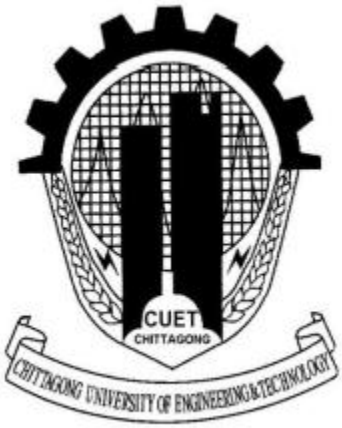
An Efficient Framework for Detecting Face Region Based on Skin Color Modeling and SVM Classifier



This thesis is submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering.

Arpita Chowdhury
ID:1004036

Supervised by
Prof. Dr. Kaushik Deb
Professor
Department of Computer Science & Engineering (CSE)
Chittagong University of Engineering & Technology (CUET)

Department of Computer Science & Engineering Chittagong University of Engineering & Technology
Chittagong-4349, Bangladesh.

The thesis titled **"An Efficient Framework for Detecting Face Region Based on Skin Color Modeling and SVM Classifier"** submitted by Roll No. 1004036, Session 2013-2014 has been accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering (CSE) as B.Sc. Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

1 _____ Chairman

Dr. Kaushik Deb

Professor

Department of Computer Science & Engineering (CSE)

Chittagong University of Engineering & Technology (CUET)

2._____ Member

Dr. Mohammed Moshiul Hoque

Professor & Head (Ex-officio)

Department of Computer Science & Engineering (CSE)

Chittagong University of Engineering & Technology (CUET)

3._____ Member

Dr. Md. Ibrahim Khan (External)

Professor

Department of Computer Science & Engineering (CSE)

Chittagong University of Engineering & Technology (CUET)

Statement of Originality

It is hereby declared that the contents of this thesis is original and any part of it has not been submitted elsewhere for the award if any degree or diploma.

Signature of the Candidate

Date:

Acknowledgment

First and foremost, I would like to thank God Almighty for giving me the strength to finish this work. The satisfaction that accompanies the successful completion of this thesis would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I am grateful to my honorable project Supervisor Prof. Dr. Kaushik Deb, professor of the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, for the guidance, inspiration and constructive suggestions which were helpful in the preparation of this project. I would also like to extend my gratitude to all of my teachers for their valuable guidance in every step of mine during the four years of learning stage. Finally, I would like to thank my parents, friends, senior, juniors and the staff members of the department for their valuable suggestions and encouragement that had helped to complete my project successfully.

Abstract

Face is the basic inimitable symbol of personal identity. Detection of the human face is an essential step in many computer vision and biometric applications such as automatic face recognition, video surveillance and human computer interaction. This project provides a scheme for face detection using skin color modeling and then classifying through SVM classifier. Firstly skin segmentation is performed by using Hue of HSV color model and Cb and Cr of YCbCr color model. Labeling, filtering and morphological operations are performed for removing noise. After obtaining the face region, SVM classifier is used for verifying the images. For extracting features, SFTA texture extractor is used. In SFTA method, boundary fractal dimension, mean gray level and size is used for features. After extracting features of face and non-face image, training and testing by using SVM classifier is performed and classifies the images into 2 classes: class 1 is face and 2 is non-face. The detection accuracy of my proposed method is about 96.23%. The method works very successful with single face that is insensitive to pose, illumination and skin color. But when the number of faces increases and the background is very much similar to skin color, detection rate is going to decrease. The classification accuracy is about 88.17% that classify the face image with different pose and illumination.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Image processing is a method to convert an image into digital form that performs some operations in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics or parameters related to that image. The two types of methods used for Image Processing are Analog and Digital Image Processing. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Digital Processing techniques help in manipulation of the digital images by using computers

As a field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms that can be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing.

1.2 Face Detection

Face is the primary media for human interaction. It is the basic symbol of personal identity. It is natural for the face to be the primary candidate for human machine interface. Human have good recognition capabilities of faces and anything cannot affect this capability. The main reason for this is the high degree of interconnectivity, acquisition skills and abstraction capabilities of the human nervous system. Among many biometric identifiers like face, fingerprint, iris, palm veins, voice, DNA face is the most natural means of identifying people.

Initially, detecting face region is a crucial step in face recognition systems with the purpose of localizing and extracting the face region from the background. Two main

approaches are used in face detection. The first approach is image-based, a pattern recognition problem treated and solved by many methods (Eigenfaces and Neural networks). The second approach is feature-based, which uses invariant features of the face such as skin color, eyes, eyebrows, nose [1]. For detecting face there are various algorithms including skin color based algorithms. Color is an important feature of human faces. Skin color has proven to be a useful and robust cue for face detection, localization and tracking. Using skin-color as a feature for tracking a face has several advantages. Color processing is much faster than processing other facial features. Human skin has a color that is generally distinct from the colors of other objects from images. Also it is independent of face orientation and scale, geometric variation of the face shape. This property makes motion estimation much easier.

For biometric systems that use faces as non-intrusive input modules, it is imperative to locate faces in a scene before any recognition algorithm can be applied. For recognizing, facial features are extracted and trained by a classifier like SVM, Neural Network, AdaBoost and so on. When the input data to an algorithm is too large to be processed, then it can be transformed into a reduced set of features (features vector). This process is called feature extraction. To detect facial features accurately for applications such as digital cosmetics faces need to be located and registered first to facilitate further processing. It is evident that face detection plays an important role for the success of any face processing systems.

1.3 Motivation

Detection of the human face is an essential step in many computer vision and biometric applications such as automatic face recognition, video surveillance, human computer interaction (HCI) and large-scale face image retrieval systems. Due to the individual differences in face detection, a hundred percent accurate face detection software has not yet been developed. Many existing systems use a window-based classifier that scans the entire image for the presence of the human face and such systems suffers from scale variation, pose variation, illumination changes. So, the proposed method is based upon skin tone information of the input color image that is insensitive to lighting, pose,

expressions. After detecting face the verification of the face is a crucial step for further processing. In several methods the false detection rate is high that have not contain any face. So, in the proposed method finally verification of the face image is performed.

1.4 Applications

Human face localization and detection is often the first step in applications such as video surveillance, human computer interface, face recognition, video chat service, and photography and image database management. Detecting face region is a crucial step in face recognition systems with the purpose of localizing and extracting the face region from the background. In identity management solution, surveillance and security systems, law enforcement, criminal investigation face detection is necessary.

1.5 Challenges

Face detection is a challenging task because of variability in scale, location, facial expression, illumination, orientation, pose etc. The factors that give rise to certain challenges are as follows:

- **Pose:** The images of a face vary due to the relative camera-face pose (frontal, 45°, profile, and upside down).
- **Presence or absence of structural components:** Facial features such as beards, mustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape color and size.
- **Facial expression:** The appearance of faces is directly affected by a person's facial expression like smiling, crying, fearing etc.
- **Occlusion:** Faces may be partially or fully occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.
- **Image Conditions:** When the images is formed, factors such as lighting (Spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.
- **Image Orientation:** Face images directly vary for different rotations about the camera's optical axis.

- **Face size:** In the image, human face can exist with different sizes. Sometimes, size of existed face is too small or big to describe clearly different facial components information.

1.6 Contributions of the Work

- Detection of face region from any given face image is performed by skin color modeling. By using hue, Cb and Cr for thresholding, skin detection is performed quite accurately. The data set is randomly chosen by us and that includes several kinds of images in indoor and outdoor background. The detection method is invariant to facial expression and illumination.
- Classification of the face image is done by using SVM. After obtaining face image, either the region is face or not is performed. The features fractal dimension, mean and area give accurate classification of face and non-face image.

1.7 Organization of the Report

The report is organized in the following manner: In chapter 2, different approaches used for face recognition are described. It also includes the basic idea of color models, morphological analysis, connected component analysis, region properties, texture properties, SVM classifier, general framework in face detection and related work. In chapter 3, methodology of the proposed system is described which include skin color segmentation, face candidate localization, rejection of non-human face and verification of the face region. In chapter 4, the implementation of the proposed method is given. In chapter 5, experimental result & analysis is given. Conclusion and future recommendation is given in chapter 6.

1.8 Conclusion

In the chapter, a basic introduction of the thesis is represented. In the next chapter, the basic concepts related with the proposed method are described.

Chapter 2

Literature Review

2.1 Introduction

This Chapter describes a brief description of theories related to the proposed method such as color model, morphological operation, SVM classifier etc. General framework in face detection is also included. At last related work of the proposed method is included in which limitation of the work is focused.

2.2 Different Approaches of Face Detection

Face detection methods are divided into four categories these are given in table 2.1.

Table 2.1: Face Detection Methods

Feature -Based Method	Bottom-up methods
Appearance-Based Method	Eigenfaces Neural Network based approach Statistical Model based Approach or SVM based approach Sparse Network of Windows Naïve Bayes Classifier
Knowledge -Based Method	Top-down methods
Template Matching Method	Predefined face templates Active Shape Model

2.2.1 Feature Based Approach

The feature based approaches aim to detect invariant face features. These are structural features of a face that exist even when the pose, viewpoint or lighting conditions vary.

The main advantage of the feature oriented face detection approaches consists in the fact that these features are invariant to rotation changes. Their main drawback is the difficulty to locate facial features in a complex background. Representative features are eigen features, Haar-like features, edge features and so on. Each feature is briefly explained below.

2.2.1.1 Eigen feature

Eigen faces have long been used for face detection and recognition purposes. Principal component analysis (PCA) is performed on a set training face images to obtain the eigen vectors, which are called eigen faces. The projections of the mean 3 adjusted face images along the eigen vectors are used as the eigen features for training and classification purposes.

2.2.1.2 Haar Feature

Haar-like features are the digital image features used in object recognition. Viola and Jones were the first to use Haar wavelets as a basis for developing Haar-like features. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in these regions and calculates the difference between them. This difference is then used to categorize subsections of an image. Given that in all faces the region of the eyes is darker than the region of the cheeks, a common Haar-like feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of the rectangle is defined relative to a detection window that acts like a bounding box to the face. A window of the target size is moved over the input image, and the Haar-like feature is calculated for each subsection of the image. This difference is then compared to a learned threshold to separate non-faces from faces. The key advantage of a Haar-like feature over most other features is its calculation speed, since a Haar-like feature of any size can be computed in a constant time by using integral images.

2.2.1.3 Edge Feature

Edge features are used to compute the edges of the image. The Canny edge detector, Robert's edge detector and Sobel edge detector are commonly used to extract edge features. These edges normally represent the contour lines of the face including eyes, eyebrows, lips, nose and face boundary.

2.2.1.4 Skin Color

Detection of skin color in color images is a very popular and useful technique for face detection. Color is an important feature of human faces. Using skin-color as a feature to detect a face has several advantages. Color processing is much faster than processing other facial features. Human skin has a color that is generally distinct from the colors of other objects from images. Also it is independent of face orientation and scale, geometric variation of the face shape [1]. However, color is not a physical phenomenon related to the spectral characteristics of electromagnetic radiation in the visible wavelengths striking the retina. Specifically said, color models convert the color image into an appropriate color space, such as RGB, HSV, YCbCr or YIQ and fix the threshold value to detect skin color. Those color spaces are more robust to the lighting conditions than the RGB color space. In the case of HSV and YCbCr, by separating luminance from chrominance, makes the color spaces luminance independent and more adequate than RGB for skin color segmentation.

2.2.2 Appearance Based Approach

In appearance-based method, certain window is usually scanned through the image and then related part of an image grouped into two patterns as face or non-face. Generally a given image is resized to detect the faces which have different sizes. Alternatively the size of the sample can be calibrated appropriately to detect faces with different scales. Face detection by using this method, is based on finding the differences between face and non-face patterns. To distinguish between face and non-face patterns, many pattern recognition techniques have been used .The following subsections are devoted to well-known appearance based methods.

2.2.2.1 Linear Subspace Methods

Images of human faces lie in a subspace of the overall image space. To represent this subspace, one can use neural approaches but there are also several methods more closely related to standard multivariate statistical analysis which can be applied. There are many famous techniques in this category that has been used for different analysis, a few of these methods includes principal component analysis (PCA), linear discriminant analysis (LDA), and factor analysis (FA).

2.2.2.2 Neural Network Based Approach

Neural networks have become a popular technique for pattern recognition problems, including face detection. Modular architectures, committee ensemble classification, complex learning algorithms, auto associative and compression networks, and networks evolved or pruned with genetic algorithms are all examples of the widespread use of neural networks in pattern recognition. For face recognition, this implies that neural approaches might be applied for all parts of the system. This system is very effective in detecting frontal faces.

2.2.2.3 Statistical Model Based Approach

Statistical model-based approaches use a statistics-based classifier, such as a support vector machine (SVM), to classify the candidate region as a face or a non-face. Specifically, the SVM trains a classifier by solving the optimization problem to decide which instances of the training data set are support vectors. These support vectors are the necessarily important instances to form the SVM classifier [2]. The learned SVM classifier is further employed in any classification task.

2.2.3 Knowledge Based Approach

The knowledge-based methods encode human knowledge of what constitutes a typical face, usually the relationships between facial features. A face is represented using a set of human-coded rules. These rules are then used to guide the face search process. The advantages of the knowledge-based techniques are: the easy rules to describe the face features and their relationships, and the good results obtained for face localization in

uncluttered background. Their disadvantages are: the difficulty to translate the human knowledge in rules precisely and the difficulty to extend these methods to detect faces in different poses, respectively.

2.2.4 Template Matching Based Approach

The template matching based techniques use stored face templates. Usually, these approaches use correlation operations to locate faces in images. The templates are hand coded, not learned. The correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently on a given input image. This approach is simple to implement however it lacks the capacity of detecting face with variations in scale, pose and shape.

2.3 Color Models

Color is a powerful descriptor that often simplifies object identification. It is a fundamental attribute of human viewing experience. To work with color, it must be defined mathematically. Here color model comes in action. A color model is an abstract mathematical model that describes the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted, the resulting set of colors is called color space. The purpose of a color model is to facilitate the specification of colors in some standard way. Some frequently used color models are RGB, HSI or HSV, YUV or YCbCr and CMYK.

2.3.1 RGB Color Model

The RGB color space consists of the three additive primaries: red, green and blue. Spectral components of these colors combine additively to produce a resultant color. The RGB model is represented in Figure 2.1 by a 3-dimensional cube with red green and blue at the corners on each axis. Black is at the origin. White is at the opposite end of the cube. The gray scale follows the line from black to white. In a 24-bit color graphics system with 8 bits per color channel, red is (255, 0, 0). On the color cube, it is (1, 0, 0).

The importance of the RGB color model is that it relates very closely to the way that the human eye perceives color.

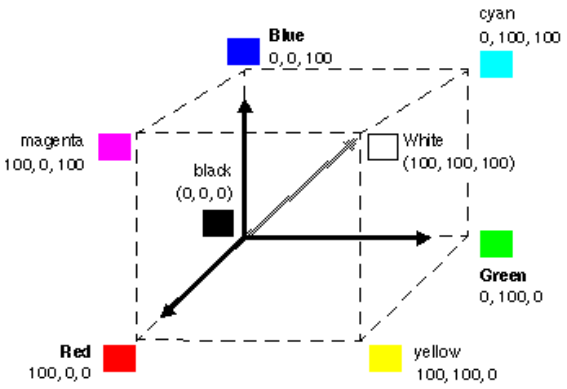


Figure 2.1: RGB Color Space

The RGB color space is the most prevalent choice for computer graphics because color displays use red, green and blue to create desired color. Therefore the choice of RGB color space simplifies the architecture and design of the system. Also, a system that is designed using RGB color space can take advantage of a large number of existing software routines, since this color space has been around for a number of years.

But the color space is not ideal for all applications. The red, green and blue color components are highly correlated. This makes it difficult to execute some image processing algorithms. Many processing techniques, such as histogram equalization, work on the intensity component of an image only.

2.3.2 HSV Color Model

HSV is a color model that describes colors (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness (value or luminance). The cone of HSV color model is usually represented in the three-dimensional form. The saturation is calculated using the radius of the cone and value is the height of the cone. Figure 2.2 represents HSV color model.

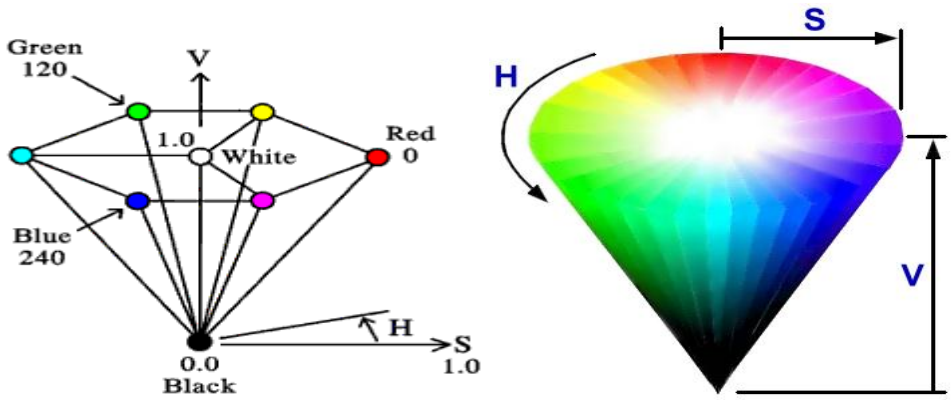


Figure 2.2: HSV Color Space

- **Hue** is expressed as a number from 0 to 360 degrees representing hues of red (starts at 0), yellow (starts at 60), green (starts at 120), cyan (starts at 180), blue (starts at 240) and magenta (starts at 300).
- **Saturation** is the amount of gray (0% to 100%) in the color.
- **Value** (or Brightness) works in conjunction with saturation and describes the brightness or intensity of the color from 0% to 100%.

The application of the cylindrical model of HSV color space is similar to the conical model. Calculations are done in a similar way. Theoretically, the cylindrical model is the most accurate form of HSV color space calculation. The HSV color space is quite similar to the way in which humans perceive color. The colors used in HSV can be clearly defined by human perception, which is not always the case with RGB or CMYK.

2.3.3 YCbCr Color Model

The YCbCr format is widely used for digital video. It is used as a part of the color image pipeline in video and digital photography systems. In this format, luminance information is stored as a single component (Y), and chrominance information is stored as two color difference components (Cb and Cr). Cb represents the difference between the blue component and a reference value, and Cr represents the difference between the red component and a reference value.

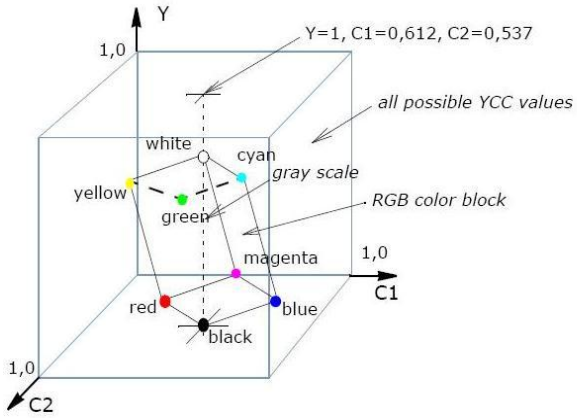


Figure 2.3: YCbCr Color Space

YCbCr color space is shown in Figure 2.3. YCbCr is used in JPEG image compression and MPEG video compression. It has the smallest overlap between the skin and non-skin data in under various illumination conditions. Y is the luma component and C_B and C_R are the blue-difference and red-difference chroma components. Y is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.

The YCbCr color space is commonly used in image processing as it separates the luminance, in Y component, form the chrominance described through Cb and Cr components.By separating the luminance component from chrominance, makes YCbCr color space luminance independent and more adequate than RGB model.

2.3.4 CMYK Color Model

The CMYK color model is a subset of the RGB model and is primarily used in color print production. In the print industry, cyan, magenta, yellow and black are used as the primary colors. CMYK is an acronym for cyan, magenta, and yellow along with black (noted as K). CMYK is the standard color model used in offset printing for full-color documents. Because such printing uses inks of these four basic colors, it is often called four-colorprinting. Many paint and draw programs can make use of either the RGB or the CMYK model. Figure 2.4 represents the CMYK color model.

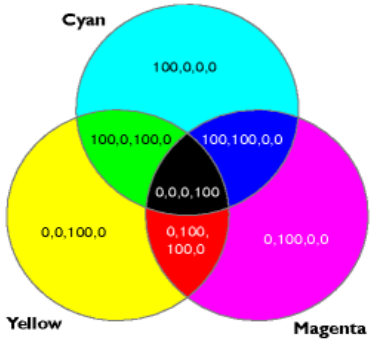


Figure 2.4: CMYK Color Model

2.4 Thresholding

Thresholding is the simplest method of image segmentation. Thresholding is used to create binary images from grayscale image. The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant or a white pixel if the image intensity is greater than that constant. It is an important step of color segmentation such as skin color segmentation. A grayscale image converted to binary image by thresholding is represented in Figure 2.5 & Figure 2.6.

Figure 2.5: Original Image

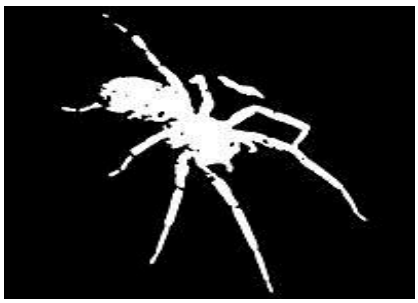


Figure 2.6: Thresholding Image

Thresholding techniques can be categorized into two classes:

- Global thresholding
- Local (adaptive) thresholding

Global thresholding: A single threshold value is used in the whole image. The global thresholding is simple and easy to implement, it has been a popular technique in many

years. It works well when the intensity distribution of objects and background pixels are sufficiently distinct.

Local thresholding: A threshold value is assigned to each pixel to determine whether it belongs to the foreground or the background pixel using local information around the pixel. Local thresholding works well in this case since the threshold value at any point depends on the properties of neighborhood of that point. In our real life we have images whose intensity distributions are not distinct. Local thresholding works well in this case since the threshold value at any point depends on the properties of neighborhood of that point. There are several algorithms of thresholding:

- Otsu Method
- Balanced histogram thresholding (BHT)

2.4.1 Otsu Method

Otsu Method is used to automatically perform clustering-based image thresholding or the reduction of a graylevel image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their intra-class variance (the variance within the class) is minimal, or equivalently that their inter-class variance is maximal.

In Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance defined as a weighted sum of variances of the two classes: The algorithm is as follows:

1. Compute histogram and probabilities of each intensity level.
2. Set up initial class probability and class mean.
3. For all possible thresholds up to maximum intensity, update class probability, class mean and Compute variance.
4. Desired threshold corresponds to the maximum variance.

5. Two maximum are computed.
6. Desired or optimum threshold is computed.

Multi Otsu method is multilevel thresholding. Multilevel thresholding is a process that segments a gray level image into several distinct regions. This technique determines more than one threshold for the given image and segments the image into certain brightness regions, which correspond to one background and several objects. The method works very well for objects with colored or complex backgrounds.

2.5 Morphological Analysis

The field of mathematical morphology provides a number of important image processing operations, including erosion, dilation, opening and closing. All these morphological operators take two pieces of data as input. One is the input image, which may be either binary or grayscale for most of the operators. The other is the structuring element. Structuring element is this that determines the precise details of the effect of the operator on the image. Structuring Element is an essential part of the morphological operations that is used to probe the input image. A structuring element is a matrix consisting of only 0's and 1's that can have any arbitrary shape and size. The pixels with values of 1 define the neighborhood. The center pixel of the structuring element, called the origin, identifies the pixel of interest - the pixel being processed. These pixels are also considered in dilation or erosion processing. Figure 2.7 illustrates a diamond-shaped structuring element.

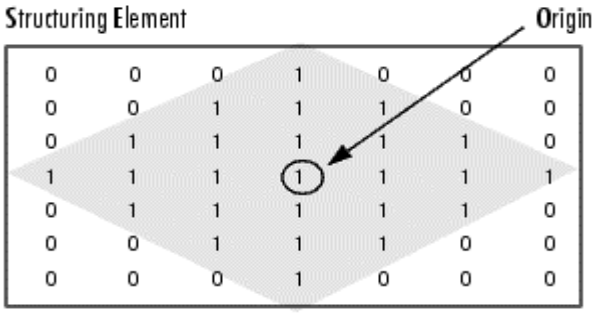


Figure 2.7: A Diamond-shaped Structuring Element

2.5.1 Dilation

Dilation is one of the basic operations in mathematical morphology. Originally developed for binary images, it has been expanded first to grayscale images, and then to complete lattices. The dilation operation usually uses a structuring element for probing and expanding the shapes contained in the input image. Dilate expands the selected area of an object and shrinks the background. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (white pixels, typically). Thus areas of foreground pixels grow in size while holes within those regions become smaller. The dilation operator takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element.

To compute the dilation of a binary input image by this structuring element, we consider each of the background pixels in the input image in turn. For each background pixel (which we will call the input pixel) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position. If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value. If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.

As an example of binary dilation, suppose that the structuring element is a 3×3 square, with the origin at its center, as shown in Figure 2.8.

1	1	1
1	1	1
1	1	1

Figure 2.8: A 3×3 Square Structuring Element

For our example a 3×3 structuring element, the effect of this operation is to set to the foreground color any background pixels that have a neighboring foreground pixel (assuming 8-connectedness). Such pixels must lie at the edges of white regions, and so the practical upshot is that foreground regions grow (and holes inside a region shrink). The effect of a dilation using this structuring element on a binary image is shown in Figure 2.9.

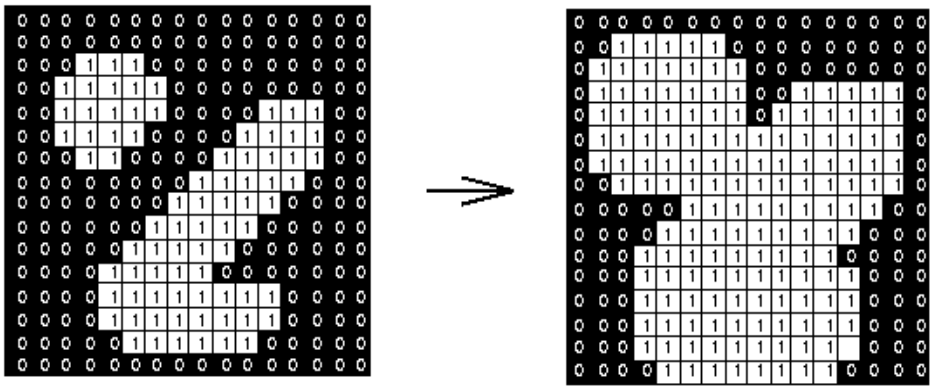


Figure 2.9: Effect of Dilation Using a 3×3 Square Structuring Element

2.5.2 Erosion

The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels (white pixels). Thus areas of foreground pixels shrink in size, and holes within those areas become larger.

To compute the erosion of a binary input image by this structuring element, we consider each of the foreground pixels in the input image in turn. For each foreground pixel (which we will call the input pixel) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value. The effect of an erosion using this structuring element on a binary image is shown in Figure 2.10.

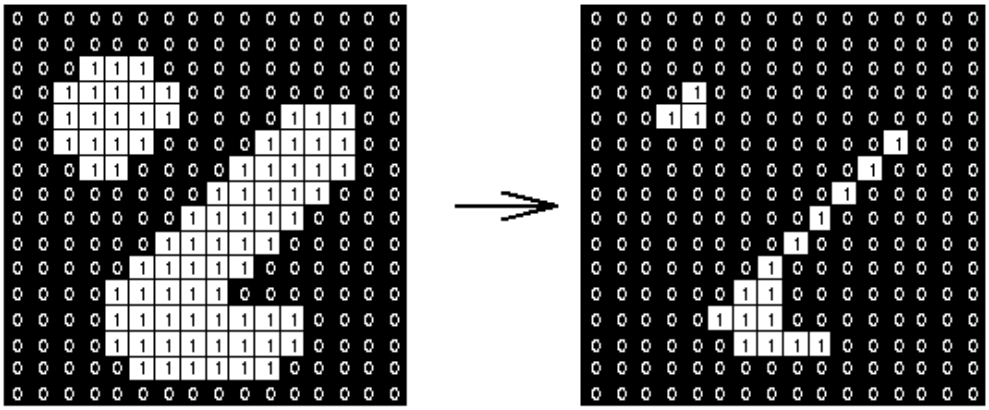


Figure 2.10: Effect of Dilation Using a 3×3 Square Structuring Element

2.5.3 Opening Operation

Opening is defined as an erosion followed by a dilation using the same structuring element for both operations. Opening smooth the contour of an object, removes small objects from the foreground pixels of an image, and placing them in the background. While erosion can be used to eliminate small clumps of undesirable foreground pixels quite effectively, it has the big disadvantage that it will affect all regions of foreground pixels indiscriminately. Opening gets around this by performing both an erosion and a dilation on the image.

While erosion can be used to eliminate small clumps of undesirable foreground pixels, *e.g.* `salt noise', quite effectively, it has the big disadvantage that it will affect all regions of foreground pixels indiscriminately. Opening gets around this by performing both an erosion and a dilation on the image.

The effect of opening can be quite easily visualized. Imagine taking the structuring element and sliding it around inside each foreground region, without changing its orientation. All pixels which can be covered by the structuring element with the structuring element being entirely within the foreground region will be preserved. However, all foreground pixels which cannot be reached by the structuring element without parts of it moving out of the foreground region will be eroded away. After the

opening has been carried out, the new boundaries of foreground regions will all be such that the structuring element fits inside them, and so further openings with the same element have no effect. The property is known as idempotence. The effect of opening on a binary image using a 3×3 square structuring element is represented in Figure 2.11.

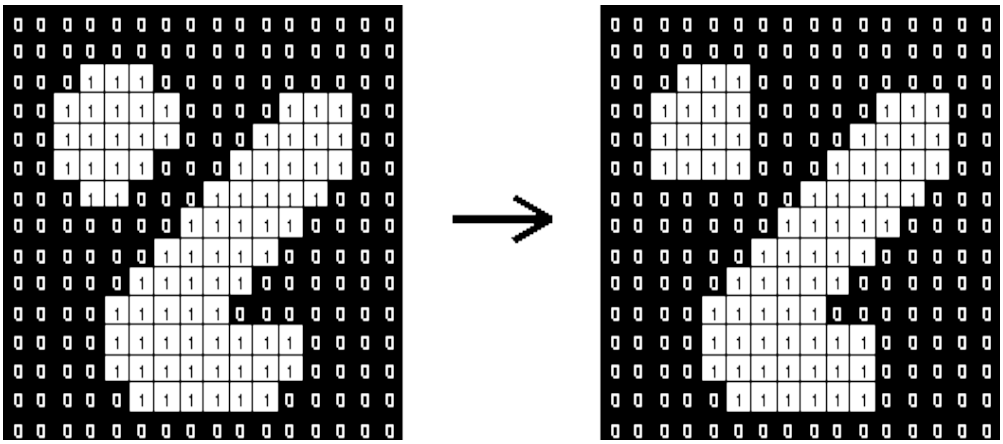


Figure 2.11: Effect of Opening Using a 3×3 Square Structuring Element

2.5.4 Closing Operation

The morphological close operation is a dilation followed by an erosion, using the same structuring element for both operations. Closing removes small holes in the foreground, changing small islands of background into foreground. One of the uses of dilation is to fill in small background color holes in images. One of the problems with doing this, however, is that the dilation will also distort all regions of pixels indiscriminately. By performing an erosion on the image after the dilation, *i.e.* a closing, we reduce some of this effect.

One of the uses of dilation is to fill in small background color holes in images, *e.g.* `pepper noise'. One of the problems with doing this, however, is that the dilation will also distort *all* regions of pixels indiscriminately. By performing an erosion on the image after the dilation, *i.e.* a closing, we reduce some of this effect.

The effect of closing can be quite easily visualized. Imagine taking the structuring element and sliding it around outside each foreground region, without changing its

orientation. For any background boundary point, if the structuring element can be made to touch that point, without any part of the element being inside a foreground region, then that point remains background. If this is not possible, then the pixel is set to foreground. After the closing has been carried out the background region will be such that the structuring element can be made to cover any point in the background without any part of it also covering a foreground point, and so further closings will have no effect. This property is known as idempotence. The effect of a closing on a binary image using a 3×3 square structuring element is illustrated in Figure 2.12.

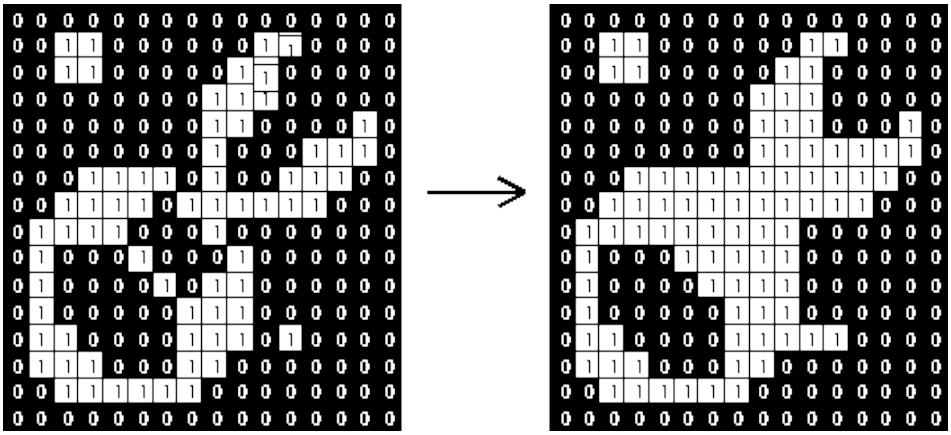


Figure 2.12: Effect of Closing Using a 3×3 Square Structuring Element

2.6 Flood Fill Operation

Flood fill operation determines the area connected to a given node in a multi-dimensional array. It is used in the "bucket" fill tool of paint programs to fill connected, similarly-colored areas with a different color. When it is applied on an image to fill a particular bounded area with color, it is also known as boundary fill.

Flood Fill Operation fills the holes in to binary image. We perform a flood-fill operation on the binary image to fill the holes created inside the face region, because of the eyes and lips or the holes created at the boundary after applying the dilation operation. These holes tend to separate all the objects from each other and therefore should be removed. For binary images, that changes connected background pixels (0s) to foreground pixels

(1s), stopping when it reaches object boundaries. Figure 2.13 represents the flood fill operation that fills the holes of the binary image.

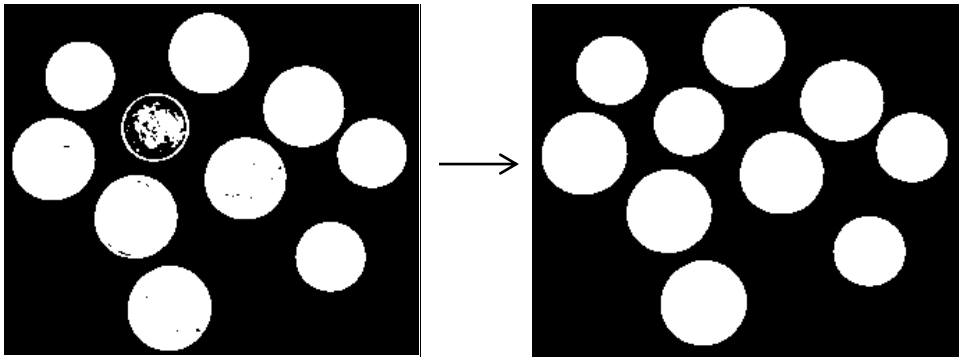
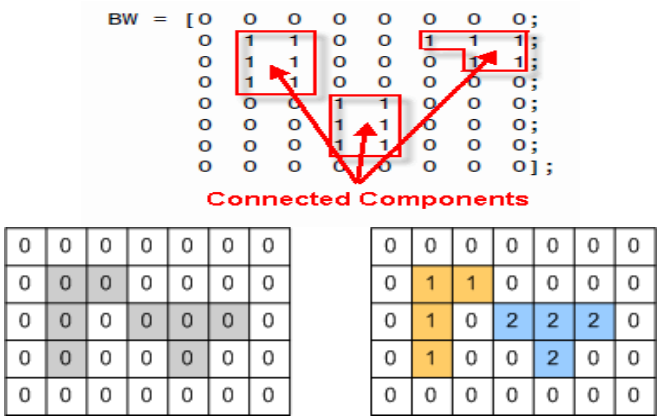


Figure 2.13: Effect of Flood Fill Operation

2.7 Connected Component Analysis

Connected-component labeling (alternatively connected-component analysis, blob extraction, region labeling, blob discovery or region extraction) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. Connected-component labeling is used to detect the connected region in binary image. Connected component labeling is a method for identifying objects in a binary image. A label matrix is an image, the same size as the input image, in which the objects in the input image are distinguished by different integer values in the output matrix.



0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	2	2	2	0
0	1	0	0	2	0	0
0	0	0	0	0	0	0

Figure 2.14: Connected Component Labeling

Objects in a binary image are pixels that are on, i.e., set to the value 1; these pixels are considered to be the foreground. When we view a binary image, the foreground pixels appear white. Pixels that are off, i.e., set to the value 0, are considered to be the background. When we view a binary image, the background pixels appear black. Two types of connectivity, 4-connected and 8-connected object are shown in Figure: 2.15.

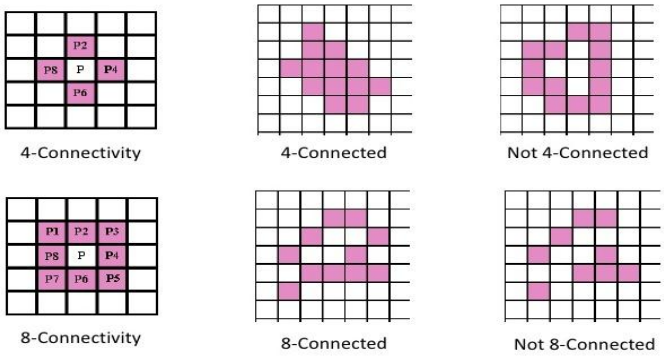


Figure 2.15: 4-Connected & 8- Connected Objects

2.8 Region Properties

There are many labeled regions in connected component analysis. A set of properties for each labeled region in the label matrix is called region properties. A binary image contains skin region. This step decides, which region is most probable human face region by using human face features method and the region properties of human face like area, eccentricity, centroid, bounding box, perimeter etc.

Area: Area is the scalar number which is the actual number of pixels in the region.

Euler Number: The number of holes in a region is computed using the Euler number of region. Euler number returns a scalar that specifies the number of objects in the region minus the number of holes in those objects. This property is supported only for 2-D label matrices

Centroid: It is a scalar number. Centroid is the center of mass of the region. Centroid returns a 1-by-*Q* vector that specifies the center of mass of the region. The first element

of Centroid is the horizontal coordinate (or *x*-coordinate) of the center of mass, and the second element is the vertical coordinate (or *y*-coordinate).

Bounding Box: Height to width ratio is known as Bounding Box property. It is the smallest rectangle of the region. It verifies the rectangularity of the region. BoundingBox is [ul_corner width], where ul_corner is in the form [x y z ...] and specifies the upper left corner of the bounding box which is in the form [x_width y_width ...] and specifies the width of the bounding box along each dimension.

Eccentricity: It is a scalar number. Eccentricity returns a scalar that specifies the eccentricity of the ellipse that has the same second-moments as the region. The eccentricity is the ratio of the distance between the foci of the ellipse and its major axis length. The value is between 0 and 1. (0 and 1 are degenerate cases. An ellipse whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment.)

Major Axis Length: It is a scalar number. The length (in pixels) of the major axis of the ellipse has the same normalized second central moments as the region. This property is supported only for 2-D input label matrices.

Minor Axis Length: It is a scalar number. The length (in pixels) of the minor axis of the ellipse has the same normalized second central moments as the region. This property is supported only for 2-D input label matrices.

Orientation: It is a scalar number. The angle (in degrees ranging from -90° to 90°) between the x-axis and the major axis of the ellipse that has the same second moments as the region. This property is supported only for 2-D input label matrices.

Perimeter: Returns a scalar that specifies the distance around the boundary of the region. Perimeter is measured by calculating the distance between each adjoining pair of pixels around the border of the region. If the image contains discontiguous regions, then perimeter returns unexpected results.

2.9 Feature

Features are distinctive properties of input patterns that help in differentiating between the categories of input patterns. Feature Extraction is transformation of input data into a set of features. Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative, non-redundant, facilitating the subsequent learning and generalization steps.

Feature extraction is related to dimensionality reduction. When the input data is too large to be processed and it is suspected to be redundant, then it can be transformed into a reduced set of features (also named a "features vector"). This process is called feature extraction. The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

2.9.1 Texture Features

Image texture, defined as a function of the spatial variation in pixel intensities (gray values), is useful in a variety of image processing applications. In general, the process of texture analyzing requires the calculation of various features for each texture.

Texture has role in domain-specific applications, such as in remote sensing, quality control and medical imaging. Additionally, texture information can be employed in image segmentation, by classifying image pixels with basis on surrounding texture information. Texture features are intended to capture the granularity and repetitive patterns of regions within an image.

The automatic analysis of images requires a first step of digitization which transforms the image into a two dimensional matrix of numbers. Each matrix entry is a pixel (picture element) which is the basic component of images with an entire value called graylevel. Usually the gray level, values are coded with an eight bit numbers, allows in distinguishing 256 gray levels from 0 (black) to 255 (white). Each pixel will contain two kinds of information. They are brightness value and locations in the co-ordinates that are assigned to the images. Application of image texture is the recognition of image regions using texture properties.

Figure 2.16: Texture Feature

Texture is a difficult concept to represent. The identification of specific textures in an image is achieved primarily by modeling texture as a two-dimensional gray level variation.

Approaches of texture feature extraction include:

- **Structured approach**: A structured approach sees an image texture as a set of primitive texels in some regular or repeated pattern. This works well when analyzing artificial textures.
- **Statistical approach**: A statistical approach sees an image texture as a quantitative measure of the arrangement of intensities in a region. In general this approach is easier to compute and is more widely used, since natural textures are made of patterns of irregular sub elements.

Methods of classifying textures include:

- Co-occurrence matrix
- Laws texture energy
- Wavelet Transform

Several Texture feature algorithm:

- Gabor Filter
- GLCM (Gray Level Co-occurrence Matrix
- SFTA (Segmentation-based Fractal Texture Analysis)
- FFS (Fast Fractal Stack)

2.9.2 SFTA (Segmentation-based Fractal Texture Analysis)

SFTA is an efficient texture feature extraction method. SFTA is faster than Filter bank-based methods (Gabor feature) and Haralick descriptors (GLCM) with respect to feature extraction time. It has shown superior performance with respect to image classification accuracy. GLCMs have to consider multiple orientations and scales that require a big volume of images and the computational cost can be critical. SFTA deal deals with computational cost. Fast Fractal Stack (FFS) feature extraction method does not perform well for all texture datasets [17].

2.9.2.1 Fractal

A fractal is a never-ending pattern. Fractals are infinitely complex patterns that are self-similar across different scales. They are created by repeating a simple process over and over in an ongoing feedback loop. Geometrically, they exist in between our familiar dimensions. Fractal patterns are extremely familiar, since nature is full of fractals. For instance: trees, rivers, coastlines, mountains, clouds, seashells, hurricanes, etc. A fractal is a natural phenomenon or a mathematical set that exhibits a repeating pattern that displays at every scale. If the replication is exactly the same at every scale, it is called a self-similar pattern.

2.9.2.2 Fractal Dimension

Fractal dimension is the measure of how complicated a self-similar figure is. It is an index for characterizing the fractal pattern. In the image analysis paradigm, fractal dimension measurements can be used to estimate and quantify the complexity of the shape or texture of objects. Fractal geometry involves various approaches to define fractional dimensions, where the most common is the Hausdorff's dimension. A line has dimension 1, a plane dimension 2, and a cube dimension 3. We can also break a line segment into N self-similar pieces, each with magnification factor N. A square may be broken into **N^2** self-similar copies of itself, each of which must be magnified by a factor of **N.**

$$dimension = \frac{\log(number\ of\ self\ similar\ pieces)}{\log(magnification\ factor)}$$

$$= \frac{\log N^2}{\log N}$$

$$= \frac{2 \log N}{\log N} = 2$$

In Koch curve that is a fractal, after each iteration, all original line segments are replaced with 4, each a self-similar copies. Self-similar copies are 1/3 the length of the origin. N= self-similar copies and S= scale factor.

Hausdorff fractal dimension=D = $\frac{\log N}{\log S}$

$$= \frac{\log 4}{\log 3} \approx 1.26$$

Thus, we take as the definition of the fractal dimension of a self-similar object.

$$Fractal\ dimension = \frac{\log(number\ of\ self\ similar\ pieces)}{\log(magnification\ factor)}$$

2.9.2.3 Two-Threshold Binary Decomposition (TTBD)

The decomposition of the input image is achieved by the Two-Threshold Binary Decomposition (TTBD) algorithm. The Two-Threshold Binary Decomposition (TTBD) takes as input a grayscale image and returns a set of binary images. The first step of TTBD consists in computing a set T of threshold values. This is achieved by employing the multi-level Otsu algorithm [18]. The multi-level Otsu algorithm consists in finding the threshold that minimizes the input image intra-class variance. Then, recursively, the Otsu algorithm is applied to each image region until the desired number of thresholds is obtained. The next step of the TTBD algorithm is decomposing the input grayscale image into a set of binary images. An important property of the TTBD is that the set the binary images obtained is a superset of all binary images that would be obtained. By

using pairs of threshold values it is possible to extract region information from the medium brightness chips that would not be segmented by a single threshold value.

2.9.2.4 Box Counting Algorithm

Box counting is a method of gathering data for analyzing complex patterns by breaking a dataset, object, image, etc. into smaller and smaller pieces, typically "box"-shaped, and analyzing the pieces at each smaller scale. Fractal dimension can be efficiently computed in linear time by the box counting algorithm proposed. If we consider an object represented by a binary image, an approximation Haussdorf fractal dimension D can be obtained through the box counting algorithm. Without loss of generality, the algorithm for the 2D case can be described as follows. Initially, the image is divided into a grid composed of squares of size e x e. The next step consists in counting the number N(e) of squares of size e x e that contains at least one pixel of the object. Then, log(N(e)) x log(1/e) points are computed and use the line fitting method (least squares method) to fit a line to the points. Finally, this curve is approximated by a straight line using a (e.g. least squares fitting). The fractal dimension D corresponds to the slope of this line.

2.10 SVM Classification

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. SVM (Support Vector Machine) is a statistical learning theory which is built on the basis of a limited number of samples in the information contained in the existing training set to get the best classification result. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value (i.e. the class labels) and several. The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. The challenge is to train the machine to understand structure from data and mapping with the right class label, for the best result, the hyper plane has the largest distance to the nearest training data points

of any class. The decision of hyper plane depends upon the feature vector known as the support vector.

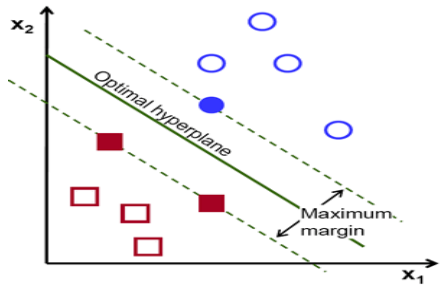


Figure 2.17: SVM Classification

Classifying data has been one of the major parts in machine learning. The idea of support vector machine is to create a hyper plane in between data sets to indicate which class it belongs to. SVM classifier finds the separating hyperplane that has the maximum distance to the closest point of the training set. These closest points are called support vectors. In Figure 2.14, H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximum margin.

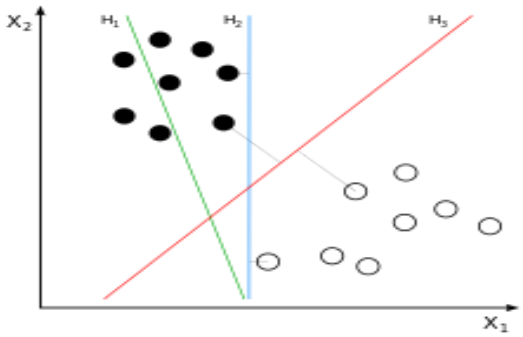


Figure 2.18: Separation of Two Classes

2.11 General Framework

Face detection in color images has gained much attention in recent years. Color is known to be a useful manner to extract skin regions, and it is only available in color images. This allows easy face localization of facial regions without any consideration of its geometrical properties. Among various methods skin color based face detection is simplest and results a better performance. Skin color based face Detection has a general method which includes some steps as in figure 2.15.

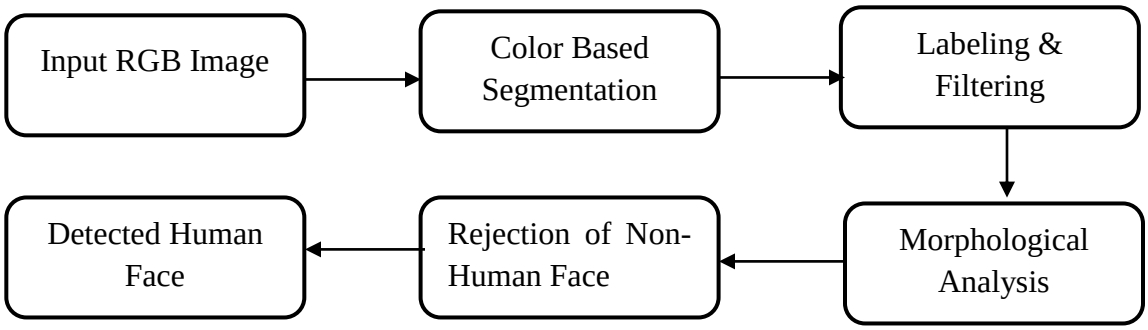


Figure 2.19: Block Diagram of Basic Face Detection

2.12 Related Work

Skin color based method is proposed by H.A. Alabbasi, F. Moldoveanu [1]. Here light compensation preprocessing (Gray World Theory) is proposed to alleviate the interferences from bad illuminating conditions. RGB image is converted to HSV and YCbCr image for segmentation, the segmented image is combined to the output of canny edge detector for better performance. The main drawback of this method is higher computational cost and seems complex because of using various color space in segmentation. Another drawback is if there is no light interference in color images and we apply light compensation preprocessing, it will lead to big change of skin color information and affect the result of skin model. The skin color model used is not so efficient so it can't detect faces of all racial origin.

T. Ahonen, A. Hadid, and M. Pietikainen proposed the method is based on LBP [2]. In this paper, a novel and efficient facial representation is proposed by dividing a facial image into small regions and computing a description of each region using local binary patterns. The LBP operator assigns a label to every pixel of an image by thresholding the 3 x 3-neighborhood of each pixel with the center pixel value. This gives an 8-digit binary number. By computing spatially enhanced histogram features are extracted. Then SVM classifier is used for classification. The main drawback of this method is: small local region reduces recognition rate and using larger local regions increases the robustness to errors. Also they produce rather long histograms, which slow down the

recognition speed especially on large-scale face database. Another limitation is classifier used to detect only gray scale and frontal image.

A skin color based face detection method is proposed by P.T. Pham-Ngoc [3]. Here, RGB color model is used for skin segmentation. The proposed skin color model is better than Lin and Peer model. Face candidates are found by connected component labeling operation and height to width ratio. For classification of faces modified Local Binary Pattern (mLBP) histogram matching and embedded Hidden Markov Model (eHMM) is used. The method gives better performance for face orientation, occlusion and illumination changes. But because of using the combined method as classifier computational cost becomes high.

P. Viola, M.J. Jones [4] proposed Integral Image. Two-rectangle Haar feature is used which is used for feature extraction. For classification the AdaBoost learning algorithm is used to select a small number of critical visual features from a very large set of potential features. Combination of the classifier in a cascade is used so that background regions of the image can be quickly discarded. The main drawback of this method is that the detector is most effective only on gray scale and frontal images. It also sensitive to lighting conditions and can hardly cope with 45° face rotation both around the vertical and horizontal axis.

A skin color modeling based face detection is proposed by K. G Prashanth and M. Shashidhara [5]. The first step is to detect the more skinned region by the fusion of RGB, YCbCr skin color region and CIEL*a*b skin labeled image using Hill Climbing segmentation with K-Means clustering. This step helps to simplify and improves the detection of human faces, since non-skin regions are eliminated. The second step is to locate the faces in a fused image by region property measures. But computational cost is higher because of using three color spaces. Detection rate is slow for all racing origin skin. L. Zou and S. Kamata [6] proposed a method that is based on skin color segmentation and region property. But use of region property is not enough that slow down the detection rate.

Y. Sujuki and T. Shibata [7] developed edge based face detection algorithm that is robust against illumination, focus and scale variations. This algorithm is based on feature vector. Feature vector is based on projected principal-edge distribution (PPED) and Cell distribution algorithm.

Another technique was presented by S. Asadi, S. Rao, V.Saikrishna [8], makes use of Eigen faces and PCA (Principal Component Analysis). PCA calculates the Eigenvectors of the covariance matrix, and projects the original data onto a lower dimensional feature space, which is defined by Eigenvectors with large Eigen values. So, the computational cost is very high. In images, even more than in human faces, the dimensionality of the original data is vast compared to the size of the dataset.

Quan Wang [9] proposed Principle Component Analysis along with Kernel Principle Component Analysis was implemented. In this method, the estimated results of face detection step are patches containing each face in the input image. Directly using these patches for face recognition have some disadvantages, first, each patch usually contains over 1000 pixels, which are too large to build a robust recognition system. Second, face patches may be taken from different camera alignments, with different face expressions, illuminations, and may suffer from occlusion and clutter.

Jebara et al. [10] also presented a mixture of Gaussians. Fleck *et al.* [11], Kjeldsen, and Kender [12] have demonstrated that skin color pixels can be grouped to give locations of face hypothesis. Dai and Nakano [13] have shown successful detection using a combination of color and texture information. Chen *et al.* [14] have demonstrated successful results using a perceptual-uniform color space and a fuzzy logic classifier. A. F Costa, G. Humpire-Mamani, A. J. M. Traina [15] uses SFTA feature.

2.13 Conclusion

In this chapter, I have presented a brief description about the theoretical concept of the proposed approach. In the next chapter, methodology of my proposed method is given.

Chapter 3

Methodology of the Proposed System

3.1 Introduction

The goal of this project is to detect face regions from an input image and categorize the images as face and non-face images. Here I have proposed a robust face detection method based on skin color modeling and later classified by implementing SVM classification.

3.2 Proposed Method

The proposed face detection method is based on three main steps: **1) Image Pre-processing**, **2) Feature Extraction** and **3) Classification.**

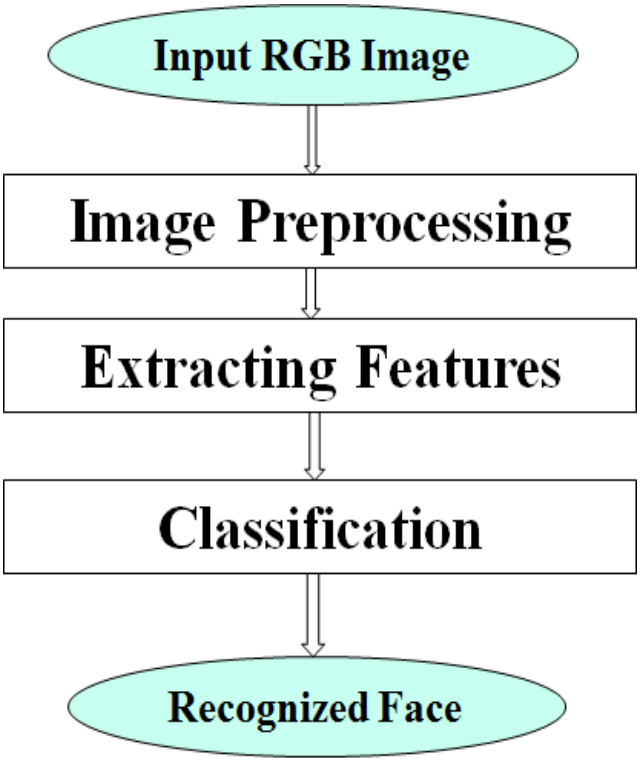


Figure 3.1: Flow chart of the proposed method

3.3 Image Pre-processing

In the proposed framework, the goal is to detect the presence of faces in an image. The pre-processing step is based on image skin regions therefore a skin-based segmentation of input color image is provided first. Then, it is decided for each skin region if it represents a human face or not, using a set of candidate criteria, implementing various morphological and filtering operations.

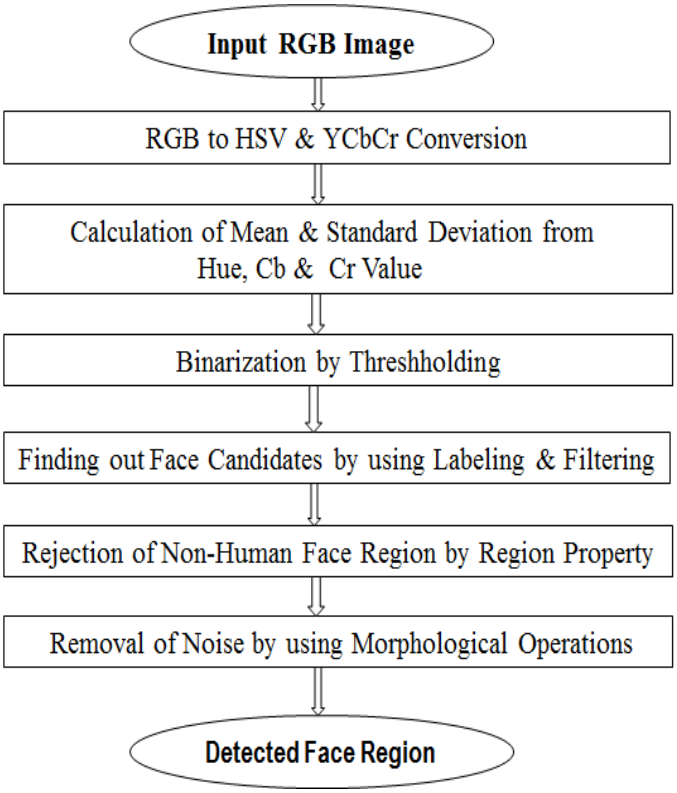


Figure 3.2: Flow chart of the Pre-processing method

3.3.1 Skin Color Segmentation

Human skin has a color that is generally distinct from the colors of other objects from images. Also it is independent of face orientation and scale, geometric variation of the face shape. This property makes motion estimation much easier. There are many challenges in this task such as different illumination conditions, human faces, and similar skin colors The major difference between skin tones is intensity e.g. due to varying lighting conditions and different human face. The color of human skin is

different from the color of most other natural objects in the world. In the proposed method, HSV and YCbCr color model is implemented because of its robustness to lighting changes for the segmentation process. The mean and standard deviation of the hue and saturation is calculated from the desired region of interest and the threshold for the human skin region is calculated from these.

Image Conversion from RGB to HSV:

$$H = \begin{cases} \emptyset \; if \; B \leq G \\ 360 - \emptyset \; if \; B > G \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.1)$$

$$\emptyset = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2 \; + (R-B) \; + (G-B) \;]^{\frac{1}{2}}}\right\} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.2)$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.3)$$

$$V = \max[R,G,B] \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.4)$$

Digital YCbCr (8 bits per sample) is derived from analogue R'G'B' as follows:

$$Y' = 16 + (0.255 \times R_d + 0.5021 \times G_d + 0.0975 \times B_d) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.5)$$

$$P_B = 128 + (-0.148 \times R_d - 0.368 \times G_d + 0.439 \times B_d) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.6)$$

$$P_R = 128 + (0.439 \times R_d - 0.368 \times G_d - 0.071 \times B_d) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.7)$$

After converting RGB to HSV, the above equations are used in the coding part of the proposed algorithm. Cr and Cb are used instead of P_B and P_R. In this project, hue channel of HSV color space and Cb and Cr value of digital YCbCr that is derived from analogue R'G'B' is used. The value of hue describes the color and values of Cb, Cr describes the blue-difference and red-difference chroma components.

In the proposed method, it is assumed that by combining the skin region from the two color space, skin region is extracted. After combining the 2 color spaces values for an image and if one color space fails and gives the false result, even then the skin is extracted by using the combination algorithm. Suppose the results from YCbCr

color space is ‘’A’’ and the result from HSV color space is ‘’B’’. If any of the result contains a skin image then the union of the two will surely be a skin image. This can be understood by the following example: let, A={a, b, c} and B = {b, d}. If “a” = “skin image” then skin image is present only in A and B is failed to extract the skin color and “d” = ” skin image” is failed to extract by “A”, then the union of the two U = (A U B) ={a, b, c, d} that has all the skin image. Hence this method can detect skin color perfectly. By calculating mean and standard deviation of hue, Cb and Cr, thresholds are found. Those regions satisfy the thresholds are considered as skin region and others are non-skin region. Decision rules of this skin color modeling are as follows:

$$\delta\big(P(x,y)\big) = \begin{cases} \mathbf{1}; if\ a\ set\ of\ condition\ is\ satisfied \\ \mathbf{0};\ otherwise \end{cases} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.8)$$

where, $P(x, y)$ is a pixel of color image.

Main reason for using hue, Cb and Cr value:

- Hue is used for its robustness to lighting changes for the segmentation process.
- The color difference components Cb and Cr have the distinguish cluster values for skin color and non-skin color. Influence of luminosity can be removed during the conversion process. This makes the segmentation less dependent on the lighting condition. Furthermore, it has been shown that skin color model based on the Cb and Cr values can provide good coverage of different human races.

3.3.2 Face Candidate Localization

Face candidate localizer is used to find out the most probable face candidate. Connected component algorithm is used to label connected skin region. Filtering and labeling, are used for this purpose.

3.3.2.1 Filtering

Among the connected region, which area is smaller than the thresholds is rejected by the proposed method. This step is called Reducing Small Region by Filtering. The step can reduce the unreasonable region to increase the processing speed.

3.3.2.2 Labeling

Skin segmentation is affected by different lighting conditions, some real face region may be lost. For recovery of these necessary regions labeling operation is used. This process ignores non-skin regions connecting directly to boundaries of their skin regions.

3.3.3 Rejection of non-face region

Each face candidate is tested to know whether region is human face or not, by using geometrical properties of human face. In the research work, a binary image of face candidate passes through these steps and removes non-human face like regions by using the properties. The flowchart of the region properties those are used shown in Figure 3.3.

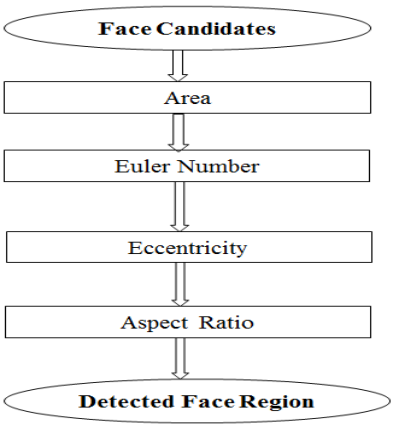


Figure 3.3: Flow chart for Rejection of Non-face Regions

3.3.3.1 Filtering by the condition of Area Pixels

Area of any skin region is calculated by counting number of skin region pixels. Average area of human skin like regions of binary image is calculated to compare this average area with each skin regions of the binary image. If any skin region is less than the average value, then that skin region will be rejected. This method is helpful for removing small skin regions from a binary image.

3.3.3.2 Filtering by the condition of Euler Number

Human faces contain some holes like eyes, eyebrows, mouth. If any skin region doesn't

contain any holes then, we can safely discard those all skin region. The number of holes in a region is computed using the Euler number of region. Euler No, E=total number of connected component of the image - total number of holes in the object. From Euler Number, number of hole is calculated. Hole=1-Euler Number. Skin region that has at least 1 hole is taken and the other regions are discarded.

3.3.3.3 Filtering by the condition of Eccentricity

Generally, shape of human face is likely to be oval shape, so the region which has an oval shape are not rejected by this stage, and those whose shapes are likely to be line are rejected. An ellipse (skin region) whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment. The oval shape of a region can be approximated by an ellipse. Eccentricity of all connected skin regions are calculated and discard all skin regions whose eccentricity is greater than 0.89905. The proposed approach rejects these skin regions which are not satisfied by equation (3.9).

$$\delta(R_e) = \begin{cases} \mathbf{1}; if(R_e < 0.89905 \\ \mathbf{0}; Otherwise \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots. (3.9)$$

where, R_e is the eccentricity of the particular region.

3.3.3.4 Filtering by the condition of Aspect Ratio

Proposed approach rejects non-human face skin region based on aspect ratio or height to width ratio. Generally, height to width ratio of skin regions is important factor for rejecting non-human face regions. The length of major axis and minor axis for each region is measured and aspect ratio is calculated. Aspect ratio=length of major axis / length of minor axis. The region whose aspect ratio is less than threshold is taken and others are discarded. Here, the threshold value for height to width ratio is decided as 2.2. The proposed approach rejects these skin regions which are not satisfied by equation (3.10).

$$\delta(H, W) = \begin{cases} \mathbf{1}; if\ 2.2 < R_a < 1 \\ \mathbf{0}; Otherwise \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots. (3.10)$$

where, R_a is the aspect ratio of connected binary region.

3.3.4 Morphological Analysis

After filtering using region property, morphological operation is used. One of the uses of dilation is to fill in small background color holes in images. One of the problems with doing this, however, is that the dilation will also distort all regions of pixels indiscriminately. By performing an erosion on the image after the dilation, *i.e.* a closing, we reduce some of this effect. Here closing removes small holes in the foreground, changes small islands of background into foreground. Hole and gaps within each region can also be closed by a flood fill operation, if there remains hole in the region.

3.4 Feature Extraction

In the proposed method texture features are used for feature extraction. Texture features are extracted by using **SFTA (Segmentation-based Fractal Texture Analysis)** method.

Reason for choosing SFTA method:

- SFTA is faster than Filter bank-based methods (Gabor feature) and Haralick descriptors (GLCM) with respect to feature extraction time.
- SFTA has shown superior performance with respect to image classification accuracy.
- GLCMs have to consider multiple orientations and scales that require a big volume of images and the computational cost can be critical. SFTA deal deals with computational cost.
- Fast Fractal Stack (FFS) feature extraction method does not perform well for all texture datasets

3.4.1 Decomposition Method

The Two-Threshold Binary Decomposition (TTBD) takes as input a grayscale image $I(x, y)$ and returns a set of binary images $I_B(x, y)$. In the decomposition method multilevel Otsu method is used. Multilevel Otsu Method image pixel is classified into

several classes and gives threshold for each of the classes. In the project, number of thresholds is 4.

In the first step, a set T of 4 thresholds are computed using multilevel Otsu method. By using 4 thresholds, 4 binary images are obtained from the grayscale image. In the next step, the pairs of thresholds are selected applying a two-threshold segmentation as follows in equation (3.11)

$$I_B\ (x,y) = \begin{cases} 1\ if\ t_l < I(x,y) \leq\ t_u \\ 0, otherwise \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (3.11)$$

Where, t_l and t_u denote, respectively, lower and upper threshold values. There are 4 pairs of thresholds by considering last threshold 1. Another 4 binary images are obtained from the pairs of thresholds. So, for 4 thresholds, there are 8 binary images.
The reason for using pairs of thresholds to compute the set of binary images is to segment objects that otherwise would not be segmented by regular threshold segmentation. By using pairs of threshold values it is possible to extract region information from the medium brightness chips that would not be segmented by a single threshold value. The flowchart of decomposition method is given in Figure 3.5.

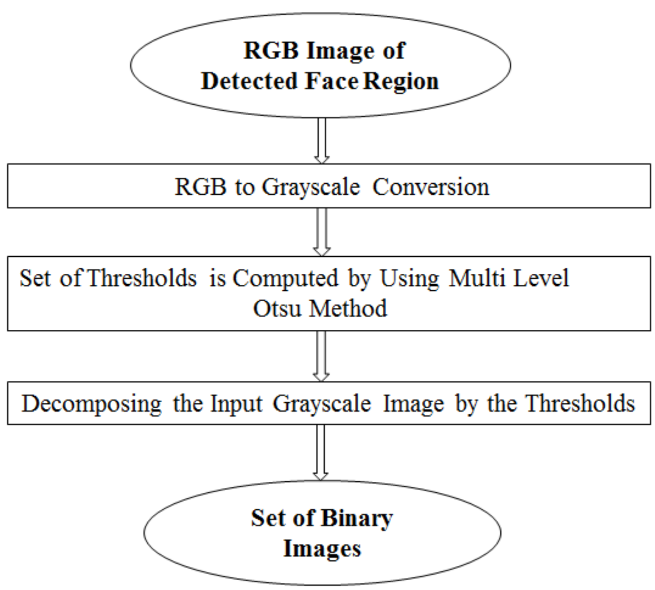


Figure 3.4: Flow Chart for Decomposition Method

3.4.2 Extraction Method

After applying the Two Threshold Binary Decomposition to the input gray level image, the SFTA feature vector is constructed as the resulting binary images' size, mean gray level and boundaries fractal dimension.

For the 8 binary images, mean gray level and area (number of pixel count) is calculated. These are the 2 features.

Then from the resulting 8 binary image, contour of the image is detected. It is called border image $\Delta(x, y)$. Border image takes 1 if the pixel at position (x, y) in the corresponding binary image has the value 1 and having at least one neighboring pixel with value 0. Otherwise the border image takes the value 0. The equation (3.12) represents the border image.

$$\Delta(x,y) = \begin{cases} 1 \; if \; \exists (x',y') \in N_8 \; [(x,y)] : \; I_b(x',y') = 0 \cap I_b(x,y) = 1 \\ 0, otherwise \end{cases} \quad \ldots\ldots\ldots\ldots (3.12)$$

For each 8 border images, fractal dimension is calculated. This is called Hausdorff dimension and calculated by using box counting algorithm. For 8 binary images, 3 features are calculated. So, there are 24 features in total. The flow chart of extraction method is represented in Figure 3.6.

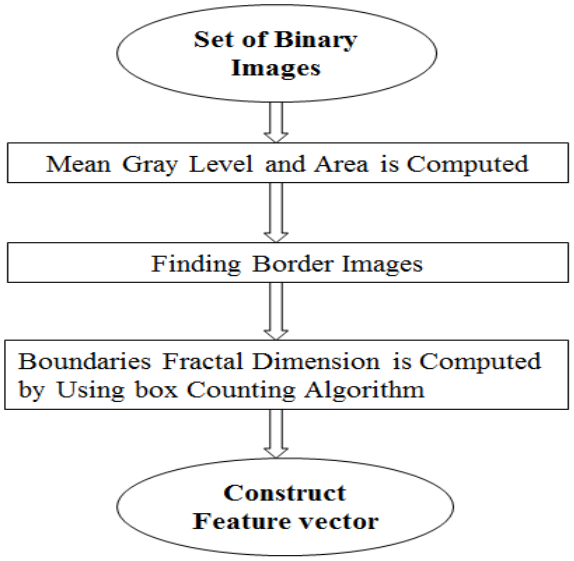


Figure 3.5: Flow Chart for Extraction Method

3.5 Classification using Support Vector Machine

SVM classifier is used for training and classification purpose. An overview of the classification is given below:

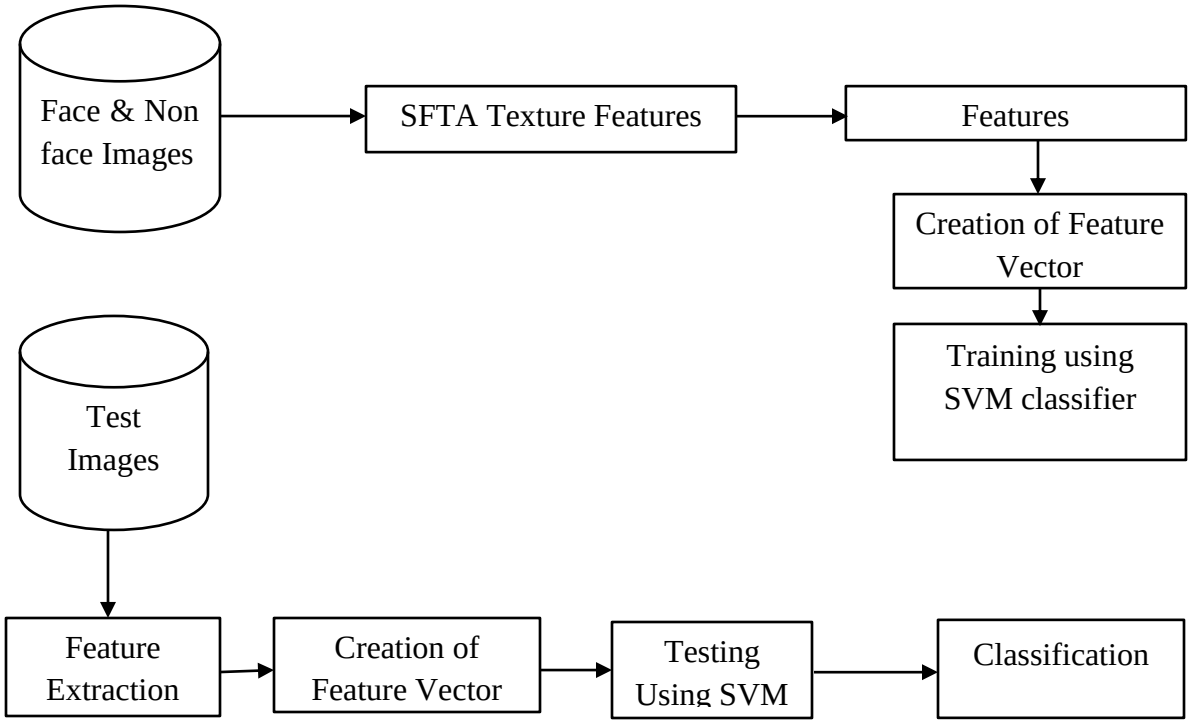


Figure 3.6: Framework of the SVM Classification Method

The feature vector created for all the database images are required to be trained for learning a SVM classifier using supervised learning. The support vector machine is explicitly told to find the best separating line. The support vector machine searches for the closest points, which it calls the "support vectors".

Once it has found the closest points, the SVM draws a line connecting them. It draws this connecting line by doing vector subtraction. The support vector machine then declares the best separating line to be the line that bisects and is perpendicular to the connecting line. The line or the hyper plane is given by the following equation:

$$Wx + b = 0$$

Where, W= Normal vector to the line/ orientation of the D- dimensional space

$x =$ Sample feature vector.

$b =$ Bias value/ Position of that line in the D- dimensional space

If the feature vector is one or two dimensional, the line is linear. When its dimension is more than two the line becomes a plane. It is then called as the hyper plane.

The purpose of training process is to widen the distance boundary from the line. The maximization of the margin is given by:

$$M = 2/|w|$$

If the sample lies on the line, the value of $wx + b$ becomes 0. If the sample feature vector lies above the line, then $wx + b$ becomes greater than 1 and if the sample feature vector lies below the separating line then the value of wx+b becomes less than 1. Support Vectors are those that is close to the boundary line. The goal of SVM is to satisfy the following equations:

$$wx_i + b \geq 1 \; if \; y_i = +1$$

$$wx_i + b \leq 1 \; if \; y_i = -1$$

$$y_i(wx_i + b) \geq 1 \; for \; all \; i$$

Here, there are two classes to be recognized. They are faces and non-faces. Face images are defined as 1 and non-face images are defined as 2 instead of +1 and -1. Feature of the face and non-face image database is extracted by using SFTA method. Feature vectors of face and non-face images are trained by using SVM. Feature of the query image or query image database is extracted. By comparing the feature vector of test image and the feature vector face and non-face database image, classification is performed by SVM. At last, the query image or query image database are assigned a particular label of either 1 or 2. 1 is for face image and 2 is for non-face image.

3.6 Conclusion

Here, I have presented the whole methodology of the proposed framework. The next chapter describes the implementation procedure of the whole method with the help of the sample images.

Chapter 4

Experiments

4.1 Introduction

In this chapter, a brief description about the data sets and experimental tools that I have used during my project is included. The evaluation measurements from which analysis of my proposed method is done also included. Then I will describe the overall implementation process of the face detection and its Classification results. Finally experimental results and analysis is included.

4.2 Data Collection

In this project, the data set for detecting face region are selected by me randomly. The image database quality of a well-known server which can be downloaded from internet is very easy to implement. In the project, I ignore these types of images. I collected images from my friends, relatives and juniors. There are 80 images in my data set which are captured in different environment and illuminations. A portion of data set is given in Figure 4.1.

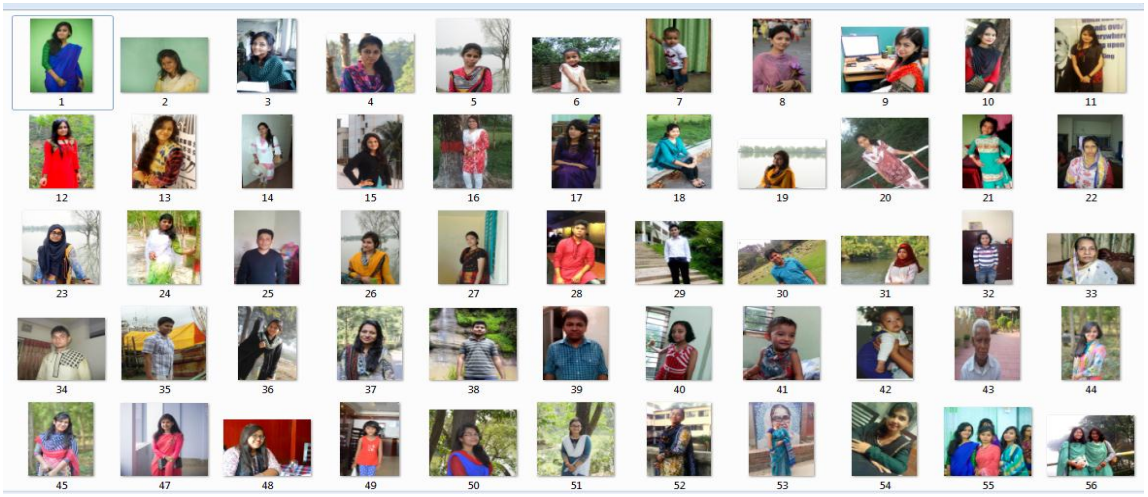


Figure 4.1: A Portion of Data Set

4.3 Environment and Tool

Environment: The experiment is done under the following configuration of the machine
Processor: 2.00 GHZ

RAM: 1 GB

Digital Camera: 16.1 Megapixel

Operating system: windows 7

Tool:

- MATLAB 2015

4.4 MATLAB Overview

The name MATLAB stands fo*r matrix laborator*y. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows solving many technical computing problems, especially those with matrix and vector formulations. The reason we have decided to use MATLAB for the development of this project is its toolboxes. Toolboxes allow learning and applying specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular

classes of problems. It includes among others image processing and neural networks toolboxes.

4.5 Evaluation Measurements

The accuracy rate of my proposed method is measured from the evaluation measurements. The performance of the system can be measured in terms of its precision rate. Precision is the ratio between the numbers of relevant images detected that is the number of face detected from the given face images (hits) and total number of images detected that is the total number of faces in the images. For Classification, precision rate is the ratio between the numbers of faces classified (hits) and total number of face to be classified. Precision rate is defined as follows:

$$Precision\ Rate = \frac{Total\ number\ of\ relevant\ images\ Detected}{Total\ number\ of\ Images\ Detected} \times 100\% \ldots\ldots (4.1)$$

4.6 Implementation of the Proposed Face Detection Method

In this work, the proposed algorithm has three sections, such as: Image Pre-processing, Feature Extraction and SVM Classification.

4.6.1 Implementation of Image Pre-processing

Image pre-processing consists of Skin Color Segmentation, Face Candidate Localization and Rejection of non-face skin regions. Each section is described consecutively.

4.6.1.1 Implementation of Skin Color Segmentation

In this project, HSV and YCbCr color model is used for segmenting the RGB image. Hue value is found by converting HSV color model and Cb, Cr values are found from the equation (3.6) and (3.7).

4.6.1.1.1 Input RGB Image

In the first step, an input of RGB image is obtained that may contain a human face. Input RGB image is shown in Figure 4.1.

Figure 4.1: Input RGB Image

4.6.1.1.2 Conversion to HSV and YCbCr color space

The RGB image is then converted to HSV color space by using `rgb2hsv()` for extracting out the hue channel. From gamma corrected RGB, value of Cb, Cr are found. Figure 4.2 and 4.3 shows an image of HSV and YCbCr color space.

Figure 4.2: HSV Image

Figure 4.3: YCbCr Image

4.5.1.1.3 Binarization by Threshholding

The color segmentation performed in the previous step is used to find out the proper threshold value for human skin pixel which used to produce respective binary image. The threshold value is found by calculating mean and standard deviation of hue, Cb and Cr. Lower value of threshold is (mean - standard deviation) and higher value is (mean + standard deviation). Thresholds for hue, Cb and Cr are:

$$0.01 < Hue < 0.1$$

$$140 < Cb < 195$$

$$140 < Cr < 165$$

The binary image of the input image shown in fig: 4.4.

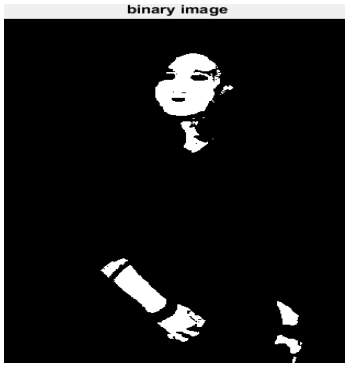


Figure 4.4: Binary Image

4.6.1.2 Implementation of Face Candidate Localization

Face candidate localizer is used to find out most probable face candidates. In this approach, there are steps to find out face candidates. Such as: labeling and filtering.

4.61.2.1 Labeling Operation

Here labeling operation is used to detect the connected skin region. Labeled image is shown in Figure 4.5. In this figure, there are 16 connected regions.

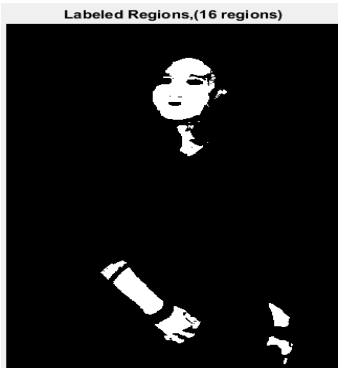


Figure 4.5: Labeled Image

4.6.1.3 Implementation of Rejection of Non-Face Region

Now, it is necessary to discard the portion of images that are not face regions. This is filtering by the region properties of human face. Each face candidates may be face region if it satisfies some region properties of human face.

4.6.1.3.1 Implementation of Filtering Process

To find out the human face region, several properties have been used such as: area, Euler number, Eccentricity and Aspect ratio. Using `regionprops` the properties are calculated. The extracted face regions are shown in the Figure 4.6, 4.7, 4.8 and 4.9.

- **Implementation of the Area Property:** Average area of human skin like regions of binary image is calculated. If area of skin regions is greater than the average value, then that skin region is considered as skin and others are non-skin. In Figure 4.6, filtering using Area is shown. There are 4 regions in the figure that is less than the previous labeled image.

Figure 4.6: Filtering by Using Area Property

- **Implementation of the Euler Number Property:** The number of holes in a region is computed using the Euler number of region. Human faces contain holes like eyes, eyebrows, mouth etc. So, the skin region that has at least 1 hole is taken and the other regions are discarded. Figure 4.7 represents filtering by using Euler Number. There are 3 regions in the figure.

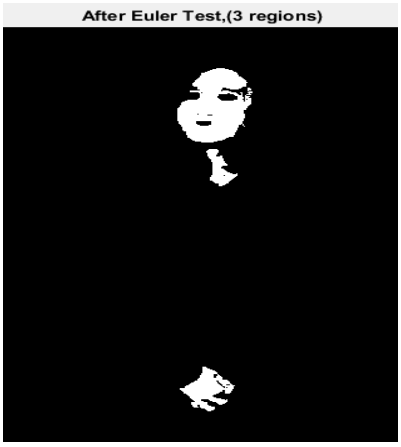


Figure 4.7: Filtering by Using Euler Number Property

- **Implementation of the Eccentricity Property:** Shape of human face is likely to oval shape that can be approximated by an ellipse. An ellipse (skin region) whose eccentricity is 0 is actually a circle, while an ellipse whose eccentricity is 1 is a line segment. The region that has eccentricity less than 0.89905 is rejected and others are accepted. Figure 4.8 represents filtering using eccentricity. There are 2 regions.

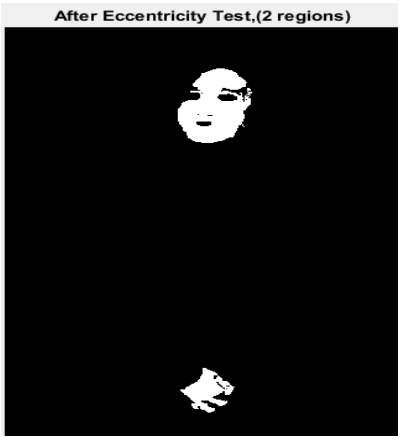


Figure 4.8: Filtering by Using Eccentricity Property

- **Implementation of the Aspect Ratio Property:** Using `regionprops()` function length of major axis and minor axis for each region is found and aspect ratio is calculated. The region whose aspect ratio greater than 2.2 and less than 1

is taken and others are discarded. In Figure 4.9, filtering using Aspect ratio is represented. The reduced number of region in the figure is 1. This is the final step of filtering in my proposed method.

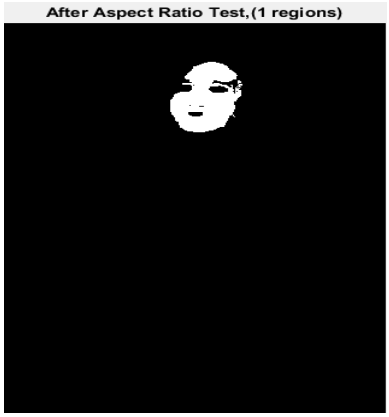


Figure 4.9: Filtering by using Aspect Ratio Property

4.6.1.4 Implementation of Morphological operation

There are even many noises in the region extracted by the filtering process. In this step, morphological operation is used to refine the region extracted from the filtering step. The holes are filled and extra edges are removed by closing operation and finally face region is obtained. For closing operation `imclose()` is use. Figure 4.10 represents the binary image using morphological operation.

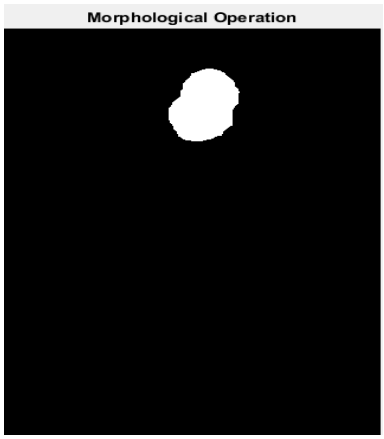


Figure 4.10: Noise Reduction by Morphological Analysis

4.6.1.5 Merging Face Region with the Input Image

The face region find in the previous step is merging with the input image by performing logical AND operation. Figure 4.11 represents the superimposed image of face region.

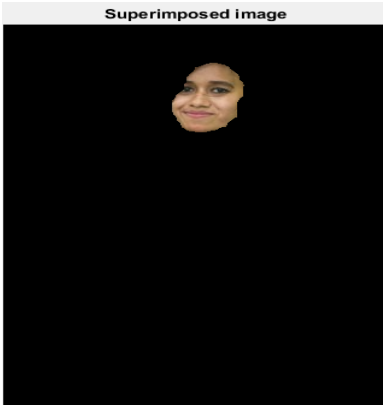


Figure 4.11: Superimposed Image of Face Region

4.6.1.6 Original Image with Rectangular Box

Finally, original image is represented by a rectangular box around the face region. This is done by using Bounding Box region property that gives smallest bounding box around the region of interest. The final result of detected face region is given in Figure 4.12.

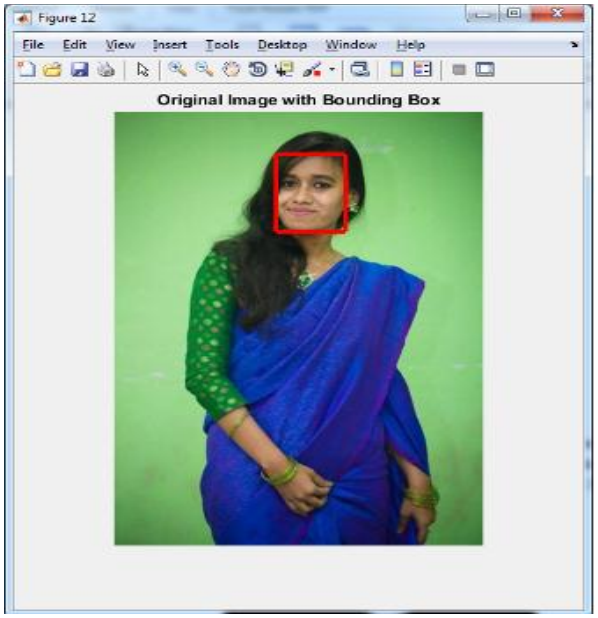


Figure 4.12: Detected Face Region

4.7 Implementation of Feature Extraction

The features extracted in this project are mean gray level, size and fractal dimension of the region. Features are extracted from the face image and also from non-face image. The implementation of these feature extraction is given below.

4.7.1 Implementation of Decomposition Method:

As, I used 4 thresholds in the project, there are 8 binary images by using these thresholds. For using 4 single thresholds 4 binary images are obtained. The other 4 binary images are obtained by using pairs of thresholds. The resulting 8 binary images from grayscale image by using two thresholds binary decomposition method are represented in Figure 4.13.

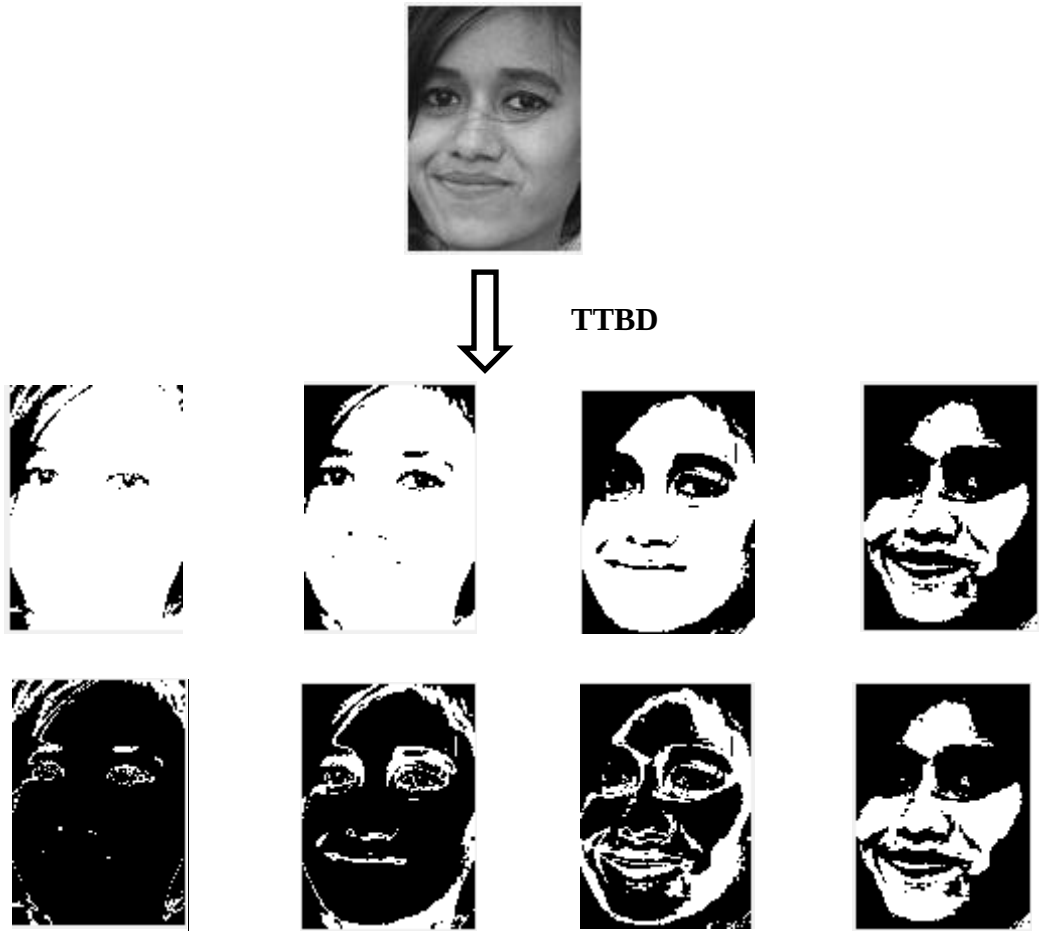


Figure 4.13: Decomposition of Face Region by Using TTBD method

4.7.2 Implementation of Extraction Method:

From the 8 binary images, mean gray level and area is measured. But for computing fractal dimension contour of the binary images is necessary. In the next step contour of the each binary image is detected from which Hausdorff fractal dimension is calculated that measures of the local size of a set of images**.** Figure 4.14 represents the contour of the each binary image.

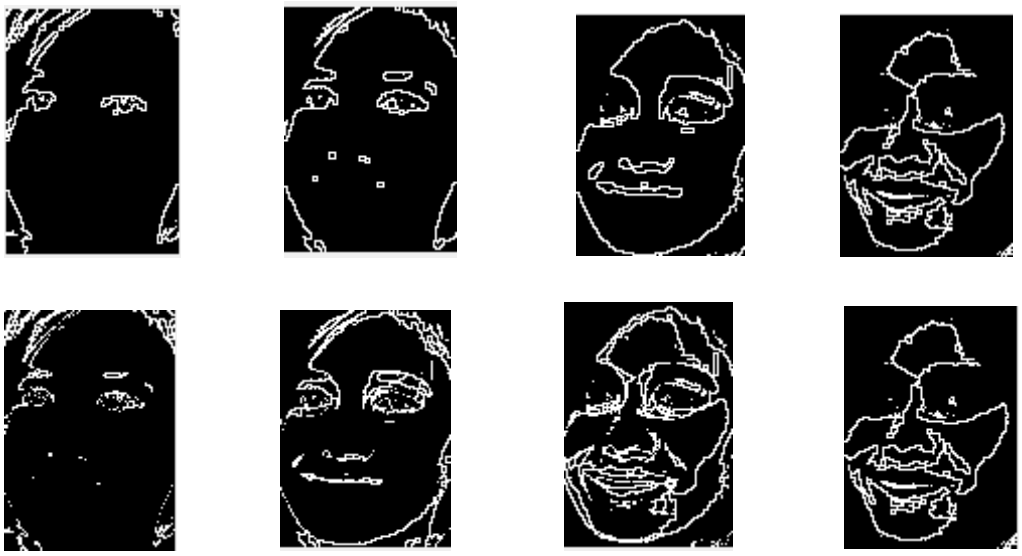


Figure 4.14: Contour Detection for Each Binary Image

4.8 Implementation of Classification

In this part, the various parts used for SVM classification are given with a view to categorizing the images into either faces or non-faces.

4.8.1 Creation of Feature Vector

After calculating the SFTA texture features, they are inserted into a vector called feature vector. The features of face and non-face images of the database are calculated. The feature vector is first initialized to zero. After extracting the features, the columns of the feature vector are updated. There are three features for resulting 8 binary images. So, there are 24 features in total. A portion of the face image and non-face image database and their feature vectors are given below:

Figure 4.15: A Portion of Face Image Database

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_10	D_11	D_12	D_13	D_14	D_15	D_16	D_17	D_18	D_19	D_20	D_21
1.042452	59.81466	232	1.239404	73.97796	499	1.353552	95.28933	750	1.201195	125.3935	338	1.194644	50.4533	364	1.347001	71.77315	745	1.444041	100.6072	1
1.361201	59.39329	834	1.411708	84.86489	977	1.232335	143.6322	484	1.162754	186.6223	368	1.481773	60.66032	1366	1.483762	90.90113	1325	1.27472	140.1826	
1.271089	51.20574	209	1.324456	72.21631	282	1.406302	102.5346	376	1.399463	138.0436	367	1.194002	36.98817	169	1.369051	65.919	321	1.558334	111.3323	
1.290139	60.16447	608	1.369518	73.41315	852	1.422704	109.1985	1043	1.344306	145.5613	734	1.331435	50.04539	727	1.492964	74.81785	1367	1.516733	112.8787	1
1.134518	60.52759	290	1.25974	91.4159	541	1.356194	143.9749	796	1.289984	196.8331	605	1.239423	61.57364	516	1.458915	109.2617	1215	1.493623	159.2074	1
1.352977	83.03657	793	1.417534	109.9147	1032	1.40496	158.3961	977	1.358381	200.9235	758	1.396513	76.03648	932	1.5278	113.5278	1656	1.479862	161.9925	1
1.299655	65.10047	637	1.361388	89.70732	820	1.402774	131.7063	960	1.267617	184.8896	498	1.375227	59.52983	838	1.501804	95.29223	1468	1.462229	135.9737	1
1.327023	66.09484	717	1.419473	91.8286	1021	1.513325	129.9238	1509	1.292922	195.3902	592	1.350681	54.66667	762	1.559984	92.64372	1816	1.560982	131.9736	1
1.291857	72.40246	569	1.373133	97.35165	819	1.422431	138.7778	1017	0.996837	211.3591	181	1.324053	66.71642	670	1.531166	105.9421	1607	1.460457	144.6084	1
1.416051	49.08506	1011	1.446039	74.22498	1129	1.07771	178.2746	244	1.102789	204.814	242	1.511843	48.56533	1523	1.463772	74.13663	1215	1.009371	160.6331	
1.412387	59.02425	1031	1.450755	91.99495	1187	1.437636	144.3139	1099	1.293231	197.982	612	1.493191	58.91701	1458	1.55698	98.86361	1855	1.507443	151.9655	1
1.317475	67.11492	583	1.334031	90.9053	623	1.448863	117.6754	804	1.40404	146.8457	674	1.301287	54.33203	512	1.38222	83.18519	729	1.514921	116.7184	1
1.37142	65.31803	871	1.379178	90.97409	849	1.456245	122.3897	1183	1.39772	159.6569	924	1.352881	51.53333	765	1.417084	79.75345	941	1.55592	125.0467	1
1.297831	73.11728	648	1.464541	100.6734	1246	1.419946	155.5325	1016	1.344853	201.5887	637	1.439883	68.68146	1127	1.557987	105.185	1849	1.476523	159.6986	1
1.330061	52.86202	761	1.473851	74.37249	1294	1.472133	117.2958	1278	1.259029	164.1384	513	1.426962	46.84623	1060	1.594233	78.93295	2088	1.502871	116.6469	1
1.218278	33.13288	444	1.324736	50.68436	716	1.46473	88.17609	1238	1.29882	132.952	646	1.254832	27.14801	527	1.512937	54.44759	1555	1.513965	89.38069	1
1.155981	37.04054	296	1.267966	62.06394	563	1.359562	106.8647	813	1.268642	146.6095	566	1.290243	42.0774	646	1.468126	76.74687	1276	1.480012	116.7294	1
1.298088	29.45666	646	1.313936	51.63965	691	1.333389	94.74488	733	1.349211	134.9801	753	1.326737	24.67545	721	1.421113	56.64067	1077	1.483069	106.3514	1
1.397159	65.5883	974	1.46955	91.06269	1292	1.515291	122.8961	1530	1.452984	150.0341	1172	1.403752	53.49789	950	1.53629	84.79268	1611	1.570195	120.789	1
0.856307	92.39474	38	1.202288	131.073	178	1.441922	170.773	423	1.328309	202.6173	277	1.074291	94.19444	108	1.399048	132.6948	367	1.516894	173.5441	
1.285809	58.92122	622	1.472516	85.81456	1305	1.382714	155.0915	831	1.253381	211.2748	433	1.439441	55.92356	1112	1.575439	95.65058	1989	1.449263	162.5557	1
1.127614	122.4186	129	1.370077	138.5345	333	1.484735	162.9524	483	1.360838	183.0063	316	1.215821	107.9091	176	1.528802	138.9635	603	1.562749	164.7852	
1.163729	70.46	150	1.221008	97.25926	189	1.376701	133.3274	339	1.423256	174	373	1.094507	49.93636	110	1.342458	92.77273	286	1.561903	142.0922	
1.307082	38.10774	659	1.332555	68.98939	754	1.371939	118.6822	856	1.305026	171.434	636	1.368056	36.51139	878	1.450742	76.91207	1251	1.494455	132.7864	1

Figure 4.16: Creation of Feature Vector for Face Image Database

Figure 4.17: Non-face Image Database

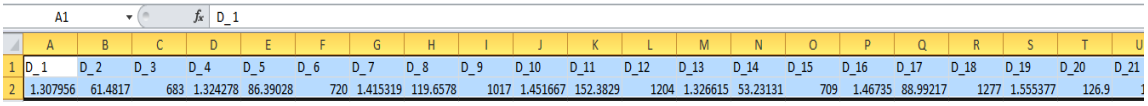
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_10	D_11	D_12	D_13	D_14	D_15	D_16	D_17	D_18	D_19	D_20	D_21
1.556037	111.0135	5494	1.723388	126.9912	12094	1.735624	149.5673	12222	1.585083	170.3564	5483	1.582608	97.68434	4790	1.737757	118.9106	11638	1.716603	143.4429	10
1.45545	86.32291	3806	1.631935	106.8307	8328	1.762752	132.5977	15128	1.628584	156.7056	6983	1.494624	71.87906	3506	1.721757	99.00545	11184	1.758007	126.7878	13
1.517737	111.817	4743	1.732888	126.5092	13385	1.775145	144.9335	16323	1.595169	165.5017	5701	1.558066	94.24543	4323	1.749867	116.4343	13092	1.751396	138.624	13
1.596442	108.9062	7124	1.749451	119.4132	14881	1.779332	135.9984	16567	1.593692	154.8155	5535	1.565514	89.56186	4373	1.759006	109.0483	13673	1.753267	129.7187	13
1.405118	65.47624	1957	1.418971	91.8584	2507	1.493444	143.4602	3924	1.318433	191.3994	1447	1.378	50.54858	1863	1.562238	94.9798	5395	1.511501	140.2412	4
1.563925	97.4395	5959	1.691496	115.6105	10293	1.693712	143.4133	10466	1.416841	186.2656	2786	1.643377	90.68166	7847	1.743056	111.5496	12994	1.701938	141.1547	10
1.369641	49.20907	1521	1.422542	74.34887	1866	1.172581	171.1457	556	1.099041	213.1116	439	1.426924	34.69032	1860	1.420989	64.35809	1804	1.233753	177.2693	
1.679754	60.61138	10092	1.625617	87.47329	7564	1.287031	178.2633	1405	1.17351	230.3347	759	1.718482	55.4663	11958	1.656028	91.17527	8832	1.300569	175.9651	1
1.419612	85.15276	2193	1.388442	116.1157	2549	1.304821	194.562	1105	1.291236	222.8203	1046	1.373935	72.29691	2233	1.416575	111.2305	2859	1.351509	189.8378	1
1.556037	111.0135	5494	1.723388	126.9912	12094	1.735624	149.5673	12222	1.585083	170.3564	5483	1.582608	97.68434	4790	1.737757	118.9106	11638	1.716603	143.4429	10
1.562608	110.6749	5746	1.739162	127.0172	12863	1.767905	148.307	14362	1.610831	169.625	5918	1.585508	95.22041	4723	1.750137	117.1677	12034	1.750346	142.0082	12

Figure 4.18: Creation of Feature Vector for Non-face Image Database

4.8.2 Image Classification

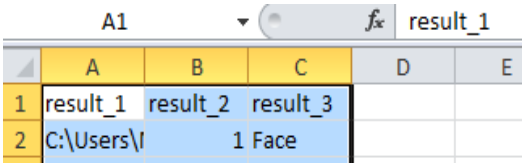
After producing the feature vector of face and non-face images, the features from query image database is calculated for classifying the images either face or not. For classification, training process is required. For training and classification, SVM classifier is used here. After creating training and testing group training, training group is passed to the function `svmtrain()`. The feature vector and some predefined classes are passed as the parameters for learning with features. The testing group and the result of training is passed to `svmclassify()` function for classifying the image in classes. Here, **class 1** represents **face** images and **class 2** represents **non-face** images. The classification steps for query images are given below.

Figure 4.19: Query Image -1



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_10	D_11	D_12	D_13	D_14	D_15	D_16	D_17	D_18	D_19	D_20	D_21
2	1.307956	61.4817	683	1.324278	86.39028	720	1.415319	119.6578	1017	1.451667	152.3829	1204	1.326615	53.23131	709	1.46735	88.99217	1277	1.555377	126.9	1

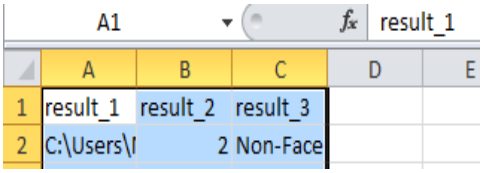
Figure 4.20: Creation of Feature Vector for Ouery Image -1



	A	B	C	D	E
1	result_1	result_2	result_3		
2	C:\Users\I	1	Face		

Figure 4.21: Classified as Face Image

Figure 4.22: Query Image -2



	A	B	C	D	E
1	result_1	result_2	result_3		
2	C:\Users\I	2	Non-Face		

Figure 4.23: Classified as Non-face Image

4.9 Experimental Results for Face Detection

Here, some images are shown along with the important steps of implementation. An input image consists of only single face with an outdoor background is given below

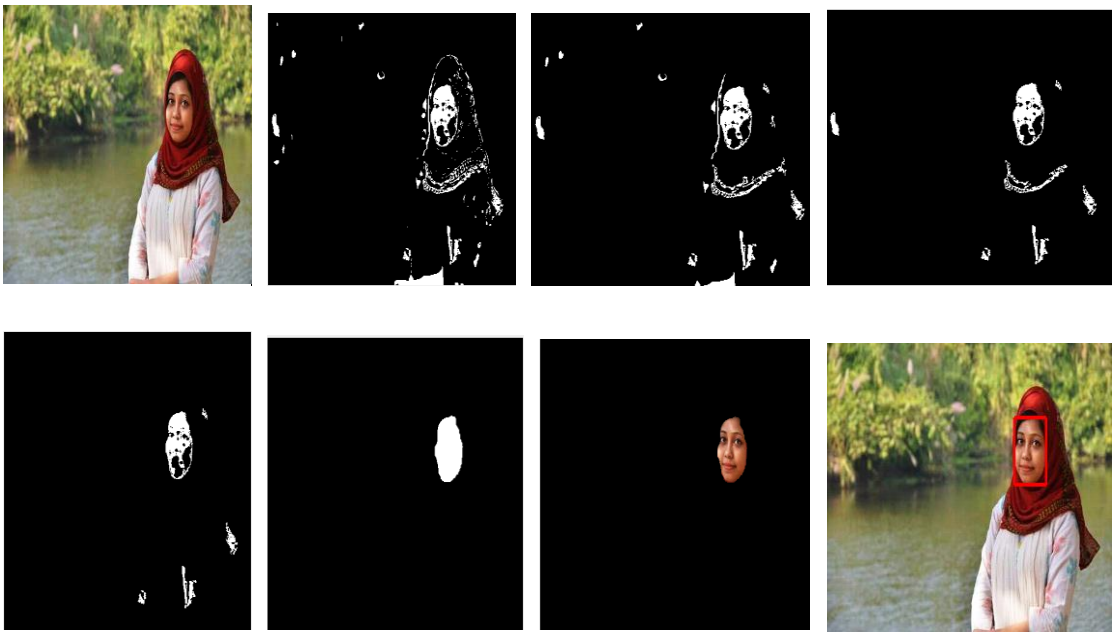


Figure 4.24: Procedural Steps of Implementation (First Sample Image)

Here, the an input image with single face in an indoor background is given below

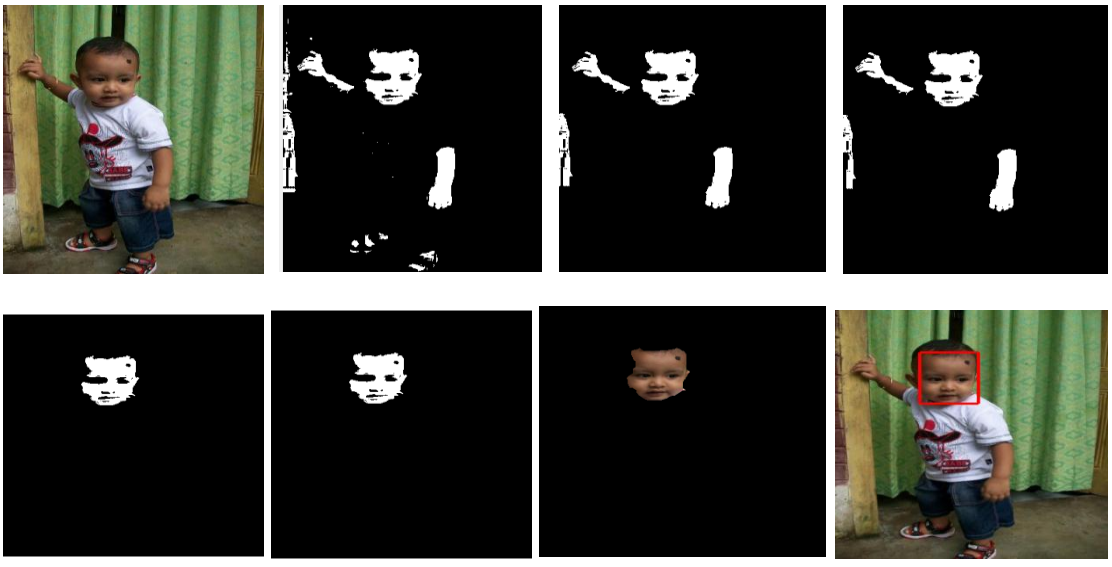


Figure 4.25: Procedural Steps of Implementation (Second Sample Image)

Here, 2 input images consist of multiple faces with indoor and outdoor background.

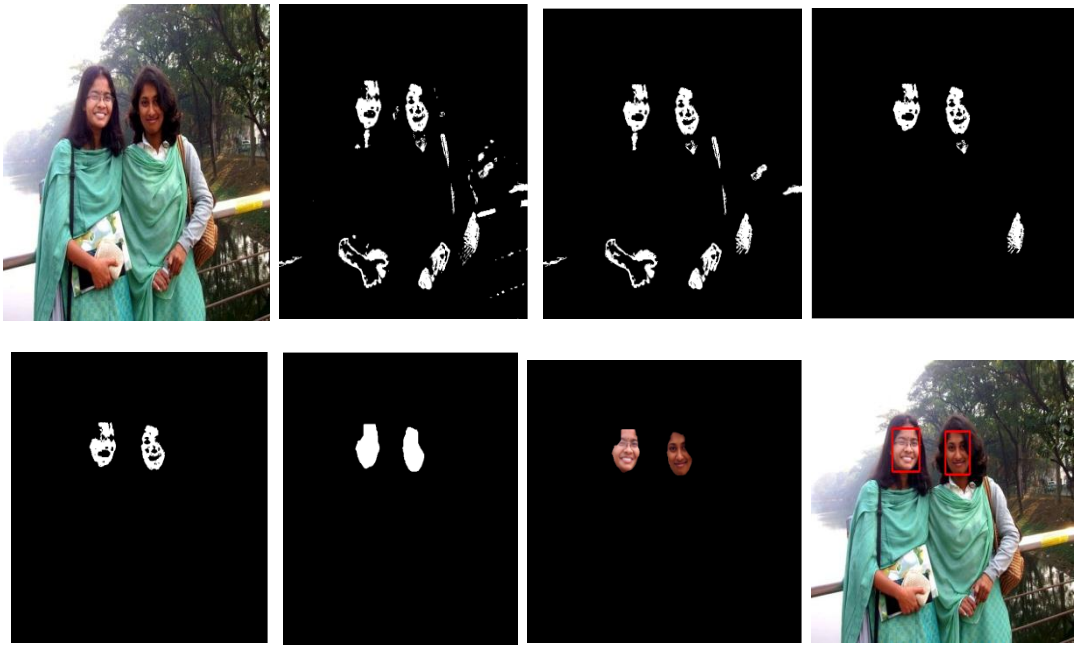


Figure 4.26: Procedural Steps of Implementation (Third Sample Image)

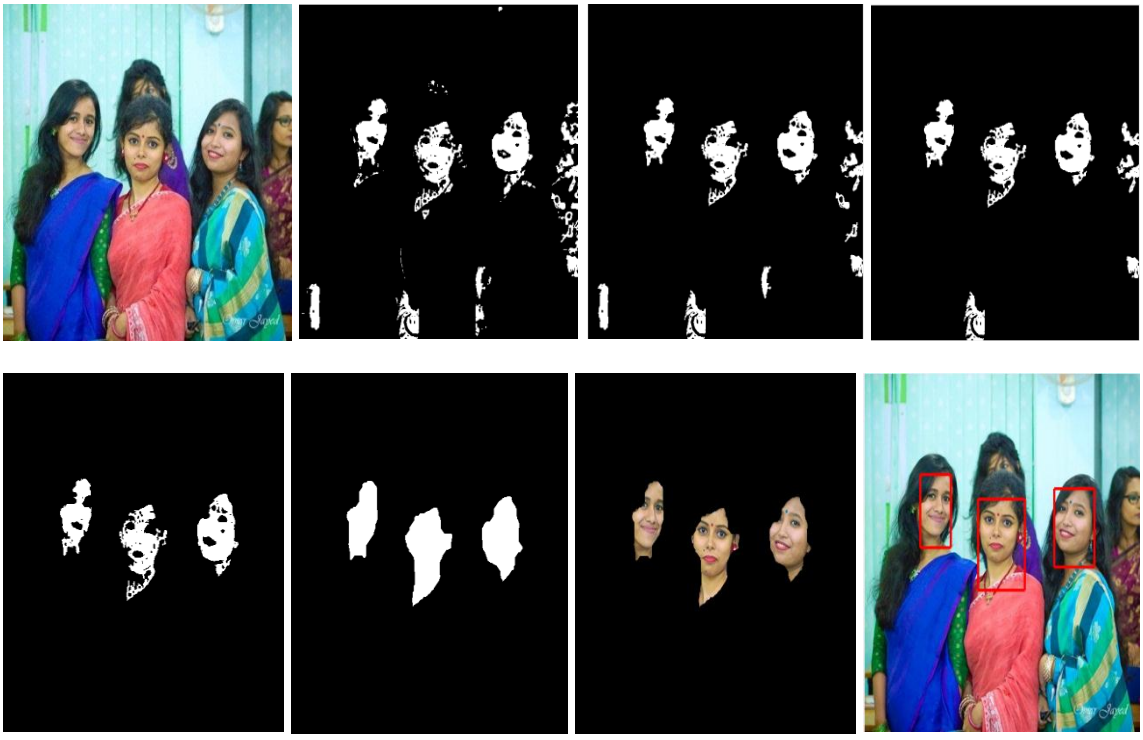


Figure 4.27: Procedural Steps of Implementation (Fourth Sample Image)

4.10 Experimental Result for Classification

In my experiment, I selected two categories of image: face and non-face. I created a database of 174 face images from 80 images. The cropped face regions are passed to the classification step.

Figure 4.28: Query Image-3

A1 result_1

	A	B	C	D	E
1	result_1	result_2	result_3		
2	C:\Users\I	1	Face		

Fig 4.29: Classified as Face Image

Fig 4.30: Query Image-4

A1 result_1

	A	B	C	D	E
1	result_1	result_2	result_3		
2	C:\Users\I	2	Non-Face		

Fig 4.31: Classified as Non-face Image

Fig 4.32: A Portion of Query Image Database

A1 result_1

	A	B	C	D	E
1	result_1	result_2	result_3		
2	C:\Users\I	2	Non-Face		
3	C:\Users\I	2	Non-Face		
4	C:\Users\I	2	Non-Face		
5	C:\Users\I	1	Face		
6	C:\Users\I	1	Face		
7	C:\Users\I	1	Face		
8	C:\Users\I	1	Face		
9	C:\Users\I	1	Face		
10	C:\Users\I	2	Non-Face		
11					

Figure 4.33: Classified Face and Non-face Image of Test Database

4.11 Analysis

4.11.1 Accuracy Rate for proposed face detection method

The proposed system correctly detects 179 face regions out of 186 face region. The method is implemented for 100 images with various illumination conditions, rotations and poses. The precision rate is measured from the equation (4.1).

The Statistical data of the proposed face detection method is given in table 5.1.

Table 5.1: Statistical Data of Face Detection Method

Number of face in an image	Image number	Face number	Hits	Misses	Precision rate
1	53	53	53	0	100%
2	22	44	44	0	100%
3	15	45	45	0	100%
4	6	24	22	2	91.6%
Above 4	4	20	15	5	75%
Total	100	186	179	7	96.23%

The detection rate or precision rate of the method is 96.23% which proves that this method can efficiently detect faces. If the number of face candidates increase, the performance of this system decrease and there will be many misses (faces that are not detected). Also non-face objects similar to face patterns in color and shape may give false detection.

Comparison of my proposed method with [6] is given in the following Table 5.2.

Table 5.2: Comparison of the Face Detection Result

Method	**Precision or Detection Rate**	**False Detection Rate**	**Missed Detection Rate**
Proposed Method	0.96	0.03	0.01
Existing Method [6]	0.95	0.03	0.02

4.11.2 Accuracy Rate for proposed classification method

There are 186 face images for classification that was detected in previous method. The number of non-face image is 20. From 186 face image, 164 are classified accurately. From 20 non-face images, 18 are classified. Precision rate is defined classification method as follows:

The statistical data of the proposed classification method is given in Table 5.3.

Table 5.3: Statistical Data of Classification Method

Face or non-face	**Number of face or non-face**	**Hits**	**Misses**	**Precision Rate**
Face	186	164	22	88.17%
Non-face	20	18	2	90%

The precision rate for classifying face images by using SVM classifier is 88.17% which proves that SVM classifier classify the face and non-face image efficiently. The face image is captured in different skin color and illuminations.

4.12 Conclusion

The primary focus of the work is to develop a system which can efficiently detect faces from an input color image. It is also created with a view to classifying the input images

as either face or non-face images. The statistical data of the method shows the efficiency of the method. The comparison shows that the method has higher precision rate. But when the number of faces increases, detection rate is going to be decreased and number of misses increases. The classification accuracy is few lower than detection. By increasing the number of training images, the accuracy rate can be increased.

Chapter 5

Conclusion

5.1 Summery of the Proposed Method

The first contribution of this thesis is the way of enhancing human skin detection step. An improved human skin modeling was created to achieve more reasonable skin segmentation for different human races in color image by reducing similar skin colors. After labeling, the proposed face detection algorithm uses the region properties for detecting human face. Different region properties are described in this research paper. Different properties have different conditions for rejecting non-human faces. Finally, morphological operation is performed for removing noise. The proposed algorithm also provides a good classification system i.e. deciding whether an image consists of a face or not. SVM classification is implemented for this task which is based on SFTA texture features. By using this feature, features are extracted from face and non-face image. Finally, training and classification of the test image is done by SVM classifier. These images were taken in various background, camera face pose and illumination conditions. The method is sensitive to multiple faces and background very similar to skin color but has good detection rate for single face with insensitive to pose, skin color and illumination condition. The detection rate is about 96.23%. The classification accuracy is 88.17%. The classification result will be increased, if there are more images for training. Another way is to increase the number of threshold used here. This will increase the number of feature and thus classification result may increase.

5.2 Limitations

The method is sensitive to multiple faces and background that is very much similar to skin color.

5.3 Future Recommendations

In the proposed method the detection rate is going to be decreased when the number of faces is five or more and background region is very much similar to the skin color. In future, the future work of the method is to introduce a faster method which works successfully for multiple faces and background similar to skin color. After Face Detection, facial recognition of various persons (identifying various persons in terms of their similar images) may be another future work for supporting modern virtual reality, intelligent human machine interface, robotics, and video phone system and computer vision communication.

Bibliography

[1] H.A. Alabbasi, F. Moldoveanu, "*Human face detection from images, based on skin color*", 18th Int. Conference on System Theory, Control and Computing, Sinaia, Romania, Oct. 2014.

[2] T. Ahonen, A. Hadid, and M. Pietikainen, "*Face Description with Local Binary Patterns:Application to Face Recognition*", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 28, no.12, Dec. 2006.

[3] P.T. Pham-Ngoc, " *Color-based Robust Multi-face Detection under Occlusion, Connection, Attitude and Illumination Changes*" ,SICE-ICCAS,IFOST,Korea,2006.

[4] P. Viola, M.J. Jones, "*Robust Real-Time Face Detection*", Int. Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, 2004.

[5] Prashanth K. G and Shashidhara M, "*Skin Color Segmentation for Detecting Human Face Region in Image*", International Conference on Communication and Signal Processing, Apr. 2014, doi: 978-1-4799-3358-7114.

[6] Li ZOU and Sei-ichiro KAMATA , "*Face Detection in Color Images Based on Skin Color Models*", 2010,IEEE.

[7] Y. Suzuki and T. Shibata, "*An edge-based face detection algorithm robust against illumination, focus, and scale variations*", in Proc. EUSIPCO 2004, pp. 2279-2282, Vienna, Austria, September 6-10, 2004.

[8] Srinivasulu Asadi, Dr.Ch.D.V.Subba Rao, V.Saikrishna, "*A Comparative study of Face Recognition with Principal Component Analysis and Cross-Correlation Technique",* International Journal of Computer Applications (0975 – 8887)**,** Volume 10– No.8, November 2010.

[9] Quan Wang, "*Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models*", 2012.

[10] Jebara, T. S., and Pentland, "*Parameterized structure from motion from 3D adaptive feedback tracking of faces*", In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 144-150, 1997.

[11] Fleck, M. M., Forsyth, D. A., and Bregler, "*Finding naked people*", In Proceedings 4th European Conference on Computer Vision, Springer, UK, vol. 2, pp.593-602, 1996.

[12] Kjeldsen, R., and Kender, "*Finding skin in color images*", In Proceedings 2nd International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society Press, Vermont, pp. 312-318, 1996.

[13] Dai, Y., and Nakano, *"Face-texture model-based on SGLD and its application in face detection in a color scene"*, Pattern Recognition, vol. 29, no. 6, pp. 1007-1017, 1996.

[14] C. Traina Jr., A. J. M. Traina, L. Wu, and C. Faloutsos, "*Fast feature selection using fractal dimension*," in Brazilian Symposium on Databases (SBBD), João Pessoa, Brazil, 2000, pp. 158–171.

[15] A. F Costa, G. Humpire-Mamani, A. J. M. Traina, "*An Efficient Algorithm for fractal analysis of textures*".

Appendix

```
clear;clc;
srcFiles = dir('D:\fd\Data_face\*.jpg');
len=length(srcFiles);
hueimg=zeros(1,2940);
cbimg=zeros(1,2940);
crimg=zeros(1,2940);

for i = 1 : len
    filename = strcat('D:\fd\Data_face\',srcFiles(i).name);
    im = imread(filename);

    hsvim=rgb2hsv(im);

    hue=hsvim(:,:,1);


    cb = 128+ 0.148* im(:,:,1) - 0.291* im(:,:,2) + 0.439 * im(:,:,3);
    cr = 128+ 0.439 * im(:,:,1) - 0.368 * im(:,:,2) -0.071 * im(:,:,3);

end

  reshapedHue=reshape(hue,[1,2940]);
  reshapedcb=reshape(cb,[1,2940]);
  reshapedcr=reshape(cr,[1,2940]);
    j=1;
    for k=1:1:2940
    hueimg(j)=reshapedHue(k);
    cbimg(j)=reshapedcb(k);
    crimg(j)=reshapedcr(k);
    j=j+1;
    end

meanHue=mean(hueimg);
meancb=mean(cbimg);
meancr=mean(crimg);


stdHue=std2(hueimg);
stdcb=std2(cbimg);
stdcr=std2(crimg);


[file, pathname] = uigetfile('*.jpg','Load Image');
cd('D:\fd\Data');
rgbInputImage=imread(file);
figure;imshow(rgbInputImage);
```

```
title('input image');
I=double(rgbInputImage);
j=rgb2hsv(rgbInputImage);
g=rgb2ycbcr(rgbInputImage);
figure;imshow(j)
figure;imshow(g);


[hue,s,v]=rgb2hsv(I);
cb = 128 + 0.148* I(:,:,1) - 0.291* I(:,:,2) + 0.439 * I(:,:,3) ;
cr = 128 + 0.439 * I(:,:,1) - 0.368 * I(:,:,2) -0.071 * I(:,:,3) ;

[w h]=size(I(:,:,1));

for i=1:w
for j=1:h
if hue(i,j)>=0.01 & hue(i,j)<=0.1  & cb(i,j)>=140 & cb(i,j)<=195 &
cr(i,j)>=140 & cr(i,j)<=165
binaryImage(i,j)=1;
else
binaryImage(i,j)=0;
end
end
end

figure;imshow(binaryImage);
title('binary image');

[m,n] = size(binaryImage);
filled=zeros(m,n);
[labelBW,num] = bwlabel(binaryImage, 8);
figure;imshow(labelBW);
title(['Labeled Regions,(',num2str(num),' regions)']);

a = regionprops(labelBW,'Area');
area=cat(1,a.Area);
sum=0;
for i=1:1:num
    sum=sum+area(i);
end
averageArea=sum/num;
region_index = find(area>averageArea);
[m,n]=size(labelBW);
areaBW=zeros(m,n);

 for i=1:length(region_index)

  [x,y] = find(bwlabel(labelBW) == region_index(i));
   bwsegment = bwselect(labelBW,y,x,8);
   areaBW=areaBW+bwsegment;
end
figure,imshow(areaBW);
title(['After Area Test,(',num2str(length(region_index)),' regions)']);
```

```
AreaBW=bwlabel(areaBW,8);
e = regionprops(AreaBW,'EulerNumber');
eulers=cat(1,e.EulerNumber);
holes=1-eulers;

region_index = find(holes>=1);
[m,n]=size(AreaBW);
eulerBW=zeros(m,n);
for i=1:length(region_index)

   [x,y] = find(bwlabel(AreaBW) == region_index(i));
   bwsegment = bwselect(AreaBW,y,x,8);
   eulerBW=eulerBW+bwsegment;
end

figure,imshow(eulerBW);
title(['After Euler Test,(',num2str(length(region_index)),'regions)']);

eulerBW=bwlabel(eulerBW,8);
 [m,n]=size(eulerBW);
ecBW=zeros(m,n);
ec = regionprops(eulerBW,'Eccentricity');
ecity=cat(1,ec.Eccentricity);
region_index = find(ecity <.89);

for i=1:length(region_index)

    [x,y] = find(bwlabel(eulerBW) == region_index(i));
   bwsegment = bwselect(eulerBW,y,x,8);
   ecBW=ecBW+bwsegment;
end

figure;imshow(ecBW);
title(['After Eccentricity Test,(',num2str(length(region_index)),'
regions)']);

ecBW=bwareaopen(ecBW,1960);
[m,n]=size(ecBW);
[labels,num] = bwlabel(ecBW,8);
[aspect_ratio]=get_aspect(labels);
region_index = find(aspect_ratio < 2.2 & aspect_ratio >1);
aspectBW=zeros(m,n);

for i=1:length(region_index)

    [x,y] = find(bwlabel(ecBW) == region_index(i));
    bwsegment = bwselect(ecBW,y,x,8);
    aspectBW=aspectBW+bwsegment;

end
figure;imshow(aspectBW);
```

```
title(['After Aspect Ratio
Test,(',num2str(length(region_index)),'regions)']);
filledBW = imfill(aspectBW,'holes');

%%Morphological Operation
se = strel('disk',14);
bw = imclose(filledBW,se);
figure,imshow(bw);
title('Morphological Operation');

I(:,:,1)=I(:,:,1).*bw;
I(:,:,2)=I(:,:,2).*bw;
I(:,:,3)=I(:,:,3).*bw;
figure;imshow(uint8(I));
title('Superimposed image');

L=bwlabel(bw,8);
BB  = regionprops(L, 'BoundingBox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);
figure,imshow(rgbInputImage);
title('Original Image with Bounding Box');
%%%%%%% show the target in rectangular frame
[s1 s2]=size(BB2);

for k=3:4:s2-1
     rectangle('Position',[BB2(1,k-2),BB2(1,k-
1),BB2(1,k),BB2(1,k+1)],'Linewidth', 4,'EdgeColor','r' )
end

img=rgbInputImage;
hold on
N=size(BB2,1);
handles.N=N;
counter=1;
for i=1:N
    face=imcrop(img,BB2(i,:));
    savenam = strcat('D:\fd\' ,num2str(counter), '.jpg'); %this is
where and what your image will be saved
    baseDir  = 'D:\fd\cropped face\';
    newName  = [baseDir num2str(counter) '.jpg'];
    handles.face=face;
    while exist(newName,'file')
        counter = counter + 1;
        newName = [baseDir num2str(counter) '.jpg'];
    end
    fac=imresize(face,[112,92]);
    imwrite(fac,newName);
figure;imshow(face);
title('Cropped Face');
 pause(.5);
end
```

```
%%%%%%%%% SFTA %%%%%%%%%%%%
I = im2uint8(I);
    if size(I,3) ~= 1
        I = rgb2gray(I);
    end;

    T = otsurec( I, nt );
    dSize = numel(T) * 6;
    D = zeros(1, dSize);
    pos = 1;
    for t = 1 : numel(T)
        thresh = T(t);
        Ib = im2bw(I, thresh);
        Ib = findBorders(Ib);

        vals = double(I(Ib));

        D(pos) = hausDim(Ib);
        pos = pos + 1;

        D(pos) = mean(vals);
        pos = pos + 1;

        D(pos) = numel(vals);
        pos = pos + 1;
    end;

    T = [T; 1.0];
    range = getrangefromclass(I);
    range = range(2);

    for t = 1 : (numel(T) - 1)
        lowerThresh = T(t);
        upperThresh = T(t + 1);

        Ib = I > (lowerThresh * range) & I < (upperThresh * range);
        Ib = findBorders(Ib);

        vals = double(I(Ib));

        D(pos) = hausDim(Ib);
        pos = pos + 1;

        D(pos) = mean(vals);
        pos = pos + 1;

        D(pos) = numel(vals);
        pos = pos + 1;
    end;
end
```

```
%%%%%%%%% otsu%%%%%%%%%%%
function [ T ] = otsurec(I, ttotal)
if ~isempty(I)
    I = im2uint8(I(:));

    num_bins = 256;
    counts = imhist(I, num_bins);
    T = zeros(ttotal, 1);
    otsurec_helper(1, num_bins, 1, ttotal);

else
    T = [];
end;

    function [] = otsurec_helper(lowerBin, upperBin, tLower, tUpper)
        if ((tUpper < tLower) || (lowerBin >= upperBin))
            return
        else
            level = otsu(counts(ceil(lowerBin) : ceil(upperBin))) +
lowerBin;
            insertPos = ceil((tLower + tUpper) / 2);
            T(insertPos) = level / num_bins;
            otsurec_helper(lowerBin, level, tLower, insertPos - 1);
            otsurec_helper(level + 1, upperBin, insertPos + 1, tUpper);
        end
    end
end

function [ pos ] = otsu(counts)
    p = counts / sum(counts);
    omega = cumsum(p);
    mu = cumsum(p .* (1:numel(counts))');
    mu_t = mu(end);

    previous_state = warning('off', 'MATLAB:divideByZero');
    sigma_b_squared = (mu_t * omega - mu).^2 ./ (omega .* (1 - omega));
    warning(previous_state);
    maxval = max(sigma_b_squared);
    isfinite_maxval = isfinite(maxval);
    if isfinite_maxval
        pos = mean(find(sigma_b_squared == maxval));
    else
        pos = 0;
    end;

%%%%%%%%% Border Image %%%%%%%%%%

function [ Im ] = findBorders( I )
    Im = false(size(I));
    I = padarray(I, [1, 1], 1);
    [h w] = size(Im);
    bkgFound = false;
```

```
    for row = 1 : h
        for col = 1 : w
            if I(row + 1, col + 1)

                bkgFound = false;
                for i = 0:2
                    for j = 0:2
                        if ~I(row + i, col + j)
                            Im(row, col) = 1;
                            bkgFound = true;
                            break;
                        end;
                    end;

                    if bkgFound
                        break;
                    end;
                end;
            end;
        end;
    end;
end

%%%%%%%%% Fractal Dimension %%%%%%%%%%
function [ D ] = hausDim( I )
    maxDim = max(size(I));
    newDimSize = 2^ceil(log2(maxDim));
    rowPad = newDimSize - size(I, 1);
    colPad = newDimSize - size(I, 2);
    I = padarray(I, [rowPad, colPad], 'post');

    boxCounts = zeros(1, ceil(log2(maxDim)));
    resolutions = zeros(1, ceil(log2(maxDim)));

    iSize = size(I, 1);
    boxSize = iSize;
    boxesPerDim = 1;
    idx = 0;
    while boxSize >= 1
        boxCount = 0;

        minBox = (1: boxSize: (iSize - boxSize) + 1);
        maxBox = (boxSize: boxSize: iSize);

        for boxRow = 1:boxesPerDim
            for boxCol = 1:boxesPerDim
                objFound = false;
                for row = minBox(boxRow) : maxBox(boxRow)
                    for col = minBox(boxCol) : maxBox(boxCol)
                        if I(row, col)
                            boxCount = boxCount + 1;
                            objFound = true;
                            break;
                        end;
                    end;
```

```
                    if objFound
                        break;
                    end;
                end;
            end;
        end;

        idx = idx + 1;
        boxCounts(idx) = boxCount
        resolutions(idx) = 1 / boxSize;

        boxesPerDim = boxesPerDim * 2;
        boxSize = boxSize / 2
    end;

    D = polyfit(log(resolutions), log(boxCounts), 1);
    D = D(1);
end

%%%%%%%%% SVM Classification %%%%%%%%%%

clc;
clear all;
tic;
whos;

while (1==1)

    choice=menu('Face Detection',...
    'Face Database',...
    'Non Face Database',...
    'Test Database',...
    'Train Test',...
    'Exit');

    if (choice == 1)
        clc;
        clear all;
        choice = 1;

        pause(0.01);
        folder_name = uigetdir;
        filenames       =   dir(fullfile(folder_name, '*.jpg'));
        total_images    =   numel(filenames);
        D = zeros(total_images,24);

    for i = 1:total_images
 full_name= fullfile(folder_name, filenames(i).name);
I= imread(full_name);
            fprintf('...\n..\n.  Image %d\n ',i);

            % Extract feature
```

```
features = sfta(I, 4);
D(i,1:24) = features(1:24);
        end
        AllData = table(D);
        writetable(AllData,['face.xlsx'],'Sheet',1);
    end

    if (choice == 2)
        clc;
        clear all;
        choice = 2;

        pause(0.01);
        folder_name = uigetdir;
        filenames = dir(fullfile(folder_name, '*.jpg'));
total_images = numel(filenames);

        D = zeros(total_images,24);

        for i = 1:total_images

            full_name= fullfile(folder_name, filenames(i).name);
            I = imread(full_name);
            fprintf('...\n..\n.  Image %d\n ',i);

            % Extract feature
            features = sfta(I, 4);
            D(i,1:24)=features(1:24);
        end
        AllData = table(D);
        writetable(AllData,['non-face.xlsx'],'Sheet',1);
    end

    if (choice == 3)
        clc;
        clear all;
        choice = 3;

        pause(0.01);
        folder_name = uigetdir;
        filenames       =   dir(fullfile(folder_name, '*.jpg'));
total_images    =   numel(filenames);

        D = zeros(total_images,24);
        for i = 1:total_images

            full_name= fullfile(folder_name, filenames(i).name);
            I = imread(full_name);
fprintf('...\n..\n.  Image %d\n ',i);
            testfile_names{i} = full_name;

            % Extract feature
            features = sfta(I, 4);
            D(i,1:24)=features(1:24);
        end
```

```
    AllData = table(D);
    writetable(AllData,['testset.xlsx'],'Sheet',1);
    save('testFileNames.mat','testfile_names');
end

if(choice == 4)

    clc;
    clear all;
    choice = 4;

    % load data
    Face = xlsread('face.xlsx');
    Non_face = xlsread('non-face.xlsx');
    [row, col] = size(Face);
    [row2, col2] = size(Non_face);

    % creating training group
    TT1=Face(1:row,1:col);
    TT2=Non_face(1:row2,1:col2);
    TT_Set = [TT1;TT2];

    TX1 = ones(row,1);
    TX2 = ones(row2,1)*2;
    TTgroup = [TX1;TX2];

    %creating testing group
    load('testFileNames.mat');
    result = testfile_names';
    testset=xlsread('testset.xlsx');
    [m,n]=size(testset);
    facetest = testset(1:m,1:n);

    % training using SVM
    SVMStruct = svmtrain(TT_Set,TTgroup, 'kernel_function', 'rbf');

    %testing using SVM model
    Group = svmclassify(SVMStruct,facetest);
    for i=1:1:size(Group,1)
        result{i,2} = Group(i);

        if(Group(i)==1)
            result{i,3} = 'Face';
        else
            result{i,3} = 'Non-Face';
        end

    end

    % saving results
    save('SVMStruct.mat','SVMStruct');
    save('Result.mat','result');

    AllData = table(result);
```

```
        writetable(AllData,['Result.xlsx'],'Sheet',1);

        fprintf('Finished! please check Result.mat or Result.xlsx file');
    end

    if (choice == 5)
        clc;
        clear all;
        choice = 5;

        clear all;
        clc;
        close all;
        return;
    end

end
```