{2.10}$$

where v_{orig} is the value in the original histogram and v is the value in the histogram after clipping. The Figure 2.6 shows an image before and after application of CLAHE.

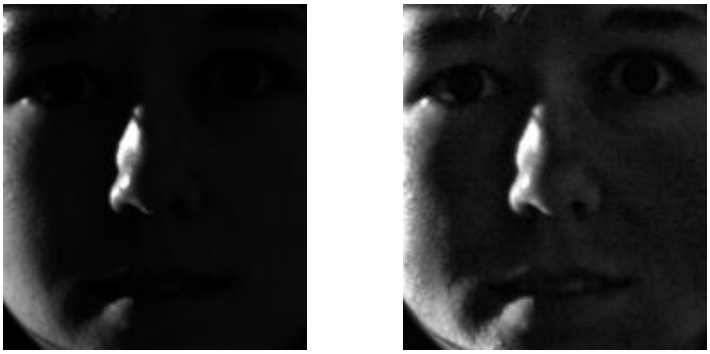

Figure 2.6: Original image (left); Image after CLAHE (right). Image courtesy: Yale face database [1]

2.5 Multi Scale Retinex with Color Restoration

The development of Multi-Scale Retinex with Color Restoration (MSRCR) algorithm started after the last published concept for Retinex by Edwin Land [4], which introduced a surround spatial form. The MSRCR algorithm combines color constancy with local contrast enhancement to make the images approach the realism of direct scene observation. This algorithm has three stages - Single Scale Retinex, Multi Scale Retinex and Color Restoration.

The **Single Scale Retinex** (SSR) uses the Gaussian surround function proposed by Hurlbert [3], given by the following expression:

$$F(x,y) = Ke^{(-r^2/c^2)} \tag{2.11}$$

where $r = \sqrt{x^2+y^2}$, c is the Gaussian surround space constant and K is the normalization factor calculated from,

$$\sum_x \sum_y F(x,y) = 1 \tag{2.12}$$

The SSR is given by,

$$R_{SSRi}(x,y) = \log[I_i(x,y)] - \log[F(x,y) * I_i(x,y)] \tag{2.13}$$

where $F(x, y)$ is the surround function, I is input image and the subscript i refers to the different planes (R,G,B for color images and the only plane for grayscale images) of the image. Choosing the value of c for the surround function is a trade-off between compression and rendition. Smaller value of c ensures good dynamic range compression whereas a large value of c ensures good tonal rendition.

The **Multi Scale Retinex** (MSR) overcomes this problem with SSR by establishing a good balance between dynamic range compression and color rendition. The MSR is represented by:

$$R_{MSRi}(x, y) = \sum_{n=1}^{N} \omega_n \big[\log \big(I_i(x, y) \big) - \log \big(F_n(x, y) * I_i(x, y) \big) \big] \tag{2.14}$$

where N is the number of scales, ω_n is the weight associated with the n^{th} scale (generally $N = 3$ and $\omega_n = \frac{1}{3}$), F_n is the surround function of n^{th} scale with surround space constant c_n and subscript i refers to the different planes of the image. The dynamic range compression results in the violation of the Gray world algorithm, which states that the average of red, green and blue components of an image should average out to a common gray value. Also the region of constant color bleaches out. Thus MSR was good enough for grayscale images but not suited for color images.

These limitations of MSR are overcome by adding a **Color Restoration** Step to arrive at the **Multi Scale Retinex with Color Restoration** (MSRCR). The color restoration is calculated using the expression,

$$C_i(x, y) = \beta \big[\log \big(\alpha I_i(x, y) \big) - \log \big(\sum_{i=1}^{S} I_i(x, y) \big) \big] \tag{2.15}$$

where β is the gain constant, α controls the strength of non-linearity and S is the number of image planes (generally 3). The final representation of MSRCR is given by:

$$R_{MSRCRi}(x, y) = G\big(C_i(x, y) \times R_{MSRi}(x, y) + b \big) \tag{2.16}$$

where G is the gain constant and b is the gain offset value. The MSRCR implementation compared to MSR gives better contrast and better color restoration. The figure 2.7 shows a color image along with its MSR output and MSRCR output. The MATLAB function used for the implementation of MSRCR is given in Appendix A.1.

Figure 2.7: Input color image (left); Image after MSR (centre); Image after MSRCR (right). Image courtesy: FERET database [5]

CHAPTER 3

Face Detection and Extraction of facial features

Face Detection is the first step in any face recognition algorithm. It will search for faces in an image or video and return the location of the faces found. The difficulty associated with face detection can be attributed to variations in scale, location, orientation (in-plane rotation), pose (out-of-plane rotation), facial expression, lighting conditions and occlusions. There have been various reported approaches for face detection. According to M.H.Yang [11], the various methods of face detection can be categorized into four categories:

1. **Knowledge based methods**, which use predefined rules based on human knowledge to determine a face.
2. **Feature invariant approaches**, which aim at finding face structure features that robust to pose and illumination variations.
3. **Template matching methods**, which use pre-stored face templates to judge if an image is a face or not.
4. **Appearance based methods**, which learn models from a set of representative training face images to perform detection.

3.1 Viola-Jones Algorithm

For the recognition algorithm explained in the next chapter (Chapter 4), MATLAB's `vision.CascadeObjectDetector` is used, which is based on the Viola-Jones Algorithm [7]. The Viola-Jones algorithm consists mainly of three parts to enable a fast and accurate detection. They are:

1. **the Integral Image**, for feature computation.
2. **Adaboost Learning**, for feature selection and classifier learning.
3. **Attentional Cascade Structure**, for efficient computational resource allocation.

The Integral Image part of the algorithm mainly deals with the computation of haar-like features. A haar-like feature considers adjacent rectangular regions at a specific location in the detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For face detection, all human faces share some similar properties. This knowledge is used to construct haar features.

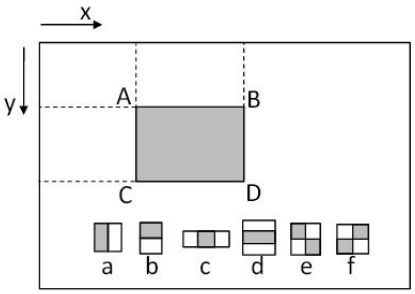

Figure 3.1: Illustration of Integral Image and Haar-like rectangular features ($a-f$). Image courtesy: [12]

The **Integral image** (also known as the summed data table), is an algorithm for fast and efficient computation of sum of values in the rectangle subset of grid. Integral images are computed as shown below:

$$S(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{3.1}$$

where $S(x,y)$ is the integral image at pixel location (x,y) and $i(x',y')$ is the original image. The sum of pixels in rectangle region ABCD in figure 3.1 can be calculated as:

$$\sum_{(x,y) \in ABCD} i(x,y) = S(D) + S(A) - S(B) - S(C) \tag{3.2}$$

The modified **Adaboost Algorithm** is used for feature selection. For a detection region of size 24x24 detection region, the number of possible rectangle features is close to 180,000. Among all these features, only a few of them are expected to give almost consistently high values when on top of a face. In order to find these

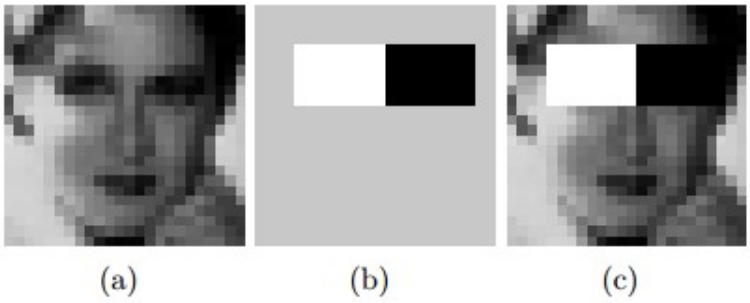

Figure 3.2: (a) given training image; (b) Considered Haar-like pattern; (c)Feature calculation with difference in sum of image pixel values in white to that of black. Image courtesy: [9]

features, Viola-Jones used a modified version of the AdaBoost algorithm. AdaBoost is a machine learning boosting algorithm which is capable of constructing a strong classifier through a weighted combination of weak classifiers. The modified algorithm is designed to select only the best features. The best performing feature is selected based on the weighted error given by it.

The next part is **Attentional cascade** and the major idea behind this part is that smaller,and thus more efficient, boosted classifiers can be built which can reject most of the negative sub-windows while keeping almost all the positive examples. As a result, majority of the sub-windows will be rejected in early stages of the detector, making the process extremely efficient.

3.2 MATLAB's implementation of Viola-Jones algorithm

The MATLAB libraries are used to implement the Viola-Jones algorithm for face detection. The cascade object detector library uses the Viola-Jones algorithm to detect faces and facial features like nose, eyes and mouth. Training image labeler tool box in MATLAB can be used to train a custom classifier to use with any System object. `detector = vision.CascadeObjectDetector(<model>)` creates an object, detector, configured to detect objects defined by the input string, `<model>`.

The `<model>` input describes the type of object to detect. There are several valid `<model>` strings, such as 'FrontalFaceCART', 'UpperBody' and 'ProfileFace'. The Classification Model property controls the type of object to be detected.

The following are the Trained Cascade Classification Models used:

1. **FrontalFaceCART** (Default) - detects faces that are upright and forward facing. This model is composed of weak classifiers, based on the classification and regression tree analysis (CART). These classifiers use Haar features to encode facial features. CART-based classifiers provide the ability to model higher order dependencies between facial features.
2. **EyePairBig/EyePairSmall** - detects a pair of eyes. The EyePairSmall model is trained using smaller images. This enables the model to detect smaller eyes than the EyePairBig model can detect.
3. **Mouth** - detects the mouth. This model is composed of weak classifiers, based on a decision stump, which use Haar features to encode mouth details.
4. **Nose** - detects the nose. This model is also composed of weak classifiers, based on a decision stump, which use Haar features to encode nose details

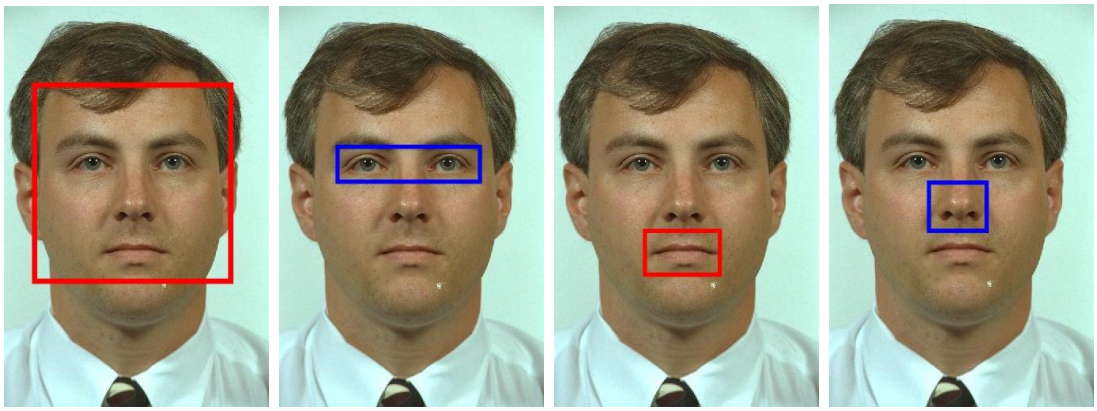

Figure 3.3: The results of face, eyes, mouth and nose detection. Image courtesy: FERET database [5]

3.3 Improving Face Detection

The face detection algorithm might not work as expected under some circumstances like in-plane rotation of faces, bad illumination conditions, etc. The results of the face detection algorithm can be improved as discussed below:

3.3.1 Dealing with non-face detections

During face detection in an image, sometimes there will be detection of some smaller non-face parts along with the detection of face. To avoid such non-face detections, the detections of dimensions less than a fraction of that of the largest detection can be ignored, resulting in the reduction in chances of false detection. This can be seen in figure 3.4.

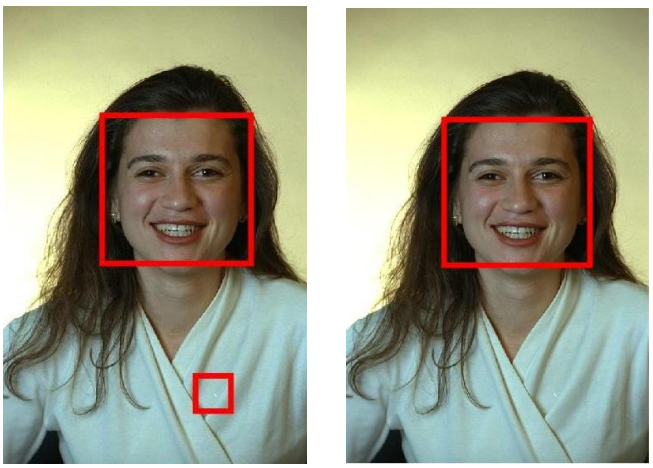

Figure 3.4: Image with false detection (left); Detection after improvement (right). Image courtesy: FERET database [5]

3.3.2 Dealing with in-plane rotation of faces

When a face is found rotated (in-plane) beyond a certain angle in the image, the face detection algorithm will not be able to detect the face. This can be handled by the rotation of the entire image in small steps on either directions (clockwise and anti-clockwise) and detecting faces at each step. When a face is tilted in certain image and if face is not detected, then according to the improvement suggested when the image is rotated in small intervals at some point of rotation of image, face will be either straight or atleast at an angle at which face could be detected. The figure 3.5 shows the improvement in the detection of faces.

Figure 3.5: (a) Given Image; (b) faces detected before improvement; (c) faces detected after improvement.

3.3.3 Dealing with bad Illumination conditions

There are many cases in which the face in the image is not properly illuminated and as a result, the face detection algorithm may not be able to detect such faces. This challenge can be overcome by applying some image preprocessing algorithms, like those mentioned in Chapter 2, on the input image before detecting faces. This increases the success rate of the face detection algorithm considerably.

Figure 3.6: Image before preprocessing (left); Image after preprocessing (right). Image courtesy: Yale Face database [1]

The figure 3.6 shows a badly illuminated face on the left, which was not detected by the face detection algorithm. The image on the right is the same image after preprocessing (High Frequency Emphasis filtering followed by histogram equalization) and now, the face detection algorithm was able to detect the face in the image.

CHAPTER 4

Face Recognition

Face recognition is the process of automatic identification or verification of a person from a digital image or a video frame from a video source by comparing selected facial features from the image and a facial database. Face recognition has been one of the most important and interesting field of research in the last two decades. This is due to the increasing need of automated recognition and surveillance systems, design of human-computer interface, etc. The face recognition algorithm is divided into three steps - **face detection**, where the face is detected in the input image, **facial feature extraction**, where the facial features like eyes, nose and mouth are extracted from the detected image and **face identification**, where the database is scanned and the correct match is identified using the extracted features. The following face recognition algorithm is based on the paper by Walaa Mohamed et al.[8].

4.1 Training

The training part of Face Recognition involves the creation of a database of several people, so that when one of them is spotted in an image frame, they can be identified. So, for the creation of this database, a lot of images of different persons are needed. Here, this algorithm is being implemented only for grayscale images, though it can be easily extended to color images. Because of this, all the frames are first converted to grayscale images.

Each grayscale image is enhanced using an image preprocessing algorithm - here High Frequency Emphasis Filtering (section 2.3) - so that the illumination and other effects are dealt with and further processing is easier. The face detection

algorithm, using `vision.CascadeObjectDetector` (section 3.2) in MATLAB, is then applied on the enhanced image and the position of the faces are located in the image. Each face located in the image is then cropped out and re-sized to a fixed size of 250×250 pixels using bi-cubic interpolation technique. These re-sized faces are stored along with a tag that relates itself with a person.

The MATLAB implementation of the training part of the face recognition algorithm is given in Appendix B.1.

4.2 Extraction of facial features

The next part of face recognition is the about the extraction of features from the face. It is these features that are actually used to differentiate one person from another. The facial features used in this recognition algorithm are the eyes, nose and mouth. The reason for choosing these as the facial features for recognition is that the other parts of the face do not account for the recognition process and under illumination conditions and other effects, these features remain unaffected. In the paper [8], a new method of extracting facial features was proposed, which was based on the feature location with respect to the whole face region.

Initially, MATLAB's `vision.CascadeObjectDetector` was used to locate the facial features - eyes, nose and mouth - in the face images (of constant size) detected and stored earlier. The coordinates of the right eye, left eye, nose and mouth were detected for a lot of face images of same size and using them, it can be concluded that the facial features for images of same size can be obtained at specific coordinates independent of the face image. Thus, for a particular face detection algorithm, the coordinates of the facial features can be considered constant.

The two face databases used for testing this face recognition algorithm were Extended Yale database B [1] and FERET database [5]. For images from yale database, the faces were of constant size of 192×168 pixels and the candidate

region for eyes was between rows 25 and 70 and between columns 1 and 70 for right eye, and between rows 25 and 70 and between columns 95 and 168 for left eye. The candidate region for nose was between rows 65 and 130 and between columns 40 and 125. The candidate region for mouth was between rows 125 and 185 and columns 20 and 150. For images from feret database, faces were detected, cropped and resized to a fixed size of 250×250 pixels and the candidate region for eyes was between rows 75 and 125 and between columns 40 and 130 for right eye, and between rows 75 and 125 and between columns 120 and 210 for left eye. The candidate region for nose was between rows 115 and 185 and between columns 75 and 175. The candidate region for mouth was between rows 170 and 240 and columns 70 and 180.

Figure 4.1: Face image from yale database and extracted features (left); Face image from FERET database and extracted features (right). Image courtesy: Yale Face database [1] and FERET database [5]

The figure 4.1 shows face images from Yale and FERET databases and the extracted facial features using the above mentioned coordinates.

4.3 Face Identification

The final part of the face recognition algorithm is Face Identification, where given an image, the person in the image has to be identified using the training database. This starts with the application of image preprocessing algorithm on the test image followed by detection of face. The detected face is then resized to

the fixed size (250×250 pixels) and the facial features are extracted using the coordinates of the features (mentioned in section 4.2).

Now the extracted facial features have to be compared with the facial features of the faces stored in the database and for this purpose, the `ssim` function in MATLAB is used. `ssim` or Structural Similarity Index function of MATLAB is used to measure the similarity of an image with respect to a reference image. A `ssim` value of 1 implies perfect similarity of the images and as the similarity between the images reduces, the `ssim` value drops below 1.

The structural similarity indices are calculated for the facial features of the test image and the facial features of each image stored in the database as shown below:

$$ssim\_total_i = ssim(re_{test}, re_i) + ssim(le_{test}, le_i) + ssim(n_{test}, n_i) + ssim(m_{test}, m_i) \tag{4.1}$$

where re_{test} is the right eye from the test image, le_{test} is the left eye from test image, n_{test} is the nose from test image, m_{test} is the mouth from test image, re_i, le_i, n_i, m_i are the corresponding features from the i^{th} image in the database. The image in database with maximum $ssim\_total_i$ value is identified as the closest match for the given test image. The MATLAB implementation of the face identification part is given in Appendix B.2.

CHAPTER 5

Experiments and Results

The Face Recognition Algorithm discussed in chapter 4 was tested using the Extended Yale Database [1] and FERET Database [5]. The Yale database consists of face images of 38 different persons with 60 images for each person, thus the database totally consists of 2280 images. All the images in the yale database are frontal face images under various illumination conditions. The FERET database consists of images of 994 different persons with a total of 5831 images. These images consist of pose variations, expressions and people with and without glasses. The figure 5.1 shows the image at different stages of the recognition algorithm.

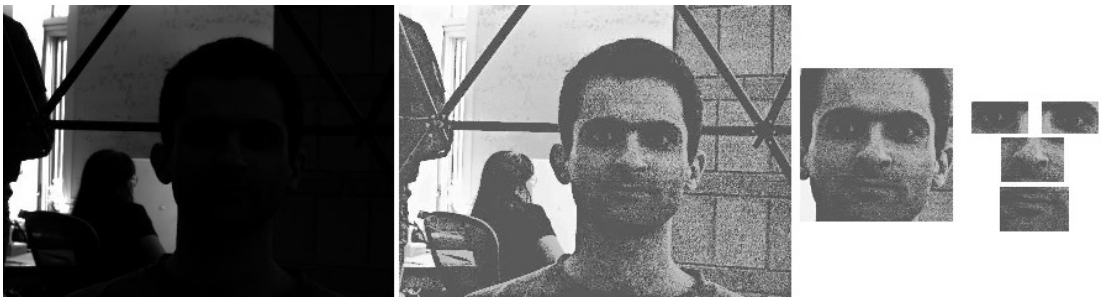

Figure 5.1: Given image, enhanced image, detected and resized face and extracted facial features. Image courtesy: Yale Face database [1]

While using Yale database for testing the recognition algorithm, the first image of each person is used for training and the remaining images were used for testing. This is the case of single enrollment. Similarly, the recognition algorithm was also tested using multiple enrollments, in which the multiple images of each person was used for training and the algorithm was tested on the remaining images. The results of testing the Recognition algorithm is given in Table 5.

Table 5.1: Results of Recognition Algorithm tested on Yale database

No. of Enrollments	Success %
1	84.83
2	90.03
3	94.31
4	98.52
5	99.42
6	99.90

While using FERET database for testing the recognition algorithm, success percentage with one enrollment was less than 30% and with multiple enrollments, it increased, but not more than 50%. This is because most of the faces in the database had pose variation, which the algorithm could not handle. When the images that succeeded and failed were examined and it was found that most of the straight faces were recognized and only the faces with pose variations failed.

CHAPTER 6

Scope for future work

The face recognition algorithm discussed is not suitable for pose variations and side faces. Thus, the algorithm can be modified to incorporate a solution to this problem. Spoofing is one major threat to current face recognition systems, where a picture of a person can be used to cheat the recognition system. Hence, depth estimation of face is one important research area which could make the current recognition algorithms robust and efficient. Also, the recognition algorithm can be made to handle optical and motion blur, which are very common in most images.

APPENDIX A

MATLAB implementation of Preprocessing Algorithms

A.1 Multi-Scale Retinex with Color Restoration

```
function [I_msr,I_msrcr] = msrcr(I)
%MSRCR Multi Scale Retinex with Colour Restoration

    I=double(I);
    [m,n]=size(I(:,:,1));
    I_msr=zeros(m,n,3);
    r=161;
    o=floor((r+1)/2);
    F=zeros(r,r,3);
%     surround space constant
    Cn=[250,250,250];
%     Creation of the gaussian surround function
    for s=1:3
        for i=1:r
            for j=1:r
                F(i,j,s) =exp(-((i-o)^2+(j-o)^2)/(Cn(s))^2);
            end
        end
    F(:,:,s)=F(:,:,s)/(sum(sum(F(:,:,s))));
    end

%     Multi-scale retinex
    G=zeros(m,n);
    for s=1:3
        K(:,:,1)=imfilter(I(:,:,s),F(:,:,1));
        K(:,:,2)=imfilter(I(:,:,s),F(:,:,2));
        K(:,:,3)=imfilter(I(:,:,s),F(:,:,3));
        for i=1:m
            for j=1:n
                G(i,j)=1/3*((log(I(i,j,s)+1)-log(K(i,j,1)+1)
                   )+(log(I(i,j,s)+1)-log(K(i,j,2)+1))+(log(
                   I(i,j,s)+1)-log(K(i,j,3)+1)));
            end
        end
%       Remapping intensity values to the range 0 to 255
        G_min=min(min(G));
```

```
        G_max=max(max(G));
        I_msr(:,:,s)=(G-G_min)*255/(G_max-G_min);
    end

%     Color Restoration
    C=zeros(m,n,3);
    beta=46;
    alpha=125;
    b=30;
    gain=192;
    for s=1:3
        for i=1:m
            for j=1:n
                C(i,j,s)=beta*(log(alpha*I(i,j,s)+1)-log(sum
                   (I(i,j,:))+1));
            end
        end
    end
    I_msrcr=zeros(m,n,3);
    for s=1:3
        for i=1:m
            for j=1:n
        I_msrcr(i,j,s)=gain*((I_msr(i,j,s)*C(i,j,s))+b);
            end
        end
        Imin=min(min(I_msrcr(:,:,s)));
        Imax=max(max(I_msrcr(:,:,s)));
        I_msrcr(:,:,s)=(I_msrcr(:,:,s)-Imin)*255/(Imax-Imin)
           ;
    end
    I_msrcr=uint8(I_msrcr);
    I_msr=uint8(I_msr);
end
```

A.2 High Frequency Emphasis Filtering

```
function [I_out] = high_freq_emp(I_in)
% High Frequency Emphasis Filtering

    [f1,f2] = freqspace(size(I_in),'meshgrid');
    D = sqrt(f1.^2 + f2.^2);
%     Butterworth high pass filter
    bw_hp=1./((1+0.1./D).^2);
%     high frequency emphasis filter
    hfe_filter=0.5+0.75*bw_hp;
    hfe_filter=ifftshift(hfe_filter);

% Filtering in the frequency domain
    I_f=fft2(I_in);
```

```
    I_f=I_f.*hfe_filter;
%     Convertion back to spacial domain
    I2=uint8(ifft2(I_f));

% histogram equalization of filtered image
    I_out=histeq(I2);

end
```

A.3 Contrast and Color Enhancement

The following is the MATLAB implementation of Contrast and Color Enhancement from the paper by Mark DeNies [2].

```
% Contrast enhancement function derived from the paper
% "Contrast and Color Enhancement"
% by Mark DeNies, DeNies Video Software. 2012

function [outImg] = ContrastEnhancement(fname)
    Img = imread(fname);
    HSV = rgb2hsv(Img);
    % amplify the intensity
    V = HSV(:,:,3);
    V = powerLaw(V);
    [maxVal vOut] = enhance(3.0, V, max(V(:)), 4);
    V = maxVal - vOut;
    V = powerLaw(V);
    [maxVal vOut] = enhance(3.0, V, maxVal, 4);
    V = maxVal - vOut;
    %put the modified intensity back
    HSV(:,:,3) = V ./ maxVal;
    % amplify the saturation
    S = HSV(:,:,2);
    [maxVal sOut] = enhance(2.0, S, max(S(:)), 8);
    S = maxVal - sOut;
    [maxVal sOut] = enhance(2.0, S, maxVal, 8);
    S = maxVal - sOut;
    % put the modified saturation back
    HSV(:,:,2) = S ./ maxVal;
    % reconstruct the RGB image
    outImg = hsv2rgb(HSV);
end

function [oImg] = powerLaw(I)
    maxVal = max(I(:));
    iAvg = mean(I(:));
    power = max(0.6667, min(1.5,(iAvg / (maxVal/2))^2));
    oImg = maxVal * (I./maxVal).^power;
```

```
end

function [maxV outImg] = enhance(delta, Img, maxVal,
   maxStretch)
    outImg = Img;
    [h,w] = size(Img);
    DeltaH = ones(h,w);
    DeltaApply = zeros(h,w);
    Range = maxVal / maxStretch;
    for i=1:128
%         calculate current threshold
        sVal = (maxVal*i) / 128;
%         Set B = true for all pixels > sVal
        B = Img > sVal;
%         Set H = hills, labeled 1:nH
        [H nH] = bwlabel(B,8);
        for j=1:nH
%             fetch current Hill
            HList = find(H == j);
%             find it's maximum value
            pMax = max(Img(HList));
            limit the stretching
            DeltaMax = min(delta, Range/(pMax - sVal));
%set List = those pixels which can be stretched more: those
%whose deltaH<delta. Set DeltaApply=the amount to stretch
%each of them: the remaining delta, or DeltaMax.
            L = HillList(find(DeltaH(HList) < delta));
            if ~isempty(List)
                DeltaApply(L) = min(DeltaMax, delta-DeltaH(L
                   ));
                outImg(L) = outImg(L) + DeltaApply(L) .* (
                   Img(L) - sVal);
                DeltaH(L) = DeltaH(L) + DeltaApply(L);
            end;
        end;
    end;
    maxV = max(outImg(:));
end
```

APPENDIX B

MATLAB implementation of Face Recognition Algorithm

B.1 Training

```
clear;
close all;
clc;

% Size of the resized image
S=250;
% For storing the detected faces
I=zeros(S,S,994);
flag=zeros(994,1);
% For storing the variance of the detected faces
variance=zeros(994,1);

for i=1:994
%     Reading image and converting to grayscale
    img=rgb2gray(imread(sprintf('E:\\BTP\\feret_db\\data\\
       images\\person (%d)\\1.ppm',i)));
%     Enhancement and detection of face
    [img,flag(i)]=face_detect(img);
%     Storing the detected face
    I(:,:,i)=img;
end
```

The face_detect function used in the above code is defined below:

```
function [Out,flag] = face_detect(In)
%FACE_DETECT - Used to detect faces in the given image.

    %Detect objects using Viola-Jones Algorithm
    %To detect Face
    FDetect = vision.CascadeObjectDetector;

%   Enhancing the image
    In_e=enhance(In);
```

```
    flag=0;

%   Returns Bounding Box values based on number of objects
    BB = step(FDetect,In_e);

    for i = 1:size(BB,1)
%                   Avoiding non-face detections
        if(BB(i,3)<200)
            continue;
        else
            Out=In(BB(i,2):BB(i,2)+BB(i,4),BB(i,1):BB(i,1)+
               BB(i,3));
%                   flag is set to 1 if face is detected
            flag=1;
            break;
        end
    end

    if(flag==1)
%     Resizing the detected face to 250x250 pixels
        Out=imresize(Out,[250 250],'bicubic');
    else
        Out=uint8(zeros(250));
    end
end
```

The enhance function used in the face_detect function is the high frequency emphasis filtering algorithm followed by histogram equalization. It can be referred from Appendix A.2.

B.2 Face Identification

The following is the MATLAB code used for testing the Recognition algorithm on FERET database.

```
clear;
close all;
clc;

% Loading the enrolled images and data.
load('database.mat');
success=0;fail=0;
```

```
for i=1:994
    if(n_img(i)<=1)
        continue;
    end
    for j=2:n_img(i)
%               Reading the input test image
        I_test=double(imread(sprintf('E:\\BTP\\feret_db\\
           data\\images\\person (%d)\\%d.jpg',i,j)));
%       Enhancing, detecting and resizing the face in the
   test image
                [I_test,f]=face_detect(I_test);
                if(f==0)
                        continue;
                end
%       Extraction of facial features from test image
        re_test=I_test(75:125,40:130);
        le_test=I_test(75:125,120:210);
        n_test=I_test(115:185,75:175);
        m_test=I_test(170:240,70:180);

        e_dist=zeros(994,4);
        eucl_dist=zeros(994,1);

        for k=1:994
            if(flag(k)==0)
                continue;
            end
%               Extraction of features from database image
            re=I(75:125,40:130,k);
            le=I(75:125,120:210,k);
            n=I(115:185,75:175,k);
            m=I(170:240,70:180,k);

%               Calculating ssim values
            e_dist(k,1)=ssim(re_test,re);
            e_dist(k,2)=ssim(le_test,le);
            e_dist(k,3)=ssim(n_test,n);
            e_dist(k,4)=ssim(m_test,m);

            eucl_dist(k)=sum((e_dist(k,:)));
        end
%               Face Identification
        [a,b]=max(eucl_dist(1:994));

        if(b==i)
            success=success+1;
            imwrite(uint8(I_test),sprintf('E:\\BTP\\
               ferret_db\\s_ssim\\%d_%d.jpg',i,j));
        else
            fail=fail+1;
```

```
            imwrite(uint8(I_test),sprintf('E:\\BTP\\
                ferret_db\\f_ssim\\%d_%d.jpg',i,j));
        end
    end
end
success_percent=success/(success+failure)
```

REFERENCES

[1] The exteded yale face database b. 2005. `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`.

[2] Mark DeNies. Contrast and color enhancement. *DeNies Video Software*, 2012.

[3] A. C. Hurlbert. The computation of color. *PhD thesis, Massachusetts Institute of Technology*, 1989.

[4] Edwin H. Land. An alternative technique for the computation of the designator in the retinex theory of color vision. *Proc Natl Acad Sci U S A*, 1986.

[5] National Institute of Standards and Technology. The color feret database. 2005. `http://www.nist.gov/itl/iad/ig/colorferet.cfm`.

[6] John D.Austin Robert Cromartie Ari Geselowitz Tery Greer Bartter Haar Romeny John B.Zimmerman Stephen M.Pizer, E.Philip Amburn and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics and Image processing*, pages 335–368, 1987.

[7] Paul Viola and Michael J.Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[8] Moheb Girgis Walaa Mohamed, Mohamed Heshmat and Seham Elaw. A new method for face recognition using variance estimation and feature extraction. *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, 2:134–141, 2013.

[9] Yi-Qing Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 2014.

[10] Zhiyuan Xu, Xiaoming Liu, and Xiaonan Chen. Fog removal from video sequences using contrast limited adaptive histogram equalization. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4, Dec 2009.

[11] Ming-Hsuan Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, Jan 2002.

[12] Cha Zhang and Zhengyou Zhang. A survey of recent advances in face detection. *Microsoft Research*, 2010.