2003

Recognition of Human Faces

Ferdousi, Afroza

University of Rajshahi

RECOGNITION OF HUMAN FACES

A thesis submitted to the University of Rajshahi for the requirement of the degree of Master of Philosophy in Applied Physics and Electronics.

Dept. of Applied Physics and Electronics
Faculty of Science
University of Rajshahi
Rajshahi-6205
Bangladesh
June, 2003

Submitted By:
Afroza Ferdousi
Roll No. 07
Session: July, 1997
University of Rajshahi
Rajshahi-6205
Bangladesh

DEDICATED TO

My husband "Md. Farukuzzaman Khan"
and daughter "Faria Ferdousi Khan"

DECLARATION

Except where full references have been given this thesis contains the independent original work of the author. This thesis has not been submitted before, nor it is being submitted anywhere else at the same time for the award of any degree.

Ferdousi 21.06.03

(Afroza Ferdousi)

M. Phil. Fellow
Department of Applied Physics and Electronics
Rajshahi University
Rajshahi

DR. RAMESH CHANDRA DEBNATH
Professor
Dept. of Applied Physics & Electronics
Rajshahi University, Rajshahi.
Tel: Office : (0721) 750041-9-/4101
Res : (0721) 750404
Email: ramesh@librabd.net

ডঃ রমেশ চন্দ্র দেবনাথ
প্রফেসর
ফলিত পদার্থবিজ্ঞান ও ইলেকট্রনিক্স বিভাগ
রাজশাহী বিশ্ববিদ্যালয়, রাজশাহী।
টেলিঃ অফিসঃ (০৭২১) ৭৫০০৪১-৯/৪১০১
বাসা ঃ (০৭২১) ৭৫০৪০৪
ই-মেইলঃ ramesh@librabd.net

Date. 21.06.03

SUPERVISORS RECOMMENDATION

Certified that the thesis entitled "Recognition of Human Faces" submitted by Mrs. Afroza Ferdousi for the degree of Master of Philosophy in Applied Physics and Electronics has been done under my supervision. I recommend its submission for evaluation with my full satisfaction.

Debnath 22.06.03

(Dr. Ramesh Chandra Debnath)

ACKNOWLEDGMENT

It is a depth of gratitude and pleasant duty to express my obligation to **Dr. Ramesh Chandra Debnath**, Professor, Deptt. of Applied Physics and Electronics, University of Rajshahi, for his constant guidance, advice, encouragement and every possible help throughout the work and preparation of this thesis.

I am also grateful to **Dr. Md. Abdus Sobhan(2),** Professor and Chairman, Deptt. of Applied Physics and Electronics, University of Rajshahi, for his valuable advice and kind co-operation.

All of my teachers of the Deptt. of Applied Physics and Electronics, University of Rajshahi, specially **Mr. Md. Rezaul Islam** were very much helpful to me throughout the work and preparation of this thesis. I am very much grateful to them.

I am thankful to **Mr. Md. Hasnat Kabir,** Assistant Programmer, Deptt. of Applied Physics and Electronics, and **Rahad**, **Masum, Arif, Ivan, Hasan** student of University of Rajshahi, for their continuous co-operation.

Finally, I am thankful to all the member of my family, specially to my father **Professor Dr. Md. Afsar Uddin** and my husband **Md. Farukuzzaman Khan** for their encouragement and every possible support to this research.

The Author

Recognition of Human Faces

Abstract

This dissertation is devoted to the development of some image processing algorithms and implementation of these algorithms in face recognition system. At first face images were scanned and stored as windows bitmap file (bmp), in order to extract features. In features extraction, the original face, edge-detected face and thresholded-face were taken to extract three types of features. The face was divided into small grids. Each grid of the face includes 10x10 matrix of gray level values. Thus 100 data were taken from a grid and the average of these 100 data were taken as a feature. The recognition systems include the template matching by distance measurement (Hamming and Euclidean) between known and unknown features and a neural network. Fifty faces of ten persons were used to test the system. The highest recognition rate was 92%. Using features from thresholded face and neural network as recognition tool achieved this result.

SUMMERY

Face recognition has been a difficult problem in the field of computer vision for many years. Face recognition requires the ability to recognize identity despite many variations in appearance that face can have in a scene. We propose a recognition system capable of identifying human faces efficiently and with invariance to rotation, deformation and illumination. The objective of this research work was to develop a face recognition system based on visual face features. In this dissertation, an analysis of some algorithms of Human Face Recognition and implementation of these algorithms in recognition system has been discussed. The recognition methods and results are studied and discussed. The research has made it possible to conclude on the better feature extraction method, as well as on the recognition method. It also made a suggestion for future research on Human Face Recognition.

A lot of different theories and paradigms have already been developed for many applications including security, surveillance, gaze-based control, affective computing, speech recognition assistance, video compression and animation. However, to date, no complete solution has been proposed that allows the recognition of faces in real (un-contrived) images. These matters, which include the discussion about the past and present status of Face recognition research, have been given in chapter-1 of this thesis.

Face recognition is a part of image processing. Image processing deals with the improvement of images for human perception, which is a general term for the wide range of techniques that exist for manipulating and modifying images in various ways. In the second chapter entitled "Image Fundamentals" describes the basic terms of image, fundamentals of image processing related with this dissertation as well as their various complexities, digital image representation and image filtering. Until now various innovative methods to locate and track

faces have been proposed. Basically they used the visual cues. Some typical systems are also introduced in chapter-2.

An Artificial Neural Network is used in this face recognition system. In the third chapter entitled "Fundamentals of Artificial Neural network", the fundamentals of Artificial Neural Network are described.

Obviously a good feature may produce a good result for any recognition system. A study of three types of face images from which meaningful feature may be collected for face recognition are used in this research. These are Original face images, Edge detected face images and thresholding face images. The steps required to extract features are described in chapter-4. The recognition process can be considered as a two-stage device. The first stage is feature extraction and the second is classification. It is not necessary to know the formal mathematical model of classification but suitable network architecture and a sufficient training set are necessary to obtain a solution through experimentation. So the use of distance measurement techniques in template matching and Neural Networks has also been included in chapter-4.

The software developed for the analysis and recognition of Human Face includes six separate program modules that are written to cover all the operations. They are the Main Program, the Edge Detection Program, the Image Thresholding Program, the Feature Extraction Program, the Recognition Program and the Neural Network Program. The programs are written in C language and compiled with Turbo C++ compiler. The techniques used in designing the program, the algorithms of various modules and the flowchart of the main program are described in chapter-5.

Chapter-6 of this dissertation explores the performance of the system for various features and recognition methods. In first step, experiments were

conducted to find best parameters and target output pattern for neural network. The target output patterns were Unit Matrix, Hamming code and 7-bit ASCII code. This chapter also describes the experiments to find best features and recognition methods. All of these tests were conducted with a reference database of ten faces. Total 50 unknown faces were included to test the systems performance.

The results of these experiments may be summarized as below.

- The best performing (92%) net parameters includes *features collected after threshold* as input pattern and *Unit Matrix* as target output. Other parameters are Hidden Unit=20, spr.=0.4, spread=0.25, eta1=0.1 and eta2=0.2
- This best performing net has also an average error of 0.000207 for *Unit Matrix* as target output. This was one source of recognition error.
- Neural Network and Template matching by Hamming and Euclidean distance measurement perform nearly the same, Neural Network = 92%, Euclidean=92% and Hamming = 90%.
- For the same set of network parameters, *features collected after threshold* proves better among the others.

After a comparative study of experimental results from different feature extraction methods and recognition methods, analysis of sources of errors, limitations etc, we concludes as below.

- From our study, it is clear that the features collected from thresholded face are proved stronger in recognition of human faces.
- The recognition of human faces is not so dependent on these methods but Artificial Neural Network should be used as better future.
- The recognition results are satisfactory and the developed system may be used in any human or object identification system where reference database is not so large.

- For better result, Motion Field Histogram, Eigenspace Modeling or other suitable features may be used and Convolutional Neural Network, Hidden Markov Model (HMM) or any other suitable recognition tool should be used in future research work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter-1

Introduction

1

INTRODUCTION

1.1 Introduction

Face detection and recognition has been a difficult problem in the field of computer vision for several years. Although humans perform the task in an effortless manner, the underlying computations within the human visual system are of tremendous complexity. The seemingly trivial task of finding and recognizing faces is the result of millions of years of evolution and we are far from fully understanding how the brain performs it.

Furthermore, the ability to find faces visually in a scene and recognize them is critical for humans in their everyday activities. For a long time, the dream of computer scientist is to develop the computer as an intelligent machine as human. The human brain is so powerful that it can solve a wide variety of problems from thinking, remembering, feeling and learning. In this time computer generation is known as fifth generation computer and the computer scientist have come up to create intelligence in these machines.

A lot of different theories and paradigms have already been developed for many applications including security, surveillance, gaze-based control, affective computing, speech recognition assistance, video compression and animation. However, to date, no complete solution has been proposed that allows the recognition of faces in real (un-

contrived) images. This introduction begins with an outline of the main issues and constraints that need to be addressed in face recognition.

The objective of this research work is to develop a face recognition system based on visual face features. Face recognition and human civilization are related to each other and it is a part and percel of every day life. The other names of Face Recognition are Person Identification, Human Face Detection. As a part of the human civilization, human face recognition as well as person identification is required. Human face recognition is mostly required to solve many problems in our life such as information retrieval, automatic banking, control of access to security areas and so on. In this research a tiny effort has been carried out to develop the Face Recognition process. This effort will reach this goal whenever it will be able to contribute something in the Human Face Recognition system.

1.2 Importances and Motivation

Given the requirement for determining people's identity, the obvious question is what technology is best suited to supply this information? There are many different identification technologies available, many of which have been in widespread commercial use for years. The most common person verification and identification methods today are Password/PIN (Personal Identification Number) systems, and Token systems (such as your driver's license). Because such systems have trouble with forgery, theft, and lapses in users' memory, there has developed considerable interest in biometric identification systems, which use pattern recognition techniques to identify people using their physiological characteristics. Fingerprints are a classic

example of a biometric identification; newer technologies include retina and iris recognition.

While appropriate for bank transactions and entry into secure areas, such technologies have the disadvantage that they are intrusive both physically and socially. They require the user to position their body relative to the sensor, and then pause for seconds to declare themselves. This `pause and declare' interaction is unlikely to change because of the fine-grain spatial sensing required. Moreover, there is an `oracle-like' aspect to the interaction: since people cannot recognize other people using this sort of data, these types of identification do not have a place in normal human interactions and social structures.

Face recognition from video and voice recognition have a natural place in the next-generation -- they are unobtrusive (able to recognize at a distance without requiring a `pause and present' interaction), are usually passive (do not require generating special electro-magnetic illumination), do not restrict user's movement, and are now both low power and inexpensive. So *Human Face Recognition System* is important for controlling to secure facilities, personal information, services like banking and credit checks etc.

Although some limited research works have been done in the field of Human Face Recognition system, but these are not sufficient in the view of individual person identification. Perhaps most important, however, is that humans identify other people by their face and voice, therefore are likely to be comfortable with systems that use face and voice recognition.

Recognition of Human Face System has a broad range of applications, as described below.

Person identification

Person identification has already been used in

- Forensic applications, such as in creating a composite sketch of a suspect and then finding a match in a mug shot database.
- Personal identification for credit cards, driver's license, passports, employee ID.
- Access control, such as the access to check-cashing ATMs, buildings or rooms.

Video coding, video databases and teleconferencing system

It is well known that people are most sensitive to coding errors in facial features. The coder would encode very precisely facial features (such as eyes, mouth, nose, etc) and less precisely the rest of the picture.

Human–computer interaction

Currently some computer games can be played with head movement, instead of mouse or keyboard (but the player must wear headgear).

Security monitoring

A face tracking system can be used as a security system in shopping malls, other public areas, or private houses.

Banking system

In banking system, human face detection system is a much-used phenomenon.

Information retrieval

In information retrieval system human face recognition system plays a very important role.

1.3 Fundamental Issues in Face Recognition

Face recognition requires the ability to recognize identity despite many variations in appearance that the face can have in a scene. The face is a 3D object, which is illuminated from a variety of light sources and surrounded by arbitrary background data (including other faces). Therefore, the appearance a face has when projected onto a 2D image can vary tremendously. If we wish to develop a system capable of performing non-contrived recognition, we need to find and recognize faces despite these variations.

Additionally, our detection and recognition scheme must also be capable of tolerating variations in the faces themselves. The human face is not a unique rigid object. There are billions of different faces and each of them can assume a variety of deformations. Inter-personal variations can be due to race, identity or genetics, while inter-personal variations can be due to deformations, expression, aging, facial hair, cosmetics and facial paraphernalia.

Furthermore, the output of the detection and recognition system has to be accurate. A recognition system has to associate an identity or name for each face it comes across by matching it to a large database of individuals. Simultaneously, the system must be robust to typical image-acquisition problems such as noise, video-camera distortion and image resolution. Thus, we are dealing with a multi-dimensional detection and recognition problem. One final constraint is the need to maintain the usability of the system on contemporary computational devices. In other words, the processing involved, should be efficient with respect to run-time and storage space.

1.4 History and Mathematical Framework

The subject of face recognition is as old as computer vision, both because of the practical importance of the topic and also of theoretical interest from cognitive scientists. Despite the fact that other methods of identification (such as fingerprints, or iris scans) can be more accurate, face recognition has always remained a major focus of research because of its non-invasive nature and because it is people's primary method of person identification.

In following years many researchers tried face recognition schemes based on edges, inter-feature distances, and other neural net approaches. While several were successful on small databases of aligned images, none successfully addressed the more realistic problem of large databases where the location and scale of the face is unknown.

Twenty years ago the problem of face recognition was considered among the hardest in Artificial Intelligence (AI) and computer vision. Surprisingly, however, over the last decade there has been a series of successes that have made the general person identification enterprise appear not only technically feasible but also economically practical.

Kirby and Sirovich (1989)[1] later introduced an algebraic manipulation which made it easy to directly calculate the eigenfaces, and showed that fewer than 100 were required to accurately code carefully aligned and normalized face images. Turk and Pentland (1991)[2] then demonstrated that the residual error when coding using the eigenfaces could be used both to detect faces in cluttered natural imagery, and to determine the precise location and scale of faces in an image. They then demonstrated that by coupling this method for detecting and localizing faces with the eigenface recognition method, one could achieve reliable, real-time recognition of faces in a minimally

constrained environment. This demonstration that simple, real-time pattern recognition techniques could be combined to create a useful system sparked an explosion of interest in the topic of face recognition.

The apparent tractability of face recognition problem has produced a huge surge of interest from both funding agencies and from researchers themselves. It has also spawned several thriving commercial enterprises. There are now several companies that sell commercial face recognition software that are capable of high-accuracy recognition with databases of over 1,000 people.

These early successes came from the combination of well-established pattern recognition techniques with a fairly sophisticated understanding of the image generation process. In addition, researchers realized that they could capitalize on regularities that are peculiar to people, for instance, that human skin colors lie on a one-dimensional manifold (with color variation primarily due to melanin concentration), and that human facial geometry is limited and essentially 2-D when people are looking toward the camera. Today, researchers are working on relaxing some of the constraints of existing face recognition algorithms to achieve robustness under changes in lighting, aging, rotation-in-depth, expression and appearance (beard, glasses, makeup) -- problems that have partial solution at the moment.

The dominant representational approach that has evolved is descriptive rather than generative. Training images are used to characterize the range of 2-D appearances of objects to be recognized. Although initially very simple modeling methods were used, the dominant method of characterizing appearance has fairly quickly become estimation of the probability density function (PDF) of the image data for the target class.

For instance, given several examples of a target class Ω in a two-dimensional representation of the image data, it is straightforward to model the probability distribution function P(X/Ω) of its image-level features X as a simple parametric function (e.g., a mixture of Gaussians), thus obtaining a two-dimensional, computationally efficient *appearance model* for the target class [3].

1.5 Current State of Face Recognition

Research in intensity image face recognition generally falls into two categories [4]: holistic (global) methods and feature-based methods. Feature-based methods rely on the identification of certain fiducial points on the face such as the eyes, the nose, the mouth, etc. The location of those points can be determined and used to compute geometrical relationships between the points as well to analyze the surrounding region locally. Thus, independent processing of the eyes, the nose, and other fiducial points is performed and then combined to produce recognition of the face. Since detection of feature points precedes the analysis, such a system is robust to position variations in the image. Holistic methods treat the image data simultaneously without attempting to localize individual points. The face is recognized as one entity without explicitly isolating different regions in the face. Holistic techniques utilize statistical analysis, neural networks and transformations. They also usually require large samples of training data. The advantage of holistic methods is that they utilize the face as a whole and do not destroy any information by exclusively processing only certain fiducial points. Thus, they generally provide more accurate recognition results. However, such techniques are sensitive to variations in position, scale and so on, which restrict their use to standard, frontal mug shot images[5].

Early attempts at face recognition were mostly feature-based. These include Kanade's[6] work where a series of fiducial points are detected using relatively simple image processing techniques (edge maps, signatures, etc.) and their Euclidean distances are then used as a feature vector to perform recognition. Yuille, Cohen and Hallinan proposed more sophisticated feature extraction algorithms. These use deformable templates that translate, rotate and deform in search of a best fit in the image. Often, these search techniques use a knowledge-based system or heuristics to restrict the search space with geometrical constraints (i.e. the mouth must be between the eyes)[7]. Unfortunately, such energy minimization methods are extremely computationally expensive and can get trapped in local minima. Furthermore, a certain tolerance must be given to the models since they can never perfectly fit the structures in the image. However, the use of a large tolerance value tends to destroy the precision required to recognize individuals on the basis of the model's final best-fit parameters. Nixon proposes the use of Hough transform techniques to detect structures more efficiently[8]. However, the problem remains that these detection-based algorithms need to be tolerant and robust and this often makes them insensitive to the minute variations needed for recognition. Recent research in geometrical, feature-based recognition[9] reported 95% recognition. However, the 30 features points used for each face were manually extracted from each image. Had some form of automatic localization been used, it would have generated poorer results due to lower precision. In fact, even the most precise deformable template matching algorithms such as Roeder's[10] and Colombo's feature detectors generally have significant errors in detection. This is also true for other feature detection schemes such as Reisfeld's symmetry operator and Graf's filtering and morphological operations. Essentially,

current systems for automatic detection of fiducial points are not accurate enough to obtain high recognition rates exclusively on the basis of simple geometrical statistics of the localization.

Holistic techniques have recently been popularized and generally involve the use of transforms to make the recognition robust to slight variations in the image. Rao[11] develops an iconic representation of faces by transforming them into a linear combination of natural basis functions. Manjunath[12] uses a wavelet transform to simultaneously extract feature points and to perform recognition on the basis of their Gabor wavelet jets. Such techniques perform well since they do not exclusively compute geometric relationships between fiducial points. Rather, they compare the jets or some other transform vector response around each fiducial point. Alternate transform techniques have been based on statistical training. For example, Pentland[13] uses the Karhunen-Loeve decomposition to generate the optimal basis for spanning mug-shot images of human faces and then uses the subsequent transform to map the faces into a lower-dimensional representation for recognition. Akamatsu[14] has also applied this technique on the Fourier-transformed images instead of the original intensity images. Recent work by Pentland involves modular eigenspaces where the optimal intensity decomposition is performed around feature points independently (eyes, nose and mouth). Pentland has also investigated the application of Karhunen-Loeve[15] decomposition to statistically recognize individuals on the basis of the spectra of their dynamic Lagrangian warping into a standard template. These transform techniques have yielded very high recognition rates and have quickly gained popularity. However, these non-feature based techniques do not fare well under pose changes and have difficulty with natural, un-contrived face images.

In most holistic face recognition algorithms, the face needs to be either segmented or surrounded by a simple background. Furthermore, the faces presented to the algorithms need to be roughly frontal and well illuminated for recognition to remain accurate. This is due to the algorithms' dependence on fundamentally linear or quasi-linear analysis techniques (Fourier, wavelet, Karhunen-Loeve decompositions, etc. are linear transformations). Thus, performance degrades rapidly under 3D orientation changes, non-linear illumination variation and background clutter (i.e. large, non-linear effects).

1.6 OBJECTIVES

The main objectives considered in this research are as follows:

- To extract and select effective image features that contains a face.
- To identify individual human face.
- To identify better system of recognition of human faces.
- To design an effective and reliable human face recognition system.

1.7 Organization of the Thesis

This dissertation is organized as follows:

In the first chapter named Introduction, past and present status of Face recognition research has been discussed. This introduction begins with an outline of the main issues and constraints that need to be addressed in face recognition. Subsequently, a survey is presented which outlines the face recognition research that has been performed to-date and the strengths and weaknesses of a variety of machine-based systems.

In the second chapter entitled "Image Fundamentals" describes the basic terms of image, fundamentals of image processing related with this dissertation as well as their various complexities, digital image representation and image filtering.

The fundamental steps of Artificial Neural Network using in this research purpose are described in the third chapter entitled "Fundamentals of Artificial Neural network".

The theoretical analyses of various features are given in the fourth chapter entitled "Feature Extraction and Recognition Methods". Three recognition methods that are adopted in this research are discussed in this chapter.

Fifth chapter titled as the "Software System for Human Face Recognition" includes the software system designed for this research. The algorithms, flowchart and techniques adopted in the software are discussed in this chapter.

The description of total experimental results is presented in chapter six titled as the "Experimental Results".

The discussion about the total work is included in the chapter seven titled as "Discussion and Conclusion". The conclusion of this paper and some suggestions for future research is also included is this chapter.

At last, Reference section and Appendices are included. Reference includes all the references used throughout the paper. Function prototypes, various forms of face images and bmp file structure are given in appendices.

Chapter-2

Image Fundamentals

IMAGE FUNDAMENTALS

2.1 Introduction

Face recognition is a part of image processing. Image processing deals with the improvement of images for human perception, which is a general term for the wide range of techniques that exist for manipulating and modifying images in various ways. The term digital image processing generally refers to the processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two dimensional data. Digital image processing is a rapidly evolving field with growing applications in science and engineering. Image processing has been developed in response to three major problems concerned with pictures:

- Picture digitization and coding to facilitate transmission, printing and storage of pictures.
- Picture enhancement and restoration in order to interpret more easily pictures of the surface of other planets taken by various probs.
- Picture segmentation and description as an early stage in Machine Vision.

2.2 Image[16]

An image is a picture, photograph, display or other form giving visual representation of an object or scene. In image processing, a *monochrome image* refers to a two-dimensional light intensity function f(x, y), where x and y are spatial coordinates and

the value of f at spatial coordinates (x, y) gives the intensity of the image and is proportional to the brightness of the image at that point. A *digital image* is an image f(x, y) that has been discretized both in spatial coordinates and in brightness. It is represented by a 2 – dimensional integer array, or a series of 2 – dimensional arrays, one for each color band. The digitized brightness value is called the *grey level* value. Each element of the array is called a *pixel* or *pel* derived from the term "picture element". Usually, the size of such array is a few hundred pixels by a few hundred pixels and there are several dozens of possible different grey levels. Thus a digital image looks like this:

$$f(x, y) = \begin{bmatrix} f(0,0)\, f(0,1) & \ldots\ldots & f(0,N-1) \\ f(1,0)\, f(1,1) & \ldots\ldots & f(1,N-1) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ f(N-1,0)\, f(N-1,1) & \ldots\ldots & f(N-1,N-1) \end{bmatrix}$$

with $0 \leq f(x,y) \leq G-1$ where usually N and G are expressed as integer powers of 2 ($N = 2^n$, $G = 2^m$).

2.2.1 Theoretical Background of Image[17]

As light is a form of energy, f(x, y) must be nonzero and finite i.e. ,

$$0 < f(x, y) < \infty \qquad (1.1)$$

The basic nature of f(x, y) may be characterized by two components:

- The amount of source light incident on the scene being viewed and
- The amount of light reflected by the objects in the scene.

Appropriately, they are called the *illumination* and *reflectance* components, and are denoted by i(x, y) and r(x, y), respectively. The functions i(x, y) and r(x, y) are combined as a product to form f(x, y):

$$f(x,y) = i(x,y)r(x,y) \quad (1.2)$$

where $0 < i(x, y) < \infty$ (1.3)

and $0 < r(x, y) < 1$ (1.4)

We call the intensity of a monochrome image f at coordinates (x, y) the *gray level* (l) of the image at that point. From equations (1.2) through (1.4), it is evident that l lies in the range

$$L_{min} \leq l \leq L_{max}$$

The interval [L_{min}, L_{max}] is called the *gray scale*, i.e. converting form of each pixel that need to display as a visual image.

2.2.1.1 Brightness and Contrast

An image must have the proper *brightness* and *contrast* for easy viewing. Brightness refers to the overall lightness or darkness, i.e., brightness values of different pixels have significance only relative to each other and they are meaningless in absolute terms. So the brightness values of the two images have somehow been normalized so that the effects of the different physical processes have been removed. When the brightness is too high the whitest pixels are saturated, destroying the detail in these areas. The reverse, i.e., setting too low brightness the blackest pixels are saturated. Contrast is the *difference* in brightness between objects or regions. For example, a white rabbit running across a snowy field has *poor* contrast; a black dog against the same white background has *good* contrast. If the contrast is too high black pixels will

be being too black and the white pixel being too white and all of the pixels will be a mid-shade of gray making the objects fed into each other for too low contrast[18].

2.2.1.2. Size of Image

Many image calculations with images are simplified when the size of the image is a power of 2. The number of bits, b, is needed to store an image of size $N \times N$ with 2^m different grey level is :

$$b = N \times N \times m \tag{1.5}$$

So, for a typical 512 x 512 image with 256 gray levels (m = 8) we need 2,097,152 bits or 262,144 8-bit bytes. It is common for 256 gray levels (quantization levels) to be used in image processing, corresponding to a single byte per pixel. There are several reasons for this. First, a single byte is convenient for data management, since this is how computers usually store data. Second, the large number of pixels in an image compensate to a certain degree for a limited number of quantization steps. Third, and most important, a brightness step size of 1/256 (0.39%) is smaller than the eye can perceive. An image presented to a human observer will not be improved by using more than 256 levels[16].

The resolution of an image expresses how much detail we can see in it and clearly depends on both N and m. Keeping m constant and decreasing N results in *checkerboard effect*. Keeping N constant and reducing m results in *false contouring*. However the storage as well as the processing requirement will also increase with the higher resolution.

2.2.1.3. Fundamental steps in Image Processing[17]

Digital image processing encompasses a broad range of hardware, software, and theoretical underpinning. The fundamental steps of image processing are:

- Image acquisition
- Preprocessing
- Segmentation
- Representation and description
- Recognition and interpretation.

• Image acquisition

The first step in the process is *image acquisition* that is to acquire a digital image. To do this an imaging sensor is required which is capable to digitize the signal produced by the sensor. In this case, the object's motion past the line scanner produces a two-dimensional image. If the output of the sensor is not already in digital form, an analog-to-digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application. Camera, Scanner etc. may be used to acquire the image.

• Preprocessing

After a digital image has been obtained, the next step deals with *preprocessing* that image. The key function of preprocessing is to improve the image in ways that increase the chances for success of other process. Preprocessing deals with techniques for enhancing contrast, removing noise and isolating regions.

• Segmentation

The next stage deals with *segmentation* that partitions an input image into its constituent parts or objects. Autonomous segmentation is one of the most difficult

tasks in digital image processing. On one hand, a rugged segmentation procedure brings the process a long way toward successful solution of an image problem. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure.

- **Representation and Description**

The output of the segmentation stage usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. The first decision that must be made is whether the data should be represented as a boundary or as a complete region. Boundary representation is appropriate when the focus is on external shape characteristics, such as corners and inflections. Regional representation is appropriate when the focus is on internal properties, such as texture or skeletal shape.

Description, also called *feature selection*, deals with extracting features that result in some quantitative information of interest or features that are basic for differentiating one class of objects from another.

- **Recognition and Interpretation**

The last stage involves in recognition and interpretation. *Recognition* is the process that assigns a label to an object based on the information provided by its descriptors. *Interpretation* involves assigning meaning to an ensemble of recognized objects.

2.2.1.4 Features of Image Processing[19]

The features of image processing are as follows:

Image modeling

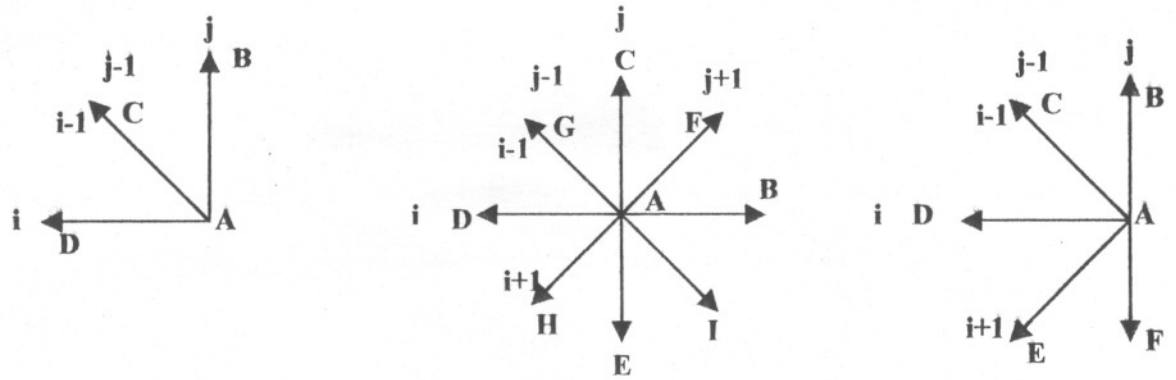
Statistical models describe an image as a member of an ensemble, often characterized by its mean and covariance functions. This permits development of algorithms that

are useful for an entire class or an ensemble of images rather than for a single image. Often the ensemble is assumed to be stationary so that the mean covariance functions can easily be estimated. Stationary models are useful in data compression problems such as transform coding, restoration problems such as Wiener filtering, and in other applications where global properties of the ensemble are sufficient. A more effective use of these models in image processing is to consider them to be spatially varying or piecewise spatially invariant.

To characterize short-term or local properties of the pixels, one alternative is to characterize each pixel by a relationship with its neighborhood pixels. There are three types of stochastic models where an image pixel is characterized in terms of its neighboring pixels.

- Causal
- Noncausal
- Semicausal

If the image were scanned top to bottom and then left to right; the model would be called a *causal model,* which is shown in Figure. 2.1(a). This is because the pixel *A* is characterized by pixels that lies in the "past". In *noncausal* model the neighbors of A lie in the past as well as the "future" in both the directions shown in Figure-2.1(b). Because of the neighbors of *A* are in the past in the *j*-direction and are in the past as well as future in the *i*- direction. Such models are useful in developing algorithms that have different hardware realization. Causal models can realize recursive filters, which require small memory while yielding an infinite impulse response (IIR). On the other hand, noncausal models can be used to design fast trans-form based finite impulse



(a) Causal Model (b) Noncausal Model (c) Semicausal Model

Figure 2.1- Three canonical forms of stochastic model.

response (FIR) filters. In *semicausal* model the neighbors of A are in the past in the j-direction and are in the past as well as future in the I-direction, which is shown in Figure. 2.1(c). Semicausal models can yield two-dimensional algorithms, which are recursive in one dimension and non-recursive in the other.

In global modeling, an image is considered as a composition of several objects. Various objects in the scene are detected and the model gives the rules for defining the relationship among various objects.

Image enhancement

In image enhancement, the goal is to accentuate certain image features for subsequent analysis or for display like contrast and edge enhancement, pseudocoloring, noise filtering, sharpening, and magnifying. Image enhancement is useful in feature extraction, image analysis, and visual information display. The enhancement process itself does not increase the inherent information content in the data. It simply

emphasizes certain specified image characteristics. Enhancement algorithms are generally interactive and application-dependent.

Image restoration

Image restoration refers to removal or minimization of known degradations in an image. It includes deblurring of images degraded by the limitations of a sensor or its environment, noise filtering, and correction of geometric distortion or non-linearity due to sensors. If the image system is linear, the image of an object can be expressed as

$$g(x, y) = \int \int h(x, y; \alpha, \beta) f(\alpha, \beta) d\alpha, d\beta + \eta(x, y) \quad (1.6)$$

where $\eta(x,y)$ is the additive noise function, $f(\alpha, \beta)$ is the object, $g(x,y)$ is the image, and $h(x, y; \alpha, \beta)$ is called the *point spread function* (PSF). A typical image restoration problem is to find an estimate of $f(\alpha, \beta)$ given the PSF, the blurred image, and the statistical properties of the noise process. *Wiener filter* is commonly used for image restoration. Several other image restoration methods such as least squares, constrained least squares and spline interpolation methods can be shown to belong to the class of Wiener filtering algorithm.

Image Analysis

Image analysis is concerned with making quantitative measurements from an image to produce a description of it. Image analysis techniques require extraction of certain features that aid in the identification of the object. Segmentation techniques are used to isolate the desired object from the scene so that measurements can be made on it subsequently.

Image reconstruction from projections

Image reconstruction from projections is a special class of image restoration problems where a two-(or higher) dimensional object is reconstructed from several one-dimensional projections. Each projection is obtained by projecting a parallel X ray (or other penetrating radiation) beam through the object. Reconstruction algorithms derive an image of a thin axial slice of the object, giving an inside view otherwise unobtainable without performing extensive surgery. Such techniques are important in medical imaging, astronomy, radar imaging, geological exploration and nondestructive testing of assemblies.

Image Data Recompression

The amount of data associated with visual information is so large that its storage would require enormous storage capacity. Although the capacities of several storage media are substantial, their access speeds are usually inversely proportional to their capacity. Image data compression techniques are concerned with reduction of the number of bits required to store or transmit images without any appreciable loss of information.

2.2.2. Digital Image Representation[20]

Images can have either digital or analog representation. In the digital representation of gray-level images, the image is presented as a two-dimensional array of number. To display a visual image, the value of each pixel is converted into a gray scale, where 255 represents white and 0 represents black and the intermediate values are shades of gray. In this form values cannot be fractional, e.g., 10.4, it has to be integer and

integers are sufficient in most of the applications. It is common for 256 gray levels to be used in image processing, corresponding to a single byte per pixel.

How a digital image is formed is shown in Fig. 2.2. In this figure an imaginary grid is placed on the image. The size of the openings in the grid determines the size of the pixel. The pixel's gray level is evaluated by the average light intensity at each opening of the grid. The finer the grid size, the higher the resolution and this purpose is to provide a higher-quality image with double the amount of resolution in the new advanced television field. The fundamental requirement of digital image processing is that images be sampled and quantized. The sampling rate (number of pixel per unit area) has to be large enough to preserve the useful information in an image. Image quantization is the analog to digital conversion of a sampled image to a finite number of gray levels. In the analog world, images are usually presented as horizontal raster lines (Fig.2.2).

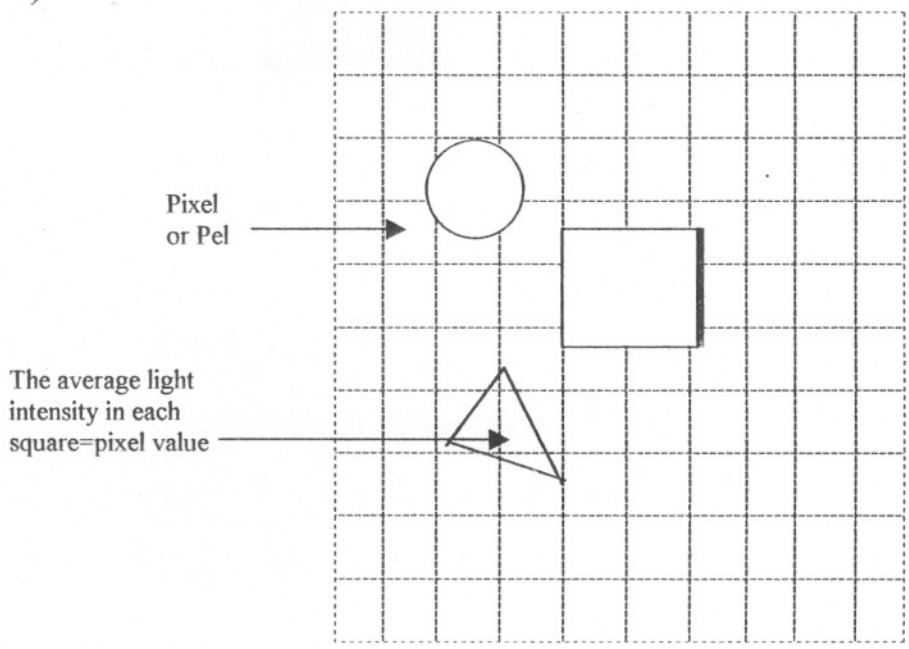


Figure 2.2 Representation of a gray-level digital image.

Each line is basically an analog signal carrying the continuous variations of light intensities along a horizontal line in the original scene. Images on television sets are displayed through raster scanning. Although the term *analog* is used in describing raster-scanned images, the image is analog only along the horizontal direction. It is discrete along the vertical direction, and should be considered as a hybrid signal. In analog images, it is to provide double the amount of lines for better resolution in large-screen television sets.

2.2.3. Image File Format[21]

Image processing involves processing or altering an existing image in a desired manner. The first step is obtaining an image. The programmer needs a simple method of obtaining image data in a standard, usable format, since usable image data is not readily available. When placing images into storage it is important to select an appropriate ***file format.*** The file format of an image will determine the process of storing image data and the additional information that stored with the pixel value. An image file consists of ***a header segment*** and ***a data segment***. The header will contain at least the width and height of that image-since it is impossible to display or process any image without knowledge of its dimensions. The headers of most file formats begin with a ***signature*** or ***magic number***, i.e. a short sequence of bytes designed to identify the file as an image with that specific format. Figure-2.3 shows the basic structure of an image file.

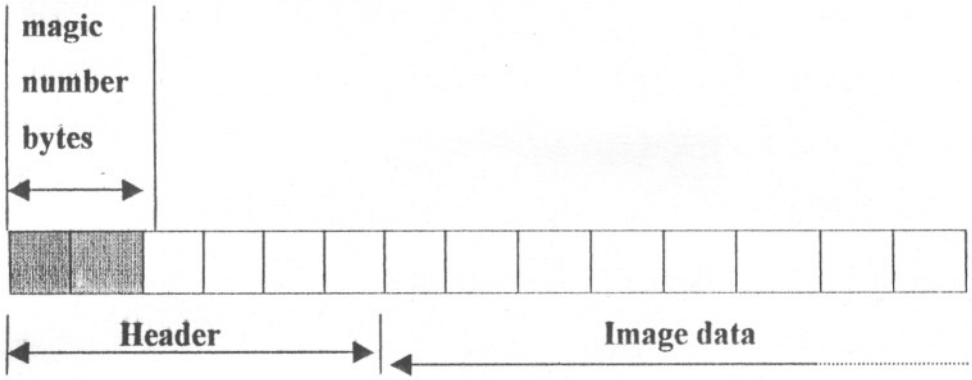


Figure 2.3 Basic structure of an image file

File formats can be grouped roughly into three categories.

Device–specialized formats have been tailored for us with specific pieces of computer hardware. The structure of the image file may be chosen to facilitate rapid display on a particular type of workstation, for example. The disadvantage of this format includes their lack of portability, their inefficiency when used with other hardware, and the tendency of the format to change when hardware is updated.

Software–specialized formats are those designed by a software vendor to be used with a particular program or class of programs. Examples include the PCX and Windows bitmap (BMP) formats commonly found on PCs, or the MacPaint format used on Apple Macintosh computers. The advantages and disadvantages are similar to those for device-specialized formats.

Interchange formats are designed to facilitate the exchange of image data between users, via removable storage media or computer networks. It is essential that they are usable with the widest possible range of hardware and software. Image compression is often a standard feature of interchange formats, since it reduces storage requirements and transmission times.

Examples of common interchange formats are

- GIF - Graphic Interchange format
- PNG – Portable Network Graphics
- JFIF – JPEG File Interchange Format
- TIFF – Tagged Image File format
- PGM – Portable Grey Map
- FITS – Flexible Image Transport System

2.3 Applications[19]

Digital image processing has a broad spectrum of applications, such as remote sensing via satellites and other spacecrafts, image transmission and storage for business applications, medical processing, radar, sonar, and acoustic image processing, robotics and automated inspection of industrial parts.

Images acquired by satellites are useful in tracking of earth resources; geographical mapping; prediction of agricultural crops, urban growth, and weather; flood and fire control; and many other environmental applications. Space image applications include recognition and analysis of objects contained in images obtained from space-probe mission.

In medical application images one is concerned with processing of chest X rays, cineangiograms, projection images of transaxial tomography, and other medical images that occur in radiology, nuclear magnetic resonance (NMR), and ultrasonic scanning. These images may be used for patient screening and monitoring or for detection of tumors or other disease in-patients.

Image transmission storage applications occur in broadcast television, teleconferencing, transmission of facsimile images for office automation communication over computer networks, closed circuit television based security monitoring systems, and in military communication systems.

Radar and sonar images are used for detection and recognition of various types of targets or in guidance and maneuvering of aircraft or missile systems.

2.4 Image Filtering[22]

All images and pictures contain spatial frequencies. The gray level in the image varies in space (not time), i.e. it goes up and down. The gray level changes many times in the space of the image. The main objective of image filtering is to remove the noise from the actual data. In the image processing and pattern recognition systems, images are entered into computer through scanner, tablet, digitizer etc. Normally these input devices do not give the exact data what a programmer wants, rather they introduce some unwanted extra data that are noise with respect to the actual data wanted.

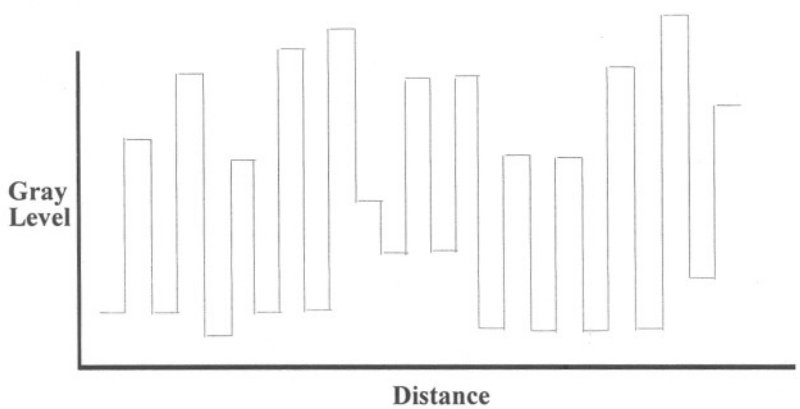


Fig-2.4: Side view of an image with high spatial frequencies

Filtering is used for many other purposes such as Smoothing, Sharpening, Embossing, Diffusing, Solarizing, Edge detection etc. Filtering can be expressed generally as the point-by-point multiplication of the spectrum by a *filter transfer function*. We can write

$$G(u,v) = F(u,v)H(u,v)$$

where F is the spectrum of the image, H is the filter transfer function and G is the filtered spectrum. An inverse Fourier transform of G must be computed in order to see the results of filtering as an image.

2.4.1 Edge Detection[20][22]

Detecting edges is a basic operation in image processing. The image items in an image hold much of the information in the image. The edges indicate where items are, their size, shape and something about their texture. Edges characterize object boundaries and are therefore useful for segmentation, registration and identification of objects in scenes. Edge points can be thought of as pixel locations of abrupt gray-level changes. The number of masks is used for edge detection.

The extraction of edges or contours from a two dimensional array of pixels (a gray-scale image) is a critical step in many image processing techniques. A variety of computations are available which determine the magnitude of contrast changes and their orientation. Extensive literature exists documenting the available operators and the post-processing methods to modify their output. A trade-off exists, though, between efficiency and quality of the edge detection. Fast and simple edge detection can be performed by filters such as the popular Sobel operator which requires the mere convolution of a small kernel (3 ×3 pixels) over the image. Alternatively, more computationally intensive contour detection techniques are available such as the

Deriche or Canny[4] method. These detectors require that a set of parameters be varied to detect the desired scale and curvature of edges in the image. It is necessary to compare the simple Sobel detector and the complex Deriche-type detectors before selecting the edge detection scheme of preference.

The simplest detectors perform minimal noise smoothing and fairly crude localization. They are based on the estimation of **gray level gradient** at a pixel. The gradient can be approximated in the x and y directions by

$$gx(x,y) \approx f(x+1,y) - f(x-1,y),$$
$$gy(x,y) \approx f(x,y+1) - f(x,y-1).$$

The gradient calculation is expressed as a pair of convolution operations

$$gx(x,y) = hx * f(x,y)$$
$$gy(x,y) = hy * f(x,y)$$

The Sobel Operator[20]

The 3 x 3 Sobel operator acts locally on the image and only detects edges at small scales. If an object with a jagged boundary is present, the Sobel operator will find the edges at each spike and twist of the perimeter. The operator is sensitive to high frequency noise in the image and will generate only local edge data instead of recovering the global structure of a boundary. Furthermore, smooth transitions in contrast that occur over too large a spatial scale to fit in the 3 x 3 window of the Sobel operator will not be detected. The Sobel mask operators are as follows

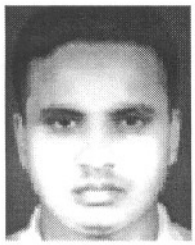
W_1

1	0	-1
2	0	-2
1	0	-1

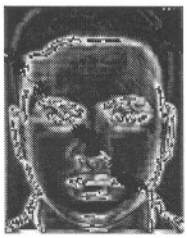
W_2

-1	-2	-1
0	0	0
1	2	1

The edge detection operation was applied to the faces using Sobel operator. Fig-2.5 shows the resulting face after edge detection.



(a) The original Image

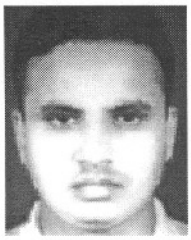


(b) Image after edge detection

Figure - 2.5 Edge Detection by Sobel Operators

2.5 Thresholding[21]

The technique of ***thresholding*** is used in a variety of different image processing operations. Thresholding transforms a dataset containing values that vary over some range into a new dataset containing just two values. If a binary (black and white) version of the gray level image is desired, it can be obtained by thresholding. All gray level values whose magnitudes are less than a threshold will be set to 0 and those greater than a threshold will be set to the maximum edge value (255).



Original face

Thresholded face

Figure - 2.6: The Effect of Image Thresholding

Thus, the maximum image contrast is compressed to 1 bit. So the most common form of image thresholding makes use of pixel gray level. Gray level applies to every pixel the rule

$$g(x, y) = \begin{cases} 0, & f(x, y) < T \\ 1, & f(x,y) > T \end{cases}$$

where T is the threshold. This equation specifies 0 and 1 as output values, giving a true binary image, but it is common to use 0 and 255 so that pixels appear black or white if the output image is displayed. Figure-2.6 shows the thresholded face after applying thresholding technique on the original face image.

2.6 Face Detection and Localization[4]

The detection of faces and facial features from an arbitrary uncontrived image is a critical precursor to recognition. A robust scheme is needed to detect the face as well as determine its precise placement to extract the relevant data from an input image. This is necessary to properly prepare the image's 2D intensity description of the face for input to a recognition system. This detection scheme must operate flexibly and

reliably regardless of lighting conditions, background clutter in the image, multiple faces in the image, as well as variations in face position, scale, pose and expression. The geometrical information about each face in the image will be used to apply geometrical transformations that will map the data in the image into an invariant form. By isolating each face, transforming it into a standard frontal mug shot pose and correcting lighting effects, the variance in its intensity image description to the true physical shape and texture of the face itself is to be limited.

The human face is a highly correlated object due to the lack of variation in skin complexion. Even though facial features (i.e. mouths and eyes) differ in color from the skin tone, the hairless skin regions dominate facial surface area allowing it stand out against most backgrounds. Thus we expect a boundary around the face to be present. The foreshortening of the 2D-face structure under most lighting conditions also accentuates the contour of the face. The set of input images will give some of the variations in the intensity image that detection must be capable of overcoming to properly localize the face. These variations need appropriate compensation to isolate only the relevant data necessary for recognition. Furthermore, note that these variations can occur in any combination and are not mutually exclusive.

2.6.1 Face Detection and Tracking Strategies[4]

Over 30 years, especially recently, face images have received increasing attention in the academic communicates in pattern recognition, computer vision, image processing and computer graphics. In this section the state of the art of face detection and tracking techniques are shortly reviewed. In this area, the researchers have used

several characteristics, such as motion, blink, color, shape, and facial features of faces in images. Different strategies are shortly described below:

Motion Detection

The motion of a face is effective information for separating the face region from the background in complex scenes. The motion of objects in an image sequence is not given directly but must be computed from image intensity. In face tracking systems, the following methods are adopted to detect or measure the 2D motion.

- Using intensity difference among a series of image frames.
- Optical flow
- Block matching
- Pel – recursive techniques
- Frequency domain techniques.

Blink Detection

Blink is a spontaneous response to keep our eyes moist. Blinks happen frequently and fast, thus most of us are not aware when we blink. Compared with head movement, blink is also a kind of motion but happens in a small area and is unique to faces. Hence blink detection is a dependable method of detecting the presence of face.

Color Segmentation

For all human beings of different faces, the skin colors under same illumination can be divided into 36 kinds[15] , from an unsaturated light color (corresponds to Caucasian), to a saturated dark brown color (correspond to African). To utilize skin color as a visual cue, an appropriate color space must be chosen so that the distribution of skin colors is as compact as possible.

Shape Analysis

Viewed from any direction, people's head outlines are roughly elliptical (with some exceptions caused by hairstyle). Many researchers have taken advantage from this feature. Their shape analysis algorithms are usually associated with color or motion detection results, or both.

Facial Feature Tracking

Facial features such as eyes, nose and mouth are the crucial visual cues in facial image processing. Facial features are distinguished by their gray level, color edge strength, or shape, different from the surrounding area. By detecting the facial features, the location, motion, pose and size of a human face can be estimated.

2.6.2 Typical Face Detection and Tracking System

Until now various innovative methods to locate and track faces have been proposed. Basically they used the visual cues. Some typical and systems are introduced below.

Recently, five commercial face recognition products were introduced in *Advanced Imaging*[23][24]. Three of them can only perform face recognition on still images, while the others can do this from live video in addition to still images.

FaceIT : A face recognition software engine

FaceIT won a US Government face recognition competition in 1996 and some other awards. It is capable of performing real-time face tracking or detection, and recognition on moving or stationary people. The whole system consists of four modules: head module, motion module, alignment module and identification module. The input image is first sent to the head module, which measures the statistical properties of elliptical shapes and segments them from background. If the input is an image sequence, then the motion module assists this task. At every pixel, a fast

matching algorithm estimates the probability that a head appears there. Then the alignment module checks the most probable positions in more detail. If the presence of a face is confirmed, then this face is transformed into a canonical image. Facial features and other characteristics are extracted and sent to the identification module for face recognition.

TrueFace by Miros

In March 1997, TrueFace became the first face recognition product installed in ATM machine. Claimed by the vendor, since that time, True Face has become the most installed face recognition product worldwide. It is mainly used for allowing or denying access to ATMs or sensitive locations. It is reported by the vendor that TrueFace rejects authorised users only 0.1% of the time (1 per 1 thousand) and catch frauds greater than 99.9% of the time. It can also distinguish between a live face and a picture of a valid user held in front of a camera by requiring either 2 cameras or multiple views with 1 camera. The face recognition time is within 1 second.

Neural Network –Based Face Detection

Rowley, Baluja and Kanade[25] in Carnegie Mellon University demonstrated a neural network-based based face detection system. A retinally connected neural network scans through an input image with a small window (20 x 20 pixels). The pixels within that window are a test pattern. Every test patterned is preprocessed and image pyramids are adopted to cope with scale variations. Nearly 1050 face examples and 1000 computer created nonface images were used in the training stage. They used a *bootstrap* algorithm for training, which adds false positives into the training set as training progresses. This reduces the number of training nonface images needed.

Example-Based learning for View-Based Human Face Detection

Kah-Kay Sung and Tomaso Pggio proposed an example-based learning approach for face detection. Their system scans through an input image and receives a small window (19 x 19 pixels) as the test pattern for examination. The preprocessing steps (masking, illumination gradient correction, and histogram equalization) and an image pyramid technique also applied. They use a database of 4,150 normalized face patterns and 6,189 normalized face-like "nonface" patterns to synthesis six "face" pattern clusters and six "nonface" pattern clusters. Normalized Manalanobis distance between the test pattern and cluster prototype, in a low dimensional subspace created by the cluster's 75 most significant Eigen vectors, and the second measurement is the normalized Euclidean distance between the test pattern and its projection in the 75-dimensional subspace. Finally, these 12 distance-measurement pairs are fed into a multiplayer perceptron net classifier that gives the ultimate decision.

Face Localization[4]

The human face is a highly correlated object due to the lack of variation in skin complexion. Even though facial features (i.e. mouths and eyes) differ in color from the skin tone, the hairless skin regions dominate facial surface area allowing it stand out against most backgrounds. Thus we expect a boundary around the face to be present. The foreshortening of the 3D face structure under most lighting conditions also accentuates the contour of the face. The face is to be considered as a blob or a region with a certain amount of contour enclosure. Furthermore, the scalp and the hair also usually trigger edge contours that extend the face blob. Consequently, both the face and head structures can be expected to behave as blobs or symmetric enclosures about a center point.

Face Blob Localization

Having formally described a blob-detector, the symmetry transform, it is elected to use it to initially detect all blobs in the image. Some detected blobs will correspond to non-facial structures however all faces and heads should trigger the interest map. Recall that the transform operates on edge maps so it should locate blobs despite the blob's intensity values or shading and illumination. Furthermore, the blob detector is rotation invariant.

An arbitrary image of a natural uncontrived scene containing people is considered. The edge map pyramid to apply the symmetry transform at various scales is used. This allows us to detect blobs of arbitrary size in the image. The blob detector uses only the 6 annular sampling regions described in Table-2.1. Then it can afford to limit the number of annular sampling regions to six at this stage since the sub averaging involved in the pyramid obviates the need for more scale invariance in the operator. The general symmetry transform to each of the edge maps is applied and marked the centers of the detected blobs on the intensity pyramid. The general transform (not the dark or bright symmetry transform) was utilized since heads and faces do not consistently appear either brighter or darker than the background of a scene. This multi-scale interest detection operation provides us with the blob detection pyramid displayed in Figure 2.7.

Table 2.1: The annular sampling regions to be used for face detection

Annular Region Number	Range of Radii
1	$0.75 < r < 2.25$ pixels
2	$1.75 < r < 3.25$ pixels
3	$2.75 < r < 4.25$ pixels
4	$3.75 < r < 5.25$ pixels
5	$4.75 < r < 7.25$ pixels
6	$6.75 < r < 9.25$ pixels

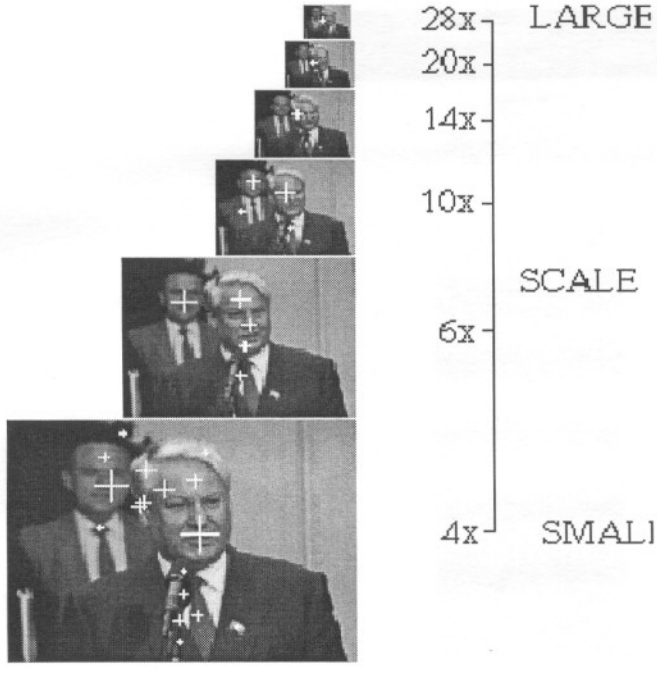


Figure 2.7: The multi-scalar interest map pyramid

The output of the interest map is thresholded so that only attentional peaks exhibiting a certain minimal level of interest will appear in the output. The threshold on the interest map is very tolerant and allows many extremely weak blobs to register. Thus, the precise selection of an interest map threshold is not critical. Furthermore, it considers the five (5) strongest peaks of interest or the five most significant blobs for each scale in the multi-scalar analysis. This is to prevent the system from spending too much time at each scale investigating blobs. It is expected the face to be somewhat dominant in the image so that it will be one of the strongest five blobs in the image (at the scale it resides in). If many faces or other blobs in the image at the

same scale are expected, this value can be increased beyond 5. This would be advantageous, for example, when analyzing group photos. Both a threshold on interest value and the limitation on the number of peaks are required since we do not wish to ever process more than 5 blobs per scale for efficiency and we require the blobs to exhibit a minimal level of significance to warrant any investigation whatsoever. Furthermore, it will be stopped applying the interest operator for scales smaller than 4x. The interest operator is limited in size to r=9 pixels and consequently, the blobs detected at scales lower than 4x would be too small and would have insufficient resolution for subsequent facial feature localization and recognition. For example, the blobs detected at scale 3x would be less than 54 x 54 pixel objects and the representation of a face at such a resolution would prevent accurate facial feature detection.

The peaks in Figure 2.7 are shown before thresholding the interest response, threshold the number of blobs per scale (maximum of 5) and before the scale of the search is limited. Once these three limits are introduced, the number of peaks generated by the face blob detection stage will drop as shown on the right hand side of Figure 2.7. Only a total of 5 peaks are valid after this filtering (as seen by the 5 square grids that remain for processing by the next stages).

There is some redundancy as some blobs are detected more than once at adjacent scales. This is due to overlap in scale-space of our symmetry transform operator. However, this redundancy or multiple-hit phenomenon is not problematic since it will use later stages to select only one 'hit' or one face out of several redundant blob responses. Additionally, the detection of non-facial blobs is not problematic at this stage. Since each blob is to undergo further processing to determine if it is truly a

face, false alarms during blob detection can be allowed. Finally, lack of accuracy in blob detector is also acceptable at this stage since it will refine the localization of faces in subsequent stages. What is most dangerous at this early stage of the algorithm is a total miss of a face or head blob in the image. Fortunately, a clear miss of the face in our multi-scalar blob detection is extremely unlikely since heads and faces have consistently strong responses in the perceptual interest map.

Chapter-3

Fundamentals of Artificial Neural Network

FUNDAMENTALS OF ARTIFICIAL NEURAL NETWORK

3.1 Introduction

Neural network can recognize, classify, convert and learn patterns with the ultimate goal to simulate human vision. *Human Face Recognition* refers to the categorization of input facial image into identifiable classes by recognizing significant features or attributes of the facial image. In traditional face recognition system, a facial image is an n-dimensional feature vector. A feature vector is a vector of feature values arranged by a certain order. The facial images are thus formulated as multidimensional curve fitting using methods of approximation. However, in neural network approach, a facial image is represented by a number of nodes along with their activation levels. It is not necessary to know the formal mathematical model of classification but suitable network architecture and a sufficient training set are necessary to obtain a solution through experimentation. As such, neural network computation offers techniques that are in the middle ground between the traditional engineering and the artificial intelligence approach.

3.2 History of Artificial Neural Network

ANN research has a rich history of approximately 40 years. It seems to be presently at an all-time peak. Biological systems implement pattern recognition computations via interconnections of physical cells called neurons. This provides motivation to consider emulation of this computational mechanism. Researchers

from diverse areas such as neuroscience, mathematics, psychology, engineering and computer science are attempting to relate underlying models for image recognition, the computation that is desired, the potential parallelism that emerges and the operation of biological neural systems. This system is known by several names including (Artificial) Neural Network, Connectionist Modeling, Neuromorphic Modeling, and Parallel Distributed Processing (PDP). The basic computing element of the human information processing system is relatively slow but the overall processing operation is achieved in a few hundred milliseconds, suggests that *the basis of the biological computation is a small number of serial steps, each massively parallel*. In parallel architecture, each of the processing elements is locally connected and relatively simple. Thus connectionist or neural computing is not new or revolutionary. It is rather, evolutionary, with roots in a number of well-understood concepts, including biological pattern recognition, perceptron and linear machines, adaptive networks and fine-grained parallel computing paradigms.

In 1946, Presper Eckert, John Mauchly and Herman Goldstine worked on the design and development of the general purpose electronic digital computer. They were aided in this effort by Von Neumann. In 1948, W. Ross Ashby publishes a paper entitled "Design for a brain" in which he discussed emulation of the brain's adaptive behaviour[26]. There are many people who had worked with the Neural network from 1949 to 1992 in various ways.

The symbolic approach, which has long dominated the field of AI, was recently challenged by the neural network approach. Several new learning laws have also been developed the prominent among them being the reinforcement learning with criticism. Several architectures were developed to address specific issues in pattern recognition. Some of these architectures are: adaptive resonance theory,

neuro-recognition being used to enhance the capability of the neural networks to deal with real world problems such as in speech, image processing, natural language processing and decision making.

Recently there has been a huge development in neural network for Human Face Recognition. In 1995, Tony S. Jebara proposed a hierarchical detection system capable of searching images for human faces efficiently and with invariance to position, deformation, illumination, scale and 3D pose using neural model[4]. In 1996, Steve Lawrence, C Lee Giles, Ah Chung Tsoi, Andrew D. Back present a hybride neural network solution for Face Recognition[27]. In 1999, Gyu-tae Park, Zeungnam Bein published a paper named "Neural network-based fuzzy observer with application to facial analysis". They performed an experiment, which demonstrated that the facial wrinkles could be indirectly estimated by the proposed fuzzy observer[28].

3.3 Biological Neuron and its Model

Artificial neural networks (ANNs), also called *parallel distributed processing systems* (PDPs) and *connectionist systems*, are intended for modeling the organizational principles of the central nervous system. The biologically inspired computing capabilities of the ANN will allow the cognitive and sensory tasks to be performed more easily and more satisfactory than with conventional serial processors. Basically three entities characterize an ANN[29]:

- The network topology, or interconnection of neural 'units'.
- The characteristics of individual units or artificial neurons; and
- The strategy for pattern learning or training.

Because of the limitations of serial computers, much has been devoted to parallel processing architectures. In the extreme case of fine-grained parallel processing

architecture, the function of a single processor is at a level comparable to that of a neuron. Neural networks are massively parallel analog processors, which are intended for modeling the organizational principles of the central nervous system of the brain. The human brain is a massive parallel personal computer based on organic threshold logic devices. These logic devices are known as *neurons*. A single neuron is shown in Figure-3.1. The cell body has an output line called the *axon*. The input lines are called *dendrites*. There are about 1000 dendrite connections to an average neuron in the human brain. The neurons are interconnected by a *synapse*. A neuron can have both excitatory and inhibitory connections. An *excitatory* connection tells a neuron to fire and an *inhibitory* connection tells a neuron not to fire.

The input signals are summed in the neuron. At a certain threshold level the neuron will fire and below this level not fire.

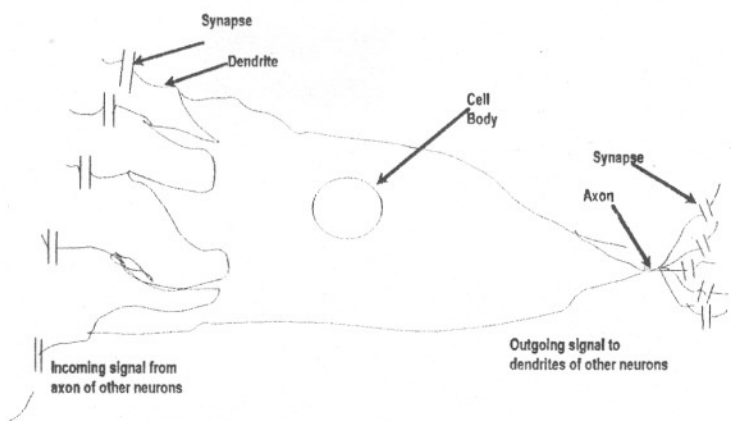


Figure-3.1: Schematic diagram of a biological neuron.

Neural network models, even neuro-biological ones, assume many simplifications over biological neural networks. Such simplifications are necessary to understand the intended properties and to attempt any mathematical analysis. The neuron is the basic processor in neural networks. Each neuron has one output, which is

generally related to the state of the neuron, its activation, which may far out to several other neurons. Each neuron receives several inputs over the connections, called synapses. The inputs are the activation of the incoming neurons multiplied by the weights of the synapses. The activation of the neuron is computed by applying a threshold function to this product. An abstract model of neuron is shown in Figure-3.2.

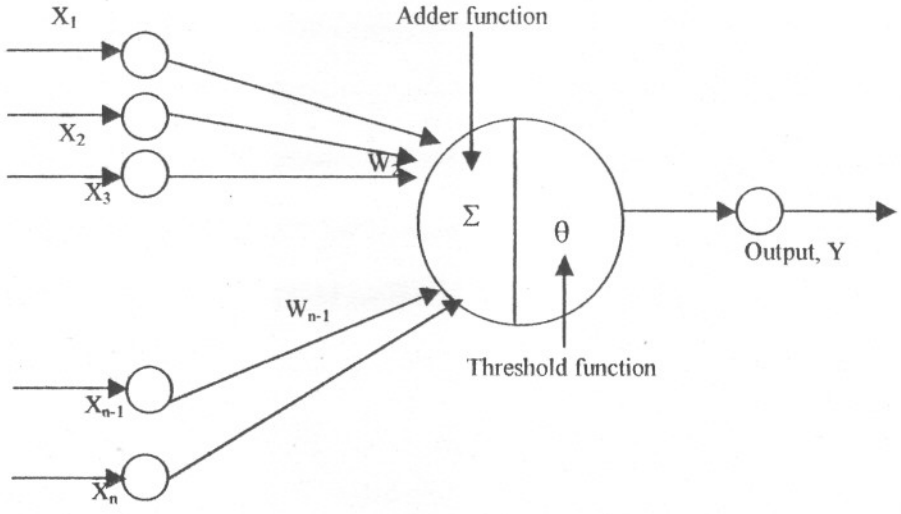


Figure-3.2: The McCulloch-Pitts neuron model

The threshold function is generally some form of nonlinear function. One simple nonlinear function that is appropriate for discrete neural nets is the step function. One variant of the step function is:

$$f(x): \begin{cases} 1 & \text{if } x > 0 \\ f(x) & \text{if } x = 0, \text{ where } f(x) \text{ refers to the previous value of } f(x) \\ -1 & \text{if } x < 0 \end{cases}$$

where x is the summation over all the incoming neuron's activation and the synaptic weight of the connection:

$$X = \sum_{i=0}^{n} XiWi$$

where n is the number of incoming neurons, X=[X_1, X2, X3 ..... Xn] is the vector of incoming neurons, and W=[W_1, W2, W3 ..... Wn] is the vector of synaptic weights connecting the incoming neurons to neurons we are examining[30].

3.4 Concepts and Terminologies[31]

To some extent the neural network system is a *nonalgorithmic, black box* strategy, which is trainable. Actually it is to 'train' the neural black box to correct response or output for each of the training samples. After training the internal (neural) structure of the artificial implementation will *self-organize* to enable extrapolation when faced with new, yet similar, patterns on the basis of 'experience' with the training set. Emulation of biological system computational structures yields superior computational paradigms for certain classes of problems. Among these problems are the following: the class of NP-hard problems, which includes labeling problems, sheduling problems, search problems and other constraint satisfaction problems; the class of pattern/object recognition problems, notably in vision and speech understanding; and the class of problems dealing with flawed, missing, contradicting, fuzzy or probabilistic data. These problems are characterized by these qualities:

- A high-dimensional problem space,
- Complex interactions between problem variables and
- A solution that may be empty, contain a unique solution or a number of solutions.

The myriad of potential neural network applications for pattern recognition includes

- Feature extraction from complex data sets (e.g., images and speech);
- Character recognition and image processing applications and
- Direct and parallel implementation of matching and search algorithms.

A neural network model is made up of the constructs defined in the following paragraphs. The neural network connections are significantly fewer than the connections in the human brain. The basic elements of neural network are described below:

Cells:

A cell (or unit) is an autonomous ***processing element*** **(PE)** that models a neuron. The cell can be thought of as a very simple computer. There is no Supervisor cell; "all cells are created equal". The purpose of each cell is to receive information from other cells, perform relatively simple processing of the combined information and send the results on to one or more other cells. In illustrations of neural networks, circles or squares usually indicate cells. Sometimes these cells are denoted as u_1, u_2... ,u_N, where N is the total number of cells in the network.

Layers:

A layer is a connection of cells that can be thought of as performing some type of the common function. These cells are usually ordered by placing numbers or letters by each cell. It is generally assumed that no cell is connected to another cell in the same layer. All neural nets have an **input layer** and **output layer** to interface with the external environment. Each input layer and each output layer has at least one cell. Any cell that is not in an input layer or an output layer is said to be in a **hidden layer,** because neither its input nor its output can be observed from outside. Sometimes the cells in a hidden layer are called feature detectors

because they respond to particular features in the previous layer. Though not all neural networks have to be thought of as layered (any cell can connect to any other), we emphasize layers here because of their extensive use.

When layers are used, a simple notation may be employed to indicate the structure of the network. This notation indicates the number of cells in each layer, from the input layer through to the output layer. For example, 64-5-7 means that there are 64 cells in the input layer, 5 cells in the hidden layer, and 7 cells in the output layer.

Arcs:

An arc (connection) can be a one-way or a two-way communication link between two cells. A feed forward network is one in which the information flows from the input cells through any hidden layer to the output layer cells without any paths whereby a cell in a lower-numbered layer receives input from a cell in a higher-numbered layer. A feedback network, by contrast, also permits communication "backward".

Weights:

A weight w_{ij} is a real number that indicates the influence that cell u_j has on cell u_j (the subscript- cell order). For example positive weights indicate reinforcement, negative weights indicate inhibition, and a weight of zero (or the absence of a weight) indicates that no direct influence or connection exists. The weights are often combined into a matrix W. These weights may be initialized as zero, initialized as given and predefined values, or initialized as random numbers. Weights may be used to modify the input from any cell. However, the cells in the input layer have no weights, i.e., the external inputs are not modified before going input layer.

3.5 The Single Layer Perceptron[20]

The single layer perceptron is feed forward network and is capable of linearly separating the input vectors into pattern classes. The single layer perceptron consists of an input and output layer. The activation function employed is a hard limiting function. An output unit will assume the value 1 if the sum of its weighted inputs is greater than its threshold. In terms of classification, an object will be classified by unit *j* into class A if

$$\sum W_{ji} X_i > \theta_j$$

where W_{ji} is the weight from unit i to j, Xi is the input from unit I, and θ_j is the threshold on unit j. Otherwise, the object will be classified as class B. If there are n inputs the equation will be

$$\sum_{i=1}^{n} WjiXi = \theta j$$

3.6 The Backpropagation Network[32]

The capabilities of single-layer perceptron are limited to linear decision boundaries and simple logic functions. The simple XOR problem does not have a solution with a single-layer perceptron. Now the question is how we are to overcome the problem of being unable to solve linearly inseparable problems with our single layer perceptron. An initial approach would be to use more than one perceptron, each set up to identify small, linearly separable sections of the inputs, then combining their outputs into another perceptron, which would produce a final indication of the class to which the input belongs. But this arrangement of perceptrons in layers will be unable to learn. Each neuron in the first layer takes the weighted sum of its inputs, thresholds it, and outputs either a one or a zero and the neurons in the second layer takes this outputs as their inputs, they do not know

which of the real inputs were on or not. Two-state neuron, being "on" or "off" gives us no indication of the scale by which we need to adjust the weights, and so

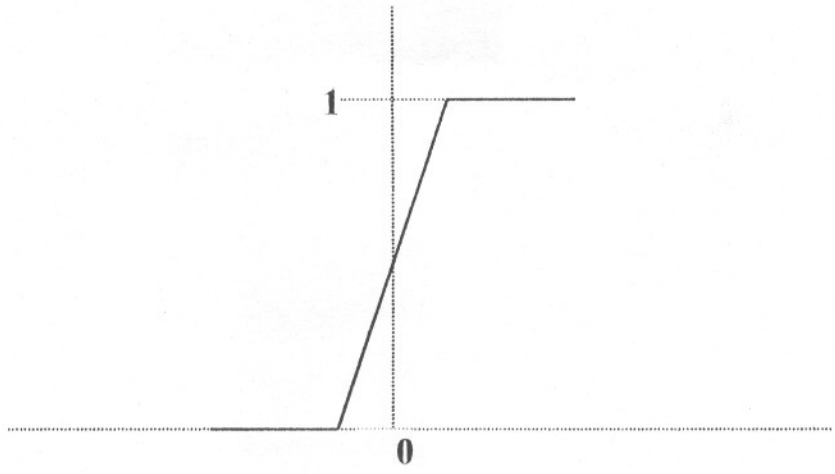


Figure-3.3 Linear threshold

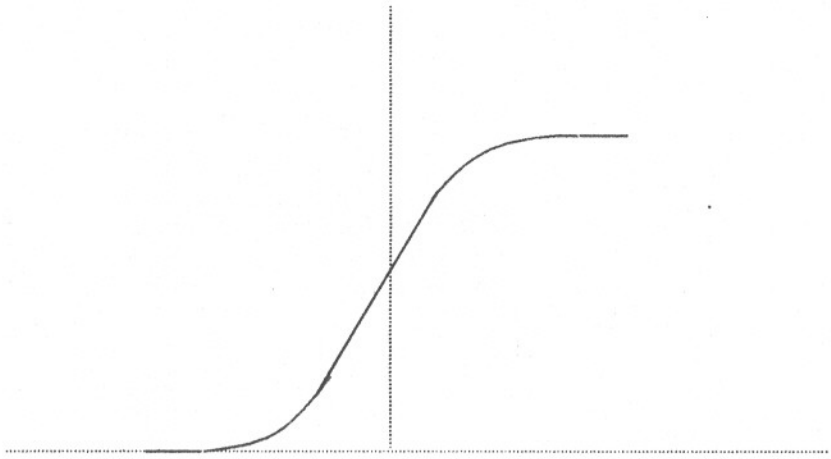


Figure-3.4 Sigmoidal Threshold

we cannot make a reasonable adjustment. The way to solve this problem is to use sigmoidal threshold instead of linear threshold. These two possible thresholding

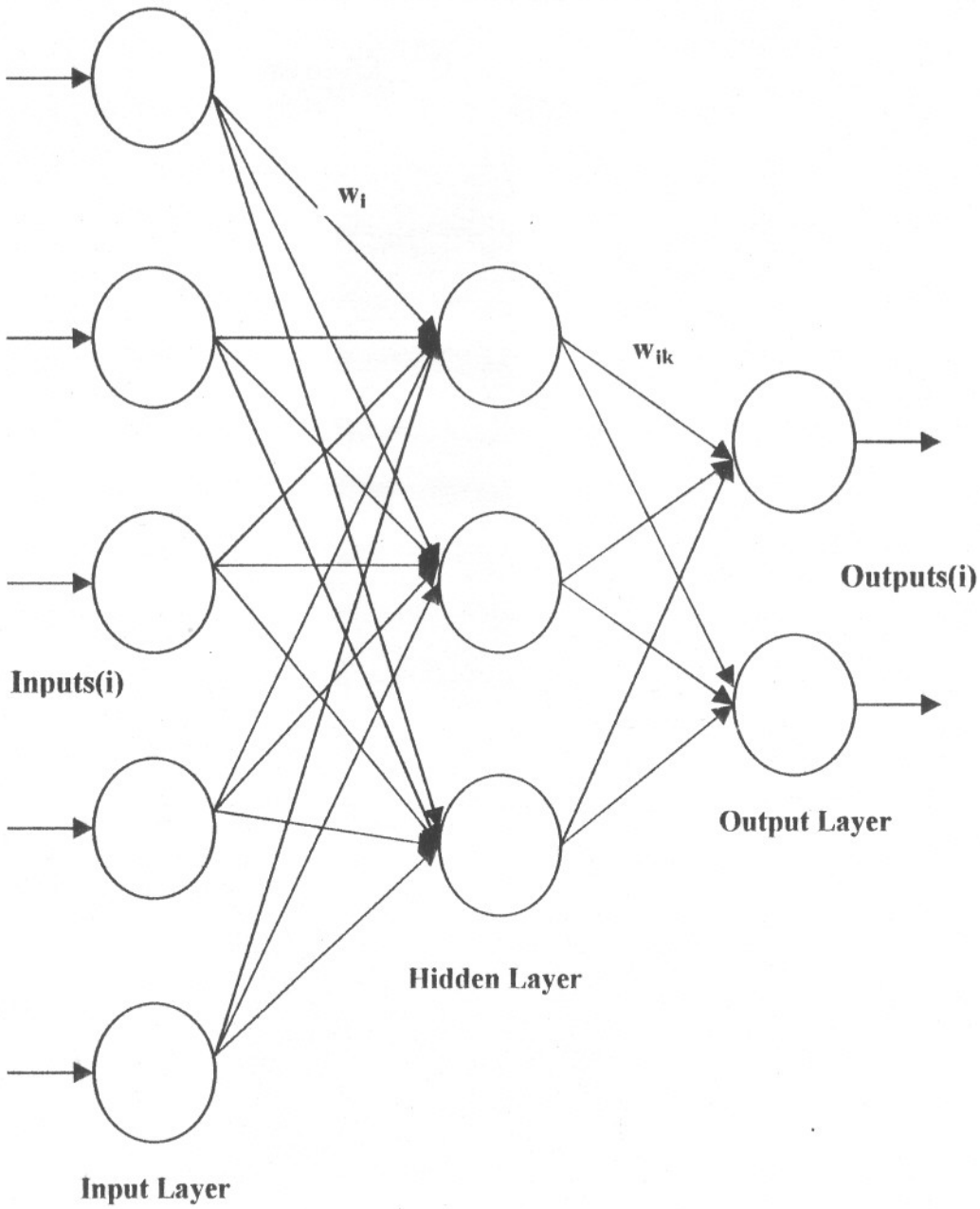


Figure-3.5. The Basic Model of Backpropagation Network

functions are shown in Figure-3.3 and 3.4. This new model is termed as multilayer perceptron or the backpropagation network. Our new model has three layers; an input layer, an output layer and a layer in between called the hidden layer. The basic structure is shown in Figure-3.5.

3.7 Learning by Backpropagation of Errors[32]

The major learning algorithm for neural networks is the back-propagation of errors discussed by Rumelhart[33], which is a generalized delta rule developed by Widrow and Hoff [34,35] and by LeCun [36]. The method seeks to minimize the error in the output of the network, as compared to a target or desired response. For a network having mutiple outputs, the rms error is given by:

$$E = \left[\sum_j (o_j - t_j)^2 \right]^{1/2}$$

where t_j and o_j are the target and actual output values for the j^{th} component of the vectors. The goal of the back-propagation learning procedure is to minimize this error. If the network is time invariant, its output will depend only on its inputs, i_i and current value of the weight matrices, w_{ij}. For a given input vector, therefore, the error is determined by the values of the weighting coefficients that connect the network. The approach used in the adaptive procedure is to modify these connections by an amount proportional to the gradient of the error in weight space:

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}}$$

This procedure generally results in reductions in the average error as the weight matrices in the network evolve. The changes in the weights after each trial are proportional to the error itself. This naturally leads to a system that settles out to a stable weight configuration, as the errors become small.

Consider a single layer within a layered, feed forward network. Its input vector, o_i, might itself be an output from a preceding layer and its output vector o_j might, in turn, provide input to a subsequent layer. The neurons have an activation function f_j and are coupled to the input vector by a weight matrix w_{ij}. The net input to each neuron is given by :

$$net_j = \sum_i o_i w_{ij}$$

and the output vector is given by

$$o_j = f_j(net_j)$$

By generalized delta rule the weight changes are:

$$\Delta w_{ij} = \eta o_i \delta_j$$

In this relationship, η is the learning rate. If the layer in question is an output layer, then δ_j is given by:

$$\delta_j = \left(t_j - o_j\right) f_j'\left(net_j\right)$$

Where tj is the target, or desired output vector and f_j' denots differentiation of the neuron's activation function with respect to the input signal. If the layer is hidden inside the network, it is not immediately apparent what its target response should be. In this case, δ_j is computed iteratively using:

$$\delta_j = f_j'(net_j)\sum_k \delta_k w_{jk}$$

where δ_k and w_{jk} refer to the layer immediately after the one in question.

3.8 Backpropagation learning algorithm

1. Initialize weights and thresholds

   Set all weights and thresholds to small random values.

2. Present input and desired output

   Present input as x_i and target output as t_i.

3. Calculate actual output

Using sigmoid function of the from $f(x) = 1/1 + \exp(-\lambda x_i)$,[37]each node calculates

$$y_j = \frac{1}{1 + \exp\left[-\lambda\left(\sum_{i=1}^{n} w_{ij} y_i^1 - \theta\right)\right]}$$

Where y_i is the output of the node in consideration[26],

y_i^1 is the output of the nodes of the previous layer - x_i in case of first layer,

w_{ij} is the weight for i-j connection

θ_j is the thresholds for the node in consideration

λ determines the slop of the function at point and is called the activation gain.

In our program, two values are used for it, one is *spr.* and other is *spread.*

4. Adapt weights and thresholds

Start from the output layer and work backwards. The weight correction for the hidden-to-output weight matrix elements

$\Delta w_{jk} = \eta_2 (t_k - y_k)(1 - y_k) y_k$ [29],[38]

$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}$

And the weight correction for the input-to-hidden weight matrix elements

$\Delta w_{ij} = \lambda \eta_1 (1 - y_k) y_k \sum_k (t_k - y_k) w_{jk}$ [29],[38]

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$

Similarly, correction for thresholds for the hidden-to-output

$\Delta th_j = \eta_1 (t_k - y_k)(1 - y_k) y_k$

$th_k(t+1) = th_k(t) + \Delta th_j$

And correction for thresholds for the input-to-hidden

$\Delta th_j = \lambda \eta_1 (1 - y_j) y_j \sum_k (t_k - y_k) w_{jk}$

$th_j(t+1) = th_j(t) + \Delta th_j$

Where η_1 and η_2 are small proportionality constant known as learning rate for input to hidden layer and hidden to output layer respectively. The value of λ is *spread* as before. t_k denotes target output.

3.9 Discussion

In this chapter Artificial Neural Networks have been discussed. This research includes the use of Backpropagation Neural Network in human face recognition. So the chapter discusses the networks with special attention to the Error Backpropagation concept. The discussion was mainly concerned with analysis and formulization to help the final stage where the network will be implemented by a computer program. The practical consideration and detailed algorithms are developed in chapter-5, where these analysis and formulas have been implemented to design a complete system for human face recognition.

Chapter-4

Feature Extraction and Recognition Methods

FEATURE EXTRACTION AND RECOGNITION METHODS

4.1 Introduction

Feature extraction is the basic part of all effective recognition systems. So it plays an important role in Human Face Recognition system. The feature extraction technique is used for the following purposes:

- There is a considerable reduction of data for further processing, i.e., a minimum number of features are retained.
- The recognition done in the classification stage with the help of the features will serve required accuracy.

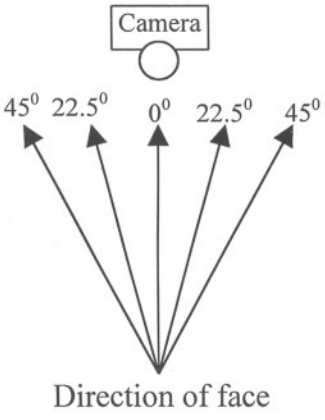
As discussed in section 1.5, Research in intensity image face recognition generally falls into two categories[4]: holistic (global) methods and feature-based methods. In Feature-based methods, the location of some fiducial points on the face such as the eyes, the nose, the mouth etc. are used as feature. But in Holistic methods, the face is recognized as one entity without explicitly isolating different regions in the face. Holistic techniques utilize statistical analysis, neural networks and transformations. They also usually require large samples of training data. In this research, the method is holistic in nature but we have used a reduced data set. The method is discussed in the following sections.

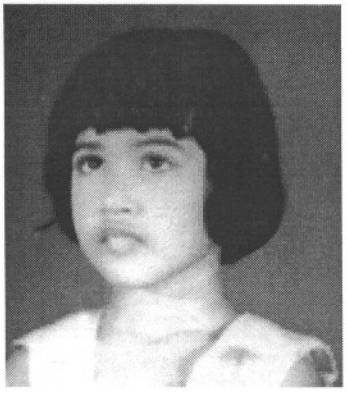
Pattern classification techniques fall into two broad categories, numeric and non-numeric. Numeric techniques can be considered as measures made on the geometric pattern space. Non-numeric techniques are those that take us into the domain of symbolic processing that is dealt with by such methods a fuzzy set. We shall consider only numeric techniques because the methods adopted in this research are of this type. More specifically, the methods used are distance measurement and the neural network learning. So the use of these methods in our Human Face Recognition system will be discussed in the last two sections.

4.2 Feature extraction from facial images

Feature extraction is a subject of effective Face Recognition and it helps to make easy the classification task. It is well known that the feature extraction plays an important role in Face Recognition system. The basic steps for feature extraction from facial image are as follows:

4.2.1 Image acquisition: A total of ten people were selected to collect photographs in five different angles for each. For each person, the maximum angle difference was 90^0 and it was divided into five equal angles to take the photographs starting from 45^0-left to 45^0-right. Figure-4.1(a) shows these angles of the rotations of faces during capturing photographs. Figure-4.1(b) shows a photograph with 0^0 rotation and Figure-4.1(c) shows another photograph with 45^0 rotation. The photographs were scanned and stored as windows bitmap file (bmp) for the next step of processing.





(a) Angles of rotations (b) 0^0 rotation (c) 45^0-left rotation

Figure-4.1: Method of Capturing face images

4.2.2 Scanning the Image

Scanning is a very simple process, which converts any picture data in a paper to digital image so that the same image is available to the computer. The scanning components in face recognition system include the scanning equipment, i.e., the Scanner machine. Scanner is used to enter data into the computer automatically. To use this form of data entry, the data must be existed in printed form either as text or graphics form. Scanner devices can then read the printed data and convert it into a digital form that the computer can process. In this research work, Astra 610P machine is used for scanning. The black and white photographs are scanned. Scanning is done in gray scale mode and resolution is set up at 100 dot per inch (dpi). After scanning the facial images are saved into Bitmap Image File format (BMP).

4.2.3 Feature Collection

Feature is the most important part for face recognition technique. To collect features, three types of face images are considered in this research.

- Original face image
- Edge detected face image and
- Thresholded face image

4.2.3.1 Original Image

In this research, original images are considered as separated face images from photographs. As said before, the face portion, being saved in BMP format after scanning the photographs, is separated by Photophinish Software and saved also in BMP format. Figure-4.2(b) shows the separated face which is considered as original face image for this work. This face is taken directly for feature collection in this method and also for edge detection and thresholding operations.

(a) The photograph (b) Separated face

Figure- 4.2: Separation of face from photograph image.

4.2.3.2 Edge detection

Edge detection is one kind of technique applying in this system. As said before the edges of items in an image hold much of the information in the image. The edges indicate where items are, their size, shape and something about their texture. Edge points can be thought of as pixel locations of abrupt gray-level changes[20]. The number of masks is used for edge detection. In this research edge detection is performed by filtering operation using Sobel operator, which requires the convolution of a small kernel (3 ×3 pixels) over the image. The Sobel mask operators are as follows

W_1

1	0	-1
2	0	-2
1	0	-1

W_2

-1	-2	-1
0	0	0
1	2	1

The edge detection operation was applied to the original faces using Sobel operator and after edge detection the edge detected faces are saved in BMP format. Figure-4.3 shows the resulting face after edge detection. This edge-detected face is taken for feature collection.

(a) Original face (b) Edge detected face

Figure-4.3: The result of edge detection

4.2.3.3 Thresholding

The technique of thresholding is another kind of operation for feature collection. Thresholding transforms a dataset containing values that vary over some range into a new dataset containing just two values[21]. For this operation the original faces i.e. separated faces are considered also and after applying thresholding technique the faces are saved in BMP format. In this case a gray level value of 200 was used as threshold value.

(a) Original face (b) Thresholded face

Figure-4.4: The result of thresholding

Gray levels more than 200 were converted to 255 and those less than 200 were converted to 0. Figure-4.4 shows the resulting face after thresholding.

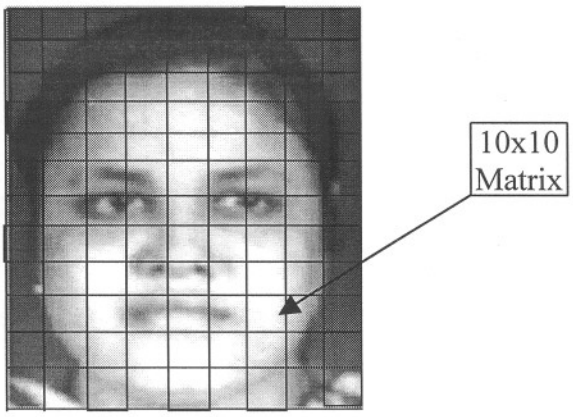


Figure-4.5: Dividing the face in small grids.

Finally, the original face (Figure-4.2b), edge detected face (Figure-4.3(b)) and Thresholded face (Figure-4.4(b)) are taken to extract three types of features. The face is divided in small grids as shown in Figure-4.5. Each section of the face then includes 10x10 matrix of gray levels. Thus 100 data are taken from a grid and an average is calculated to collect a feature. The size of face images used in various experiments in this research, were nearly 100x100 in dimension in terms of pixels. But there were some variations in this dimension. Most of the faces have height over 100 pixels and width below 100 pixels. So a compromise 100x90 was taken to cover all the faces. So total 90 feature data was collected in this step.

4.4 Recognition by Template Matching

Templates are formed by the features from the original, edge detected or thresholded face. In template matching, 10 sets of features of reference faces are stored as 10 reference templates. The features of unknown face is extracted and saved as another template. The distances between the unknown template and reference templates are measured to find the best match[4]. Several distance measurement techniques[32] are used in this method. Two of them are used in this research. They are discussed below.

Hamming Distance: The most basic measure and one that is widely used because of its simplicity, is the Hamming distance measure. For two vectors

$$X = (x_1, x_2, \ldots\ldots\ldots\ldots)$$

$$Y = (y_1, y_2, \ldots\ldots\ldots\ldots)$$

the Hamming distance is found by evaluating the difference between each component of one vector with the corresponding component of the other, and summing these

differences to provide an absolute value for the variation between the two vectors. The measure is defined by

$$H = \sum_{i=0}^{N} \left(\left| x_i - y_i \right| \right)$$

Euclidean Distance: One of the most common methods used is the Euclidean distance measure. Let us consider an example in a rectangular coordinate system where we have two vectors (X and Y) that we wish to find the distance between them d(X,Y). The shortest distance is the Euclidean distance that is defined by:

$$d(X,Y)_{euc} = \sqrt{\left(\sum_{i=1}^{N} \left(X_i - Y_i \right)^2 \right)}$$

where N is the dimensionality of the vector.

4.5: Recognition using Artificial Neural Network

The theoretical discussion on Artificial Neural Networks has been presented in chapter-3. Now the *Face Recognition system* using neural network is described below.

The complete system can be described in two stages:

i) One is the *learning stage* and

ii) Another is the *testing stage*.

The block diagram of these two stages are shown in Figure-4.6(a) and 4.6(b).

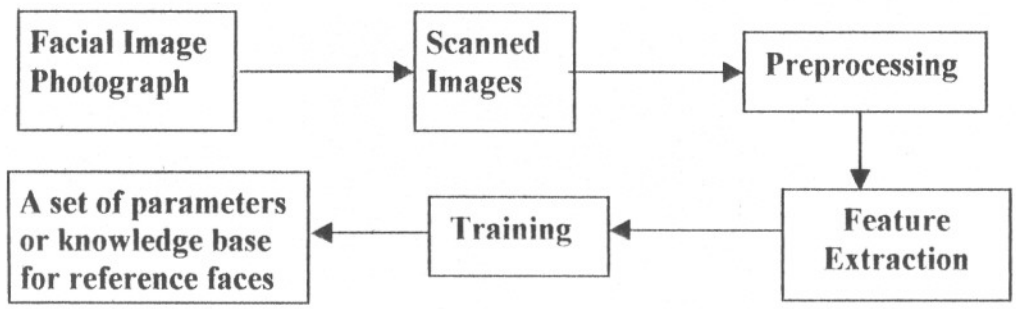


Figure 4.6-(a) Learning Stage

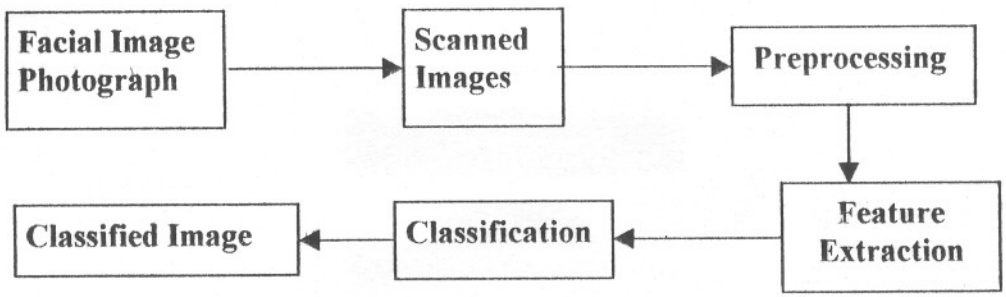


Figure-4.6 (b): Testing Stage

Figure-4.6: Block Diagram of Face Recognition System using Neural Network

In the training stage, some reference faces are selected to create a knowledge base. Facial image photographs are to be scanned by a scanner machine (Astra 610P). The next step is preprocessing which includes separation of facial part from total image and necessary enhancement to improve the quality of images. The next phase is the feature extraction in which one feature is extracted from each 10x10 block of the face image matrix. These features are fed into neural network for training and after training the output is saved as knowledge base.

In the testing stage, the system is allowed to recognize an unknown face. In this case, the all steps are done in the same way as in the learning stage before classification. In the phase of classification, the selected features having enough information within it are used to identify each facial image photograph uniquely. This is done by the system using the knowledge base and extracted features of unknown face as input to the network. At the last step, final decision about the unknown face is taken by comparing the test output with those during training phase.

4.6 Discussion

In this chapter, the methods for extracting features from human face images have been discussed. Some part of the above discussion was made in the light of theoretical discussion in chapter-2. The use of distance measurement techniques in template matching and of Artificial Neural Networks has also been discussed. The Artificial Neural Network had been discussed in detail in chapter-3, here the plan of application in the current research has discussed. The practical consideration and detailed algorithms of all the methods are developed in chapter-6.

Chapter-5

Software System for Human Face Recognition

5

SOFTWARE SYSTEM FOR HUMAN FACE RECOGNITION

5.1 Introduction

The software developed for the recognition of Human Face is presented in this chapter. In accordance with the nature and scope of this research, six separate program modules are written to cover all the operations. They are

- The Main Program
- The Edge Detection Program
- The Image Thresholding Program
- The Feature Extraction Program
- The Recognition Program
- The Neural Network Program

The programs are written in C language and compiled with Turbo C++ compiler. The methodological steps implemented in this software have been discussed in chapter-4. Here the algorithms of the main program and other modules will be discussed. The prototypes of the functions used in this software are given in appendix-A.

5.2 The Main Program

The total software system is shown in block diagram approach in figure-5.1. The main program is organized with four user defined header files, three functions and the main function itself. The header files are:

feature.h: Includes functions related to feature extraction from facial images. It also includes edge detection and image thresholding functions.

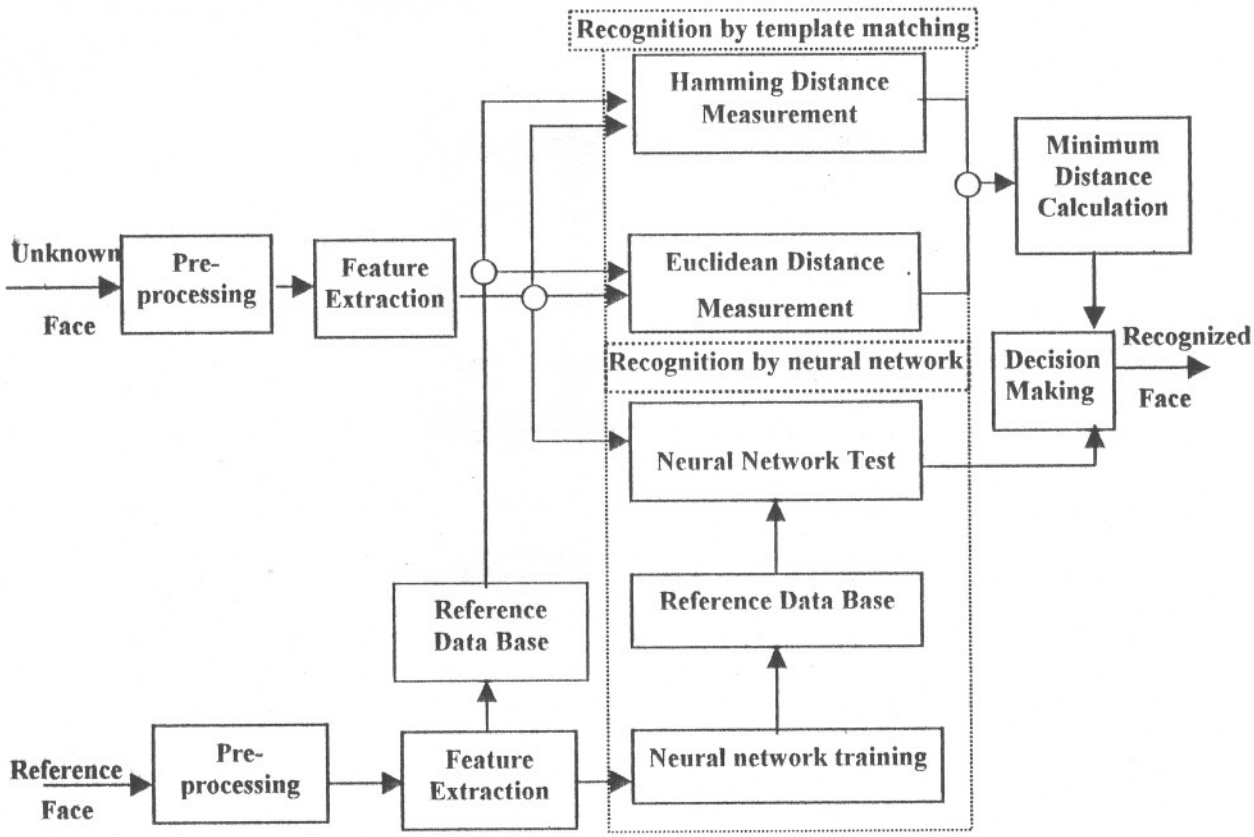


Figure-5.1: The main blocks of the face recognition system

image.h: Includes arrays of file names, which contains reference features and face data.

recog.h : includes functions related to the recognition methods.

neural.h : include functions related to the neural network.

The main function simply handles some options from a menu. The first is for the selection of feature extraction method, the options are:

1. Feature from Original Face
2. Feature from Edge Detected Face
3. Feature from Thresholded Face

An integer from 1, 2 or 3 must be entered to select a category of face. In this stage, a function *process_image()* is used for feature extraction. This function takes only one option selected from above menu as argument and drives the program with the required feature extraction method. It uses functions and data from feature.h and image.h. Next menu is the recognition menu, the options are:

1. Neural Network Training
2. Recognition

According to the choice from 1 or 2 the above operations drive the program in the required direction. The first item in the menu associates with neural network training. The function *neural()* is for this purpose. For the second option, the function *rec_result()* is used in main program for recognition. It also gets one option from first menu as its argument and drives the program with the required type of feature and reference database. It uses another function *recognition()* and data from image.h. The flowchart of the main program is shown in figure-5.2. The algorithms for functions *process_image()*, *rec_result()*, *recognition()* and *neural()* are given in algorithm-6, 7, 8 and 11 respectively. The functions *recognition()*, *rec_result()* and *neural()* are further discussed in the following sections of this chapter.

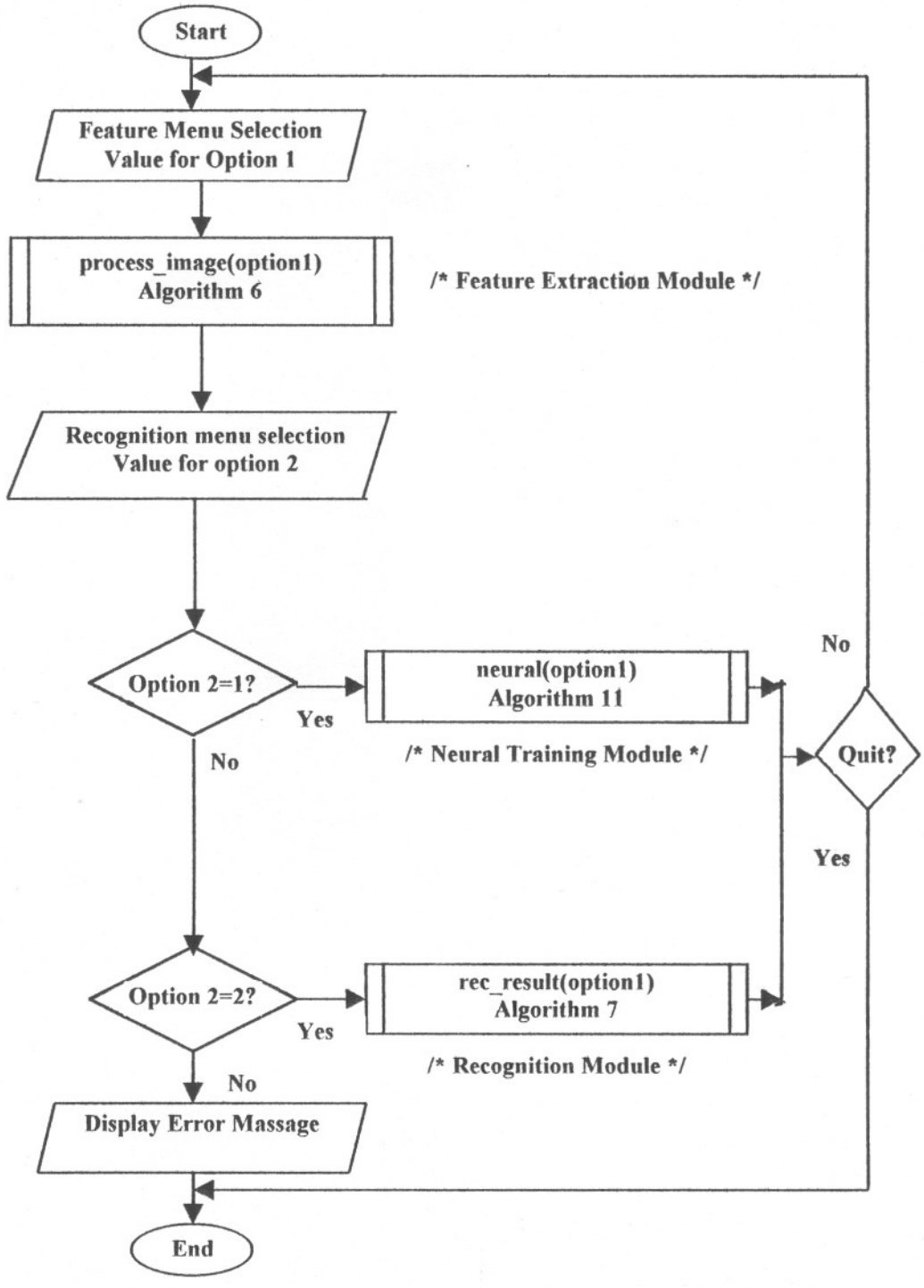


Figure-5.2: Flowchart of the Main Program

5.3 The Edge Detection Program

The related theories of edge detection of image have been given in section-2.4.1. The basic steps for edge detection are given below:

1. Read a disk file, which contains the Image data.
2. Store three lines of data in a buffer
3. Apply two Sobel mask operators on this data
4. Calculate two product-sums of this data for two masks.
5. Store the maximum absolute value of two product-sums in an output file.
6. Repeat from step-2, until end of file.

These steps are implemented by only one function *edge()*. The detailed algorithm of this function *edge()* is given in Algorithm-2. The function is included in the header file feature.h. The edge.h also includes two arrays, which contains two Sobel mask operators as below.

```
int s1[3][3]={{1,0,-1},
              {2,0,-2},
              {1,0,-1}};
```

```
int s2[3][3]={{-1,-2,-1},
              {0,0,0},
              {1,2,1}};
```

The arrays are globally declared. The function has two arguments, one is input file pointer and the other is output file pointer. The function reads input image from input file and after processing, returns edge detected image to output file.

5.4 The Thresholding Program

The related theories of image thresholding have been given in Section-2.5. The basic steps in image thresholding program are given below:

1. Read a disk file, which contains the image.
2. Copy header information to output file.

3. Store one line of data in a buffer.
4. Convert the buffer as follows:

   If a gray level value is greater than 200, then convert it to maximum (255).

   Otherwise, convert it to minimum (0).
5. Store the buffer to output file.
6. Repeat from step-3, until end of file.

These steps are implemented by using only one function *threshold()*. The function is included in the header file feature.h and its detailed algorithm is given in Algorithm-3. The function *threshold()* takes two file pointers as argument, one is input file to read the image and the other is output file to store thresholded image.

5.5 The Feature Extraction Programs

The feature extraction programs consists of two functions: *process_image()* and *min_image()*. The *min_image()* calculates a set of feature from original, edge detected or thresholded image. As described in section-4.2.3, the face is divided in small grids so that each grid includes 10x10 matrix of gray levels. Thus 100 data are taken from a grid and an average is calculated to collect a feature. The *min_image()* takes two file pointers as argument, one is input file to read the image and the other is output file to store the feature set. The detailed algorithm of this function is given in Algorithm-4. The function *process_image()* takes one option selected by first menu and an array of file names as arguments. At first, it converts the input image file to one type of face category as selected from the first menu, then it calls function *min_image()* to extract a set of features. The function

process_image() uses functions and data from feature.h and image.h. The detailed algorithm of this function is given in Algorithm-6.

5.6 The Neural Network Program

Here a program for back-propagation network for ten faces has been implemented according to the discussion in chapter-3. The function *neural()* is written for training ten faces. It takes one option selected by first menu as argument. The detailed algorithm has been given in Algorithm-11. Another function *neu_test()* has been developed to recognize unknown faces using the training result. Detailed algorithm of this function is given in Algorithm-12. It performs the same function as the module *recognition()*. But it takes only two arguments. The first argument is the unknown face-image file and the second is the index number associated with this file in the database.

5.7 The Recognition Program

The recognition program has two parts. In one part, distance measurement techniques in template matching have been used in recognition. In the other part, neural network has been used in recognition. The theoretical discussions on these methods have been given in chapters 3 and 4. The function *rec_result()* is designed to bring all the unknown face-image under recognition scheme. First of all, the function *rec_result()* selects a recognition method, template matching or neural network. Then it inputs all the image files one-by-one, determines total number of correct recognition and calculates the recognition rate. The dominating function to recognize an unknown face by template matching is *recognition()*.This

function takes three arguments. The first argument is the unknown image file and second is the index number associated with this file in the database. The other one is first menu selection parameter. This function makes decision whether the unknown file is correctly recognized or not. The detailed algorithms of these two functions are given in algorithms-7 and 8 respectively. Two distance measurement techniques for template matching, Hamming distance and Euclidean distance have been implemented by two functions *ham_dis()* and *uc_dis()* respectively. Their algorithms are given in algorithms-9 and 10. Both the functions have three arguments. Two of them are feature matrices of known and unknown face and the third is the number of features. One of these two functions is called by the function *recognition()* according to the selection of distance measurement technique from a menu. Both the functions return the distance between two feature matrices. The function *neu_test()* is for neural network test program that has been used in the function *rec_result()* as an alternative to the function *recognition()*. The function *neu_test()* is already discussed above.

5.8 Programming Algorithms

The algorithms of the programming modules are given in the following pages one after another. They are arranged on the basis of the similar programs. This means that the algorithms required for feature extraction program are arranged together and so on.

Algorithm-1

direct(Input File Pointer, Output File Pointer)

This algorithm takes two file pointers, one for input file and the other for output file. It extracts data from input file and restore to a temporary output file.

1. Start
2. Loop until end of file
3. Get a character from input file steam
4. Put the character to the output stream
5. Repeat from step-2
6. End

Algorithm-2

edge(Input File Pointer, Output File Pointer)

This algorithm takes two file pointers, one for input file and the other for output file. It extracts data from input file and after edge detection, writes data to the output file.

1. Read and store two bytes for bitmap file type
   Loop i=0 to 1 with step=1
      Get a character from input file
      Put the character to output file
   i loop end
2. Read and store four bytes for file size
   ts=0 (Initialize File Size -ts)
   Loop i=0 to 3 with step=1
      Get a character (ch) from input file
      Put the character to output file
      $ts=ts+ch\times(256)^i$
   i loop end
3. Read and store twelve bytes for four bitmap parameters

Loop i=0 to 11 with step=1

Get a character (ch) from input file

Put the character to output file

If(i=4) h=ch

i loop end

4. Read and store four bytes for bitmap width

i_w=0 (Initialize bitmap width–i_w)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$i\_w=i\_w+ch\times(256)^i$

i loop end

5. Read and store four bytes for bitmap length

i_l=0 (Initialize bitmap width –i_l)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$i\_l=i\_l+ch\times(256)^i$

i loop end

6. Read and store eight bytes for three bitmap parameters

Loop i=0 to 7 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

7. Read and store four bytes for bitmap length

ds=0 (Initialize bitmap image size – ds)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$ds=ds+ch\times(256)^i$

i loop end

8. Read and store rest of the bytes for information header

Loop i=0 to h-40 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

9. Read and store 1024 bytes of color table

Loop i=0 to 1024 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

10. Print i_l, i_w, ts, ds, h

11. Allocate memory for image transfer buffer (w) with size=3×i_w×size of char pointer

12. Initialize the buffer with 0.

13. Loop n1=0 to image length with step=1

Transfer a row of the image to the last row of buffer (w).

Loop j=0 to image width with step=1

Get a character (ch) from input image file

w_{2j} = ch

j loop end

Loop n2=0 to image width with step=1

y1=y2=0

Loop k1=-1 to 1 with step=1

Loop k2=-1 to 1 with step=1

If (n2+k2)<0 or (n2+k2-1)>= i_w then continue

temp1=$w_{(1+k1)(n2+k2-1)}$

y1=y1+s1[1+k1][1+k2]*temp1;

y2=y2+s2[1+k1][1+k2]*temp1;

k2 loop end

k1 loop end

y1 = absolute value of y1

y2 = absolute value of y2

if y1>y2 then zn2 = y1 otherwise y2

put zn2 to output file

n2 loop end

n1 loop end

14. Loop j=0 to ts-ds-1 with step=1

Get a character (ch) from input image file

Put the character to output file

j loop end

15. End

Algorithm-3

threshold(Input File Pointer, Output File Pointer)

This algorithm takes two file pointers, one for input file and the other for output file. It extracts data from input file and after thresholding, writes data to the output file.

---

1. Read and store two bytes for bitmap file type

Loop i=0 to 1 with step=1

Get a character from input file

Put the character to output file

i loop end

2. Read and store four bytes for file size

ts=0 (Initialize File Size -ts)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$ts=ts+ch\times(256)^i$

i loop end

3. Read and store twelve bytes for next four bitmap parameters

Loop i=0 to 11 with step=1

Get a character (ch) from input file

Put the character to output file

If(i=4) h=ch

i loop end

4. Read and store four bytes for bitmap width

i_w=0 (Initialize bitmap width –i_w)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$i\_w=i\_w+ch\times(256)^i$

i loop end

5. Read and store four bytes for bitmap length

i_l=0 (Initialize bitmap width –i_l)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

$i\_l=i\_l+ch\times(256)^i$

i loop end

6. Read and store eight bytes for next three bitmap parameters

Loop i=0 to 7 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

7. Read and store four bytes for bitmap image size

ds=0 (Initialize bitmap image size – ds)

Loop i=0 to 3 with step=1

Get a character (ch) from input file

Put the character to output file

ds=ds+ch×(256)^i

i loop end

8. Read and store rest of the bytes for information header

Loop i=0 to h-40 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

9. Read and store 1024 bytes of color table

Loop i=0 to 1024 with step=1

Get a character (ch) from input file

Put the character to output file

i loop end

10. Print i_l, i_w, ts, ds, h

11. Loop i=0 to image length with step=1

Loop j=0 to image width with step=1

Get a character (ch) from input image file

If ch>200 then ch=255

Otherwise ch=0

Put ch to output file

j loop end

i loop end

12. Loop j=0 to ts-ds-1 with step=1

Get a character (ch) from input image file

Put the character to output file

j loop end

13. End

Algorithm-4

min_image(Input File Pointer-*in*, Output File Pointer-*out*)

This algorithm takes two file pointers, one for input file and the other for output file. It uses a global array $image_{ij}$ to store data extracted from input file. This algorithm extracts a set of features and writes to the output file.

1. Start
2. Initialize k=0, l=0
3. Call *bmpread()* with input file pointer *in*.
4. Loop i=0 to 99 with step=10

        Loop j=0 to 89 with step=1
         sum=0
         Loop m=i to i-1+10 with step=1
                Loop n=j to j-1+10 with step=1
                    sum=sum+$image_{mn}$
                n loop end
            m loop end
                average=sum/100
                put average to output file (*out*)
        j loop end
    i loop end
5. End

Algorithm-5

bmpread(Input File Pointer-*in*)

This algorithm extracts data from bmp file and takes one file pointer as argument for input file. It uses a global array $image_{ij}$ to store data extracted from input file.

1. Start
2. Read bmp file header information
3. Read bitmap information header
4. Skip color table bytes
   Loop i=0 to 1024 with step=1
      Get one character from input file stream
   i loop end
5. Loop i=0 to image length with step=1
      Loop j=0 to image width with step=1
       Get one character (ch) from input file stream
       $Image_{ij}$=ch
    j loop end
    i loop end
6. End

Algorithm-6

process_image(type1)

Here type1 is the type of pre-processing operation (original=1, edge detection=2 and thresholding=3), This function inputs 50 bmp files containing face images from database and outputs the same number of feature files.

1. Start
2. Select a pre-processing operation from a menu (type1)
3. Loop i=0 to 49 with step=1

4. Open a file (*in*) named file_name[i] with "bmp" extention to read image data
5. Open a file (*temp*) named "temp.bmp" to write processed data
6. If type1=1 then

   Call *direct()* with file pointers *in* and *temp*

   Add extension "org" to file_name[i] and store the name to *filename*
7. If type1=2 then

   Call *edge()* with file pointers *in* and *temp*

   Add extension "edg" to file_name[i] and store the name to *filename*
8. If type1=3 then

   Call *threshold()* with file pointers *in* and *temp*

   Add extension "thd" to file_name[i] and store the name to *filename*
9. Otherwise

   Print the massage "Not in the range.." and go to step-16
10. Close the file (*temp*) named "temp.bmp".
11. Open a file (*out*) named *filename* to write feature data

    Open a file (*temp*) named "temp.bmp" to read processed data
12. Call *min_image()* with file pointers *temp* and *out*
13. Close the file (*temp*) named "temp.bmp".
14. Close the file opened by *in*
15. Close the file opened by *out*
16. End

Algorithm-7

rec_result(type1)

This algorithm determines total and percentage of recognized faces.

1. Start
2. Select recognition method from a menu (d or n)
3. Initialize count = total count = 0
4. Loop i = 0 to 9 with step =1
5. if method = 'd', count = count + recognition(i,filename,type1)[Algorithm 8]
6. if method = 'n', count = count + neu_test(i, filename)[Algorithm 12]
7. i loop end
8. Output count
9. Total count = Total count + count
10. count = 0
11. Repeat step-1 for all group of face image samples
12. Output total count
13. Percent count=(total count/total image files)×100
14. Output percent count
15. End

Algorithm-8

recognition(index, filename, type1)

This algorithm calculates hamming or Euclidean distance and makes decision either the input face image is exactly recognized or not by template matching technique. Here the arguments bear the same meaning as used for the above algorithms.

1. Start
2. Open an input file named as *filename*
3. Read data from filename and store in an array (data1).

4. Close input file
5. Loop j = 0 to 9 with step=1
6. If type1=1, select a file from original face database with index j
7. If type1=2, select a file from edge detected face database with index j
8. If type1=3, select a file from thresholded face database with index j
9. Open the selected file
10. Read the file and store in an array (data2).
11. Close the file
12. Select distance measurement techniques from a menu
13. Call *ham_dis()*[Algorithm 9] or *uc_dis()*[Algorithm 10] according to the menu selection with data1, data2 and 90 as arguments to calculate the distance
14. If the distance is less than previous one then record the current index, otherwise skip
15. j loop end
16. If the recorded index and argument index belongs to the same group return 1, Otherwise return 0
17. End

Algorithm-9

ham_dis(udata[ ], rdata[ ], n)

This module calculates Hamming distance between features of unknown face image in the array udata[ ] and that of reference in the array rdata[ ]. n is the number of features in the two feature sets.

---

1. Start
2. distance=0
3. Loop i=0 to n-1 with increment step=1

distance=distance+| udata[i]-rdata[i] |

End of i loop

4. return distance
5. End

Algorithm-10

uc_dis(udata[ ], rdata[ ], n)

This module calculates Euclidean distance between features of unknown face image in the array udata[ ] and that of reference in the array rdata[ ]. n is the number of features in the two feature sets.

1. Start
2. distance=0
3. Loop i=0 to n-1 with increment step=1

   distance=distance+[udata[i]-rdata[i]]^2

   End of i loop
4. distance=$\sqrt{}$(distance)
5. return distance
6. End

Algorithm-11

neural(type1)

This algorithm is for training ten human face images. It takes only one argument. The argument is the input category (type of features).

1. Start
2. Initialize target vector
3. Initialize some temporary variables

   IUNITS=90, OUTUNITS=10, HUNITS=20, eta1=0.1 , eta2=0.2 , spr= 0.4, spread= 0.25, scale_down_factor=200, trmax=30000

4. Open output files to store weights and output pattern
5. Initialize random number generator
6. Loop i= 0 to 9 with step = 1
    a) If type1=1, get i numbered filename from original face feature database
    b) If type1=2, get i numbered filename from edge detected face feature database
    c) If type1=3, get i numbered filename from thresholded face feature database
    d) Otherwise return to calling function.
    e) Open input file with file name
    f) Loop j = 0 to IUNITS-1 with step=1

        Get one data from input file

        $$y_{i,j} = data/\text{scale\_down\_factor}$$

    g) Close the file

    i loop end
7. Initialize input-to-hidden weight vector itohweight$_{ij}$ with random weights
8. Initialize hidden-to-output weight vector htooweight$_{ij}$ with random weights
9. Initialize input-threshold vector uh$_k$ with random thresholds
10. Initialize output-threshold vector uo$_k$ with random thresholds
11. Starts learning loop

    Loop iterate = 0 to trmax-1 with step=1

        Loop a=0 to 9 with step=1

        a) Feed forward : 256 input to 20 hidden

            1. Loop j=0 to HUNITS-1 with step=1

$netih_j = 0$

Loop i=0 to IUNITS-1 with step=1

$netih_j = netih_j + itohweight_{ij} \times y_{aj}$

i loop end

j loop end

2. Loop j=0 to HUNITS-1 with step=1

$activeh_j = netih_j + uh_j$

$hout_j = 1/\left(1 + \exp(-spread * activeh_j)\right)$

j loop end

b) Feed forward : 20 hidden to 10 output

1. Loop k=0 to OUNITS-1 with step=1

$out_k = 0$

Loop j=0 to HUNITS-1 with step=1

$out_k = out_k + htooweight_{jk} \times hout_j$

j loop end

k loop end

2. Loop k=0 to OUNITS-1 with step=1

$activeo_k = out_k + uo_k$

$oout_k = 1/\left(1 + \exp(-spr * activeo_k)\right)$

$error_k = t\arg et_{ak} - oout_k$

k loop end

c) Compute the error correction for htooweight matrix elements

Loop k=0 to OUNITS-1 with step=1

Loop j=0 to HUNITS-1 with step=1

$deltawho_{jk} = error_k \times hout_j \times eta2$

$htooweight_{jk} = htooweight_{jk} + deltawho_{jk}$

$deltauo_j = error_k \times oout_k \times (1 - oout_k) \times eta2$

$$uo_k = uo_k + deltauo_j$$

j loop end

k loop end

d) Compute the error correction for itohweight matrix elements

1. Loop j=0 to HUNITS-1 with step=1

$$sigma_j = o$$

Loop k=0 to OUNITS-1 with step=1

$$sigma_j = sigma_j + error_k \times htooweight_{jk}$$

k loop end

j loop end

2. Loop i=0 to IUNITS-1 with step=1

Loop j=0 to HUNITS-1 with step=1

$$deltawih_{ij} = spread \times eta1 \times hout_j \times y_{ai} \times (1 - hout_j) \times sigma_j$$

$$deltauh_j = spread \times eta1 \times hout_j \times (1 - hout_j) \times sigma_j$$

$$itohweight_{ij} = itohweight_{ij} + deltawih_{ij}$$

$$uh_j = uh_j + deltauh_j$$

j loop end

k loop end

a loop end

iterate loop end

12. Open an output file to save weights and thresholds
13. Save final input -to- hidden weights to output file
14. Save final hidden -to- output weights to output file
15. Save final hidden-thresholds to output file
16. Save final output-thresholds to output file
17. Close the output file
18. End

Algorithm-12

neu_test (file_name, file_no)

This function takes a filename that contains unknown face image and the index number of the file in the database as its arguments. It makes decision either the unknown face is exactly recognized or not. It returns 1, if the face is recognized correctly and 0 otherwise.

---

1. Start
2. Initialize some temporary variables
   IUNITS=90, OUTUNITS=10, HUNITS=20, eta1=0.1 , eta2=0.2 , spr = 0.4, spread = 0.25, scale_down_factor=200
3. Open input file containing weights and thresholds
4. Read input to hidden weights
   Loop i=0 to IUNITS-1 with step=1
      Loop j=0 to HUNITS-1 with step=1
         Read one data from input file and assign it to $itohweight_{ij}$
      j loop end
   i loop end
5. Read hidden to output weights
   Loop i=0 to HUNITS-1 with step=1
      Loop j=0 to OUNITS-1 with step=1
         Read one data from input file and assign it to $htooweight_{ij}$
      j loop end
   i loop end
6. Read hidden thresholds
   Loop j=0 to HUNITS-1 with step=1
      Read one data from input file and assign it to uh_j

j loop end

7. Read output thresholds

Loop k=0 to HUNITS-1 with step=1

Read one data from input file and assign it to uh_k

k loop end

8. Close the input file

9. Open unknown feature file specified by file_name as argument

10. Loop j = 0 to IUNITS-1 with step=1

Get one data from input file

$y_j = data/\text{scale\_down\_factor}$

j loop end

11. Close the input file

12. Output computation for hidden neurons

a. Loop j=0 to HUNITS-1 with step=1

$netih_j = 0$

Loop i=0 to IUNITS-1 with step=1

$netih_j = netih_j + itohweight_{ij} \times y_i$

i loop end

j loop end

b. Loop j=0 to HUNITS-1 with step=1

$activeh_j = netih_j + uh_j$

$hout_j = 1/\left(1 + \exp(-spread \times activeh_j)\right)$

j loop end

13. Feed forward -- 20 hidden to 10 output

Loop k=0 to OUNITS-1 with step=1

$out_k = 0$

Loop j=0 to HUNITS-1 with step=1

$$out_k = out_k + htooweight_{jk} \times hout_j$$

j loop end

k loop end

14. Compute final outputs and maximum output

Loop k=0 to OUNITS-1 with step=1

$activeo_k = out_k + uo_k$

$oout_k = 1/(1 + \exp(-spr \times activeo_k))$

If $oout_k > \max$, $\max = oout_k$ and $final=k$

k loop end

15. Make decision - if the output index (final) and argument index (file_no) belongs to the same group return 1, Otherwise return 0

17. End

5.9 Discussion

The developed software for our research has been presented in this chapter. Its various parts are discussed with necessary explanation. The program module for edge detection was prepared as a moderated version of that in [20]. The program for Neural Network training and testing was written on the basis of a demonstration program given in [29]. Several research workers in Rajshahi University have used similar programs in various approaches. Also in Islamic University of Kushtia, similar programs for recognition of Bangla speech have been written[32]. The developed system gets help from their works. Before including within the main program, the training program was tested for its best performance with variation in different parameters. Algorithms are presented in a way that programmers can easily convert them as code in any language in future.

Chapter-6

The Experimental Results

6.1 Introduction

The programs for our Face Recognition Scheme were developed in chapter-5. This chapter explores the performance of the system for various features and recognition methods. In first step, experiments were conducted to find best parameters and target output pattern for neural network. The target output patterns were Unit Matrix, Hamming code and 7-bit ASCII code. The patterns are given below.

Names of Faces	Unit Matrix	Hamming Code	7-bit ASCII Code
Faruk	1 0 0 0 0 0 0 0 0 0	1 1 0 1 0 0 1	0 1 1 0 0 0 0
Lisa	0 1 0 0 0 0 0 0 0 0	0 1 0 1 0 1 0	0 1 1 0 0 0 1
Prisly	0 0 1 0 0 0 0 0 0 0	1 0 0 0 0 1 1	0 1 1 0 0 1 0
Sabina	0 0 0 1 0 0 0 0 0 0	1 1 0 1 1 0 0	0 1 1 0 0 1 1
Faria	0 0 0 0 1 0 0 0 0 0	0 1 0 0 1 0 1	0 1 1 0 1 0 0
Masum	0 0 0 0 0 1 0 0 0 0	1 1 0 0 1 1 0	0 1 1 0 1 0 1
Hasan	0 0 0 0 0 0 1 0 0 0	0 0 0 1 1 1 1	0 1 1 0 1 1 0
Rahad	0 0 0 0 0 0 0 1 0 0	1 1 1 0 0 0 0	0 1 1 0 1 1 1
Ivan	0 0 0 0 0 0 0 0 1 0	0 0 1 1 0 0 1	0 1 1 1 0 0 0
Arif	0 0 0 0 0 0 0 0 0 1	1 0 1 1 0 1 0	0 1 1 1 0 0 1

Experiments to find best features and recognition methods were in the next step. All of these tests were conducted with a reference database of ten faces. Total 50 unknown faces were included to test the systems performance. The detailed results

for each experiment were given in tabular form and the best performance was marked with gray shading.

6.2 Recognition by Neural Network

The neural network program was tested to find best performance with the variation in different parameters before using it with the main module. The two spread parameters were used to model the sigmoid function. These were *spread* and *spr*. The two learning rates were used to compute error corrections. These were *eta1*- learning rate for input to hidden layers, and *eta2*- learning rate for hidden to output layers. These four parameters and number of hidden unit were varied to set the value of the parameters for the best performance. During this test, input feature type and output target matrices were varied to study the network behavior. The results are given in table-6.1-6.5. The summary of this test is given in table-6.6 and also shown in two bar graphs in figure-6.1 and 6.2. In all tables, best performances are marked with gray shading. As seen, the best performing target output is Unit Matrix and best feature is that what was calculated from thresholded image. But with these, the net still has some error. The resulting output pattern and error after training is shown in table-6.7 for best performing target pattern Unit Matrix.

Table No. 6.1: Recognition rates against variation in net parameters using *Features collected after threshold* as input pattern and *Unit Matrix* as target output

No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
05	0.1	0.2	0.25	0.25	66%
10	0.1	0.2	0.25	0.25	82%
20	0.1	0.2	0.25	0.25	90%
30	0.1	0.2	0.25	0.25	86%
20	0.1	0.2	0.1	0.25	90%
20	**0.1**	**0.2**	**0.4**	**0.25**	**92%**
20	0.1	0.2	0.1	0.1	90%
20	0.1	0.2	0.25	0.1	90%
20	0.1	0.2	0.4	0.1	84%
20	0.1	0.2	0.1	0.4	92%
20	0.1	0.2	0.25	0.4	88%
20	0.1	0.2	0.4	0.4	90%
20	0.05	0.1	0.4	0.25	82%
20	0.05	0.2	0.4	0.25	92%
20	0.05	0.3	0.4	0.25	36%
20	0.1	0.1	0.4	0.25	92%
20	0.1	0.3	0.4	0.25	38%
20	0.2	0.1	0.4	0.25	86%
20	0.2	0.2	0.4	0.25	88%
20	0.2	0.3	0.4	0.25	32%

Table No. 6.2: Recognition rates against variation in net parameters using *features collected after edge detection* as input pattern and *Unit Matrix* as target output.

No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
05	0.05	0.05	0.25	0.25	48
10	0.05	0.05	0.25	0.25	72
15	0.05	0.05	0.25	0.25	78
20	0.05	0.05	0.25	0.25	78
30	0.05	0.05	0.25	0.25	78
15	0.05	0.05	0.25	0.35	78
15	**0.05**	**0.05**	**0.25**	**0.4**	**80**
15	0.05	0.05	0.35	0.25	76
15	0.05	0.05	0.35	0.35	80
15	0.05	0.05	0.35	0.4	80
15	0.05	0.05	0.4	0.25	76
15	0.05	0.05	0.4	0.35	78
15	0.05	0.05	0.4	0.4	78
15	0.05	0.1	0.25	0.25	76
15	0.05	0.2	0.25	0.25	70
15	0.1	0.05	0.25	0.25	74
15	0.1	0.1	0.25	0.25	76
15	0.1	0.2	0.25	0.25	74
15	0.2	0.05	0.25	0.25	74
15	0.2	0.1	0.25	0.25	74
15	0.2	0.2	0.25	0.25	80

Table No. 6.3: Recognition rates against variation in net parameters using *features collected from original face* as input pattern and *Unit Matrix* as target output.

No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
10	0.05	0.05	0.25	0.25	86
15	0.05	0.05	0.25	0.25	84
20	0.05	0.05	0.25	0.25	90
30	0.05	0.05	0.25	0.25	88
20	0.05	0.05	0.25	0.35	84
20	0.05	0.05	0.25	0.4	88
20	0.05	0.05	0.35	0.25	82
20	0.05	0.05	0.35	0.35	84
20	0.05	0.05	0.35	0.4	82
20	0.05	0.05	0.4	0.25	86
20	0.05	0.05	0.4	0.35	84
20	0.05	0.05	0.4	0.4	88
20	0.05	0.1	0.25	0.25	88
20	0.05	0.2	0.25	0.25	88
20	0.1	0.05	0.25	0.25	86
20	0.1	0.1	0.25	0.25	88
20	0.1	0.2	0.25	0.25	84
20	0.2	0.05	0.25	0.25	86
20	0.2	0.1	0.25	0.25	86
20	0.2	0.2	0.25	0.25	84

Table No. 6.4: Recognition rates against variation in net parameters using *features collected after Threshold* as input pattern and *Hamming code* as target output.

No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
05	0.05	0.1	0.4	0.25	62
10	0.05	0.1	0.4	0.25	78
15	**0.05**	**0.1**	**0.4**	**0.25**	**90**
20	0.05	0.1	0.4	0.25	62
30	0.05	0.1	0.4	0.25	76
15	0.05	0.1	0.25	0.25	88
15	0.05	0.1	0.25	0.35	88
15	0.05	0.1	0.25	0.4	88
15	0.05	0.1	0.35	0.25	90
15	0.05	0.1	0.35	0.35	90
15	0.05	0.1	0.35	0.4	88
15	0.05	0.1	0.4	0.35	90
15	0.05	0.1	0.4	0.4	90
15	0.05	0.05	0.4	0.25	90
15	0.05	0.2	0.4	0.25	90
15	0.1	0.05	0.4	0.25	90
15	0.1	0.1	0.4	0.25	90
15	0.1	0.2	0.4	0.25	90
15	0.2	0.05	0.4	0.25	90
15	0.2	0.1	0.4	0.25	90
15	0.2	0.2	0.4	0.25	90

Table No.6.5: Recognition rates against variation in net parameters using *features collected after Threshold* as input pattern and *7 bit ASCII code* as target output.

No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
05	0.05	0.1	0.4	0.25	60
05	0.2	0.2	0.4	0.25	66
10	0.05	0.1	0.4	0.25	62
10	0.2	0.2	0.4	0.25	72
15	0.05	0.1	0.4	0.25	72
15	0.2	0.2	0.4	0.25	76
20	0.05	0.1	0.4	0.25	78
20	**0.2**	**0.2**	**0.4**	**0.25**	**80**
30	0.05	0.1	0.4	0.25	70
30	0.2	0.2	0.4	0.25	72
20	0.05	0.1	0.25	0.25	76
20	0.05	0.1	0.25	0.35	74
20	0.05	0.1	0.25	0.4	72
20	0.05	0.1	0.35	0.25	76
20	0.05	0.1	0.35	0.35	74
20	0.05	0.1	0.35	0.4	70
20	0.05	0.1	0.4	0.35	72
20	0.05	0.1	0.4	0.4	76
20	0.05	0.05	0.4	0.25	74
20	0.05	0.2	0.4	0.25	72
20	0.1	0.05	0.4	0.25	76
20	0.1	0.1	0.4	0.25	74
20	0.1	0.2	0.4	0.25	76
20	0.2	0.05	0.4	0.25	72
20	0.2	0.1	0.4	0.25	76

Table No.6.6: Summary of Recognition rates against variation in net parameters

Target Matrix Type	Operation before feature collection	No. of Hidden Units	η_1	η_2	spr.	spread	Recognition (%)
Unit Matrix	No Operation	20	0.05	0.05	0.25	0.25	90
Unit Matrix	Edge Detection	15	0.05	0.05	0.25	0.4	80
Unit Matrix	Threshold	**20**	**0.1**	**0.2**	**0.4**	**0.25**	**92**
Hamming code	Threshold	15	0.05	0.1	0.4	0.25	90
7421 ASCII	Threshold	20	0.2	0.2	0.4	0.25	80

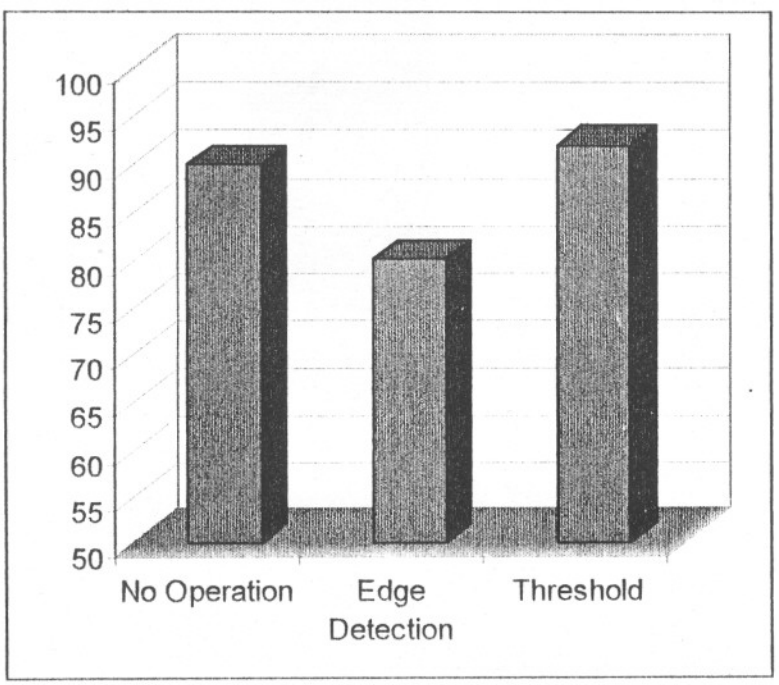


Figure-6.1: Variation in recognition rates for different pre-processing operations.

(Output target was unit matrix.)

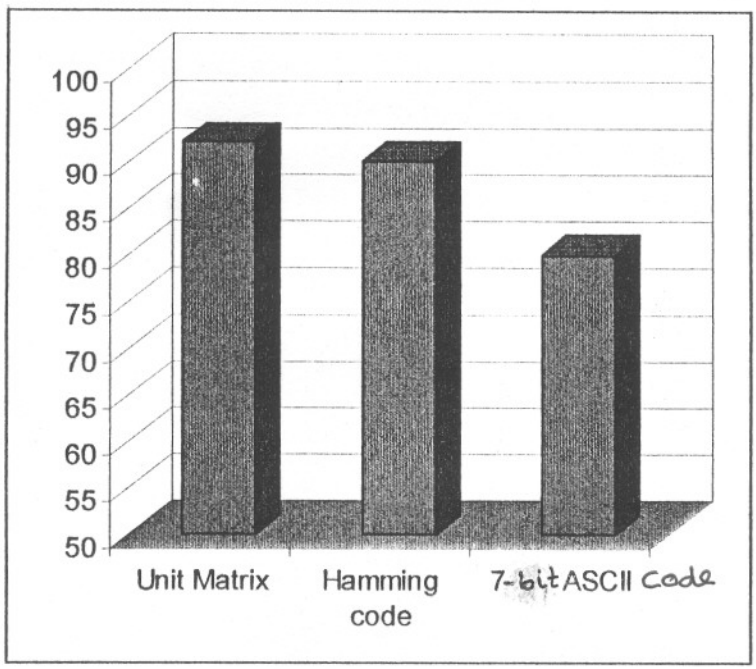


Figure-6.2: Variation in recognition rates for different target output of the net.
(Pre-processing operations was Thresholding)

Table- 6.7: Output Pattern and Errors for best performance of the Neural Net.

Faces	Node0	Node1	Node2	Node3	Node4	Node5	Node6	Node7	Node8	Node9	Errors
Faruk	0.999801	0.000046	0	0.000033	0	0.000052	0.000087	0.000056	0.000019	0	0.000238
Lisa	0.000057	0.999845	0.000042	0.000017	0.000002	0.000001	0.000018	0.000058	0	0.000030	0.000184
Prisly	0	0.000010	0.999834	0.000015	0.000056	0	0	0.000057	0.000043	0.000056	0.000190
Sabina	0.000021	0	0.000016	0.999908	0	0.000002	0	0	0.000015	0	0.000097
Faria	0	0.000004	0.000104	0.000036	0.999837	0.000064	0.000063	0.000059	0	0	0.000225
Masm	0.000034	0	0	0.000012	0.000027	0.999782	0.000019	0.000003	0.000065	0	0.000233
Hasan	0.000096	0.000033	0	0.000007	0.000097	0.000107	0.999745	0.000029	0.000018	0.000137	0.000341
Rahad	0.000042	0.000074	0.000014	0	0.000019	0.000019	0.000019	0.999781	0.000051	0.000007	0.000243
Ivan	0.000019	0	0.000038	0.000038	0	0.000045	0	0.000038	0.999861	0	0.000161
Arif	0	0.000006	0.000045	0.000004	0.000010	0	0.000024	0.000002	0	0.999851	0.000158

6.3 Experimental result with Various Recognition Methods

The objective of this test was to study the result from the recognition of human faces using three recognition methods. The methods were Neural Network, Hamming and Euclidean distance measurement techniques. Features collected from thresholded image were used in this experiment. Like first experiment, in this case also, fifty face images of ten people were included in the test. The detailed results for this experiment is given in table-6.8 and the best performance are marked with gray shading.

Table No. 6.8: Recognition rates for various recognition methods using features collected from thresholded face images.

Name of the Faces	Template Matching by Hamming distance	Template Matching by Euclidean distance	Neural Network
Faruk	5	5	5
Lisa	5	5	5
Prisly	4	4	4
Sabina	5	5	5
Faria	5	5	5
Masum	3	3	3
Hasan	4	5	5
Rahad	4	4	4
Ivan	5	5	5
Arif	5	5	5
Total	45	46	46
Recognition (%)	90	**92**	**92**

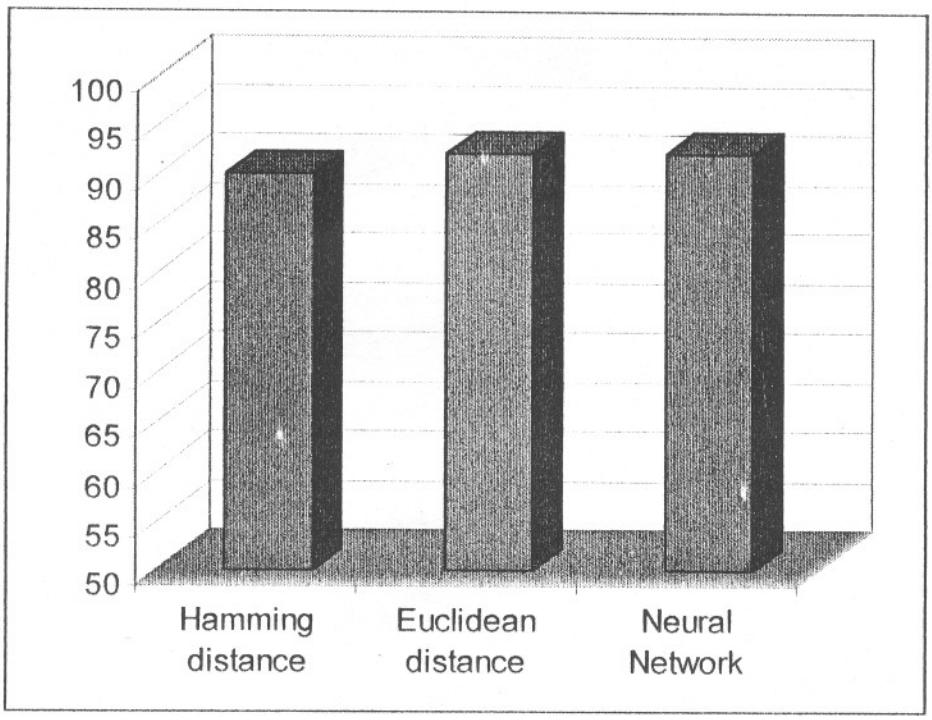


Figure-6.3: Variation in recognition rates for different recognition methods. (Used features was collected after thresholding)

6.4 Experimental result with different features

In this test, three types of features were used to recognize fifty unknown faces of ten different people as above. These were features collected from original face images, edge detected face images and thresholded face images. Each feature type was again tested with variation in the number of feature data used. The Artificial Neural Network with Unit Matrix as target output was used in recognition. The detailed results are shown in table-6.9 and the best performance is marked with gray shading.

Table No. 6.9: Recognition rate against variation in features using ANN with Unit Matrix as target output.

Names of the Faces	Total No. of Used Images	Features direct from photo image	Features after threshold	Features after edge detection
Faruk (a)	5	5	5	5
Lisa (b)	5	5	5	5
Prisly (c)	5	3	4	3
Sabina (d)	5	5	5	5
Faria (e)	5	5	5	5
Masum (f)	5	3	3	3
Hasan (g)	5	5	5	4
Rahad (h)	5	4	4	3
Ivan (i)	5	5	5	5
Arif (j)	5	5	5	2
Total	50	45	46	40
Recognition (%)		90	**92**	80

6.5 Discussion

The various experiments and their results are discussed in this chapter. The results as seen in the above tables can be summarized as below.

- From table-6.1, there are four combinations of net parameters to produce the best performance (92%) using *features collected after threshold* as input pattern and *Unit Matrix* as target output. One of them is as follows: Hidden Unit=20, spr.=0.4, spread=0.25, eta1=0.1 and eta2=0.2
- From table-6.2, there are four combinations of net parameters to produce the best performance (80%) using *features collected after edge detection* as input

pattern and *Unit Matrix* as target output. One of them is as follows: Hidden Unit=15, spr.=0.25, spread=0.4, eta1=0.05 and eta2=0.05.

- From table-6.3, Net parameters, Hidden Unit=20, spr.=0.25, spread=0.25, eta1=0.05 and eta2=0.05 were the best performing parameters (90%) using *Features collected from original face image* as input pattern and *Unit Matrix* as target output.
- From table-6.4, there are thirteen combinations of net parameters to produce the best performance (90%) using *features collected after threshold* as input pattern and *Hamming code* as target output. One of them is as follows: Hidden Unit=15, spr.=0.4, spread=0.25, eta1=0.05 and eta2=0.1.
- From table-6.5, Net parameters, Hidden Unit=20, spr.=0.4, spread=0.25, eta1=0.2 and eta2=0.2 were the best performing parameters (80%) using *features collected after Threshold* as input pattern and *7-bit ASCII code* as target output.
- From the summary table no.-6.6, the best performance (92%) includes *features collected after threshold* as input pattern and *Unit Matrix* as target output. Net parameters are Hidden Unit=20, spr.=0.4, spread=0.25, eta1=0.1 and eta2=0.2
- From table 6.7, the best performing net has also an average error of 0.000207 for *Unit Matrix* as target output. This was one source of recognition error.
- From table-6.8, Euclidean distance and Neural Network perform nearly the same, Neural Network = 92%, Euclidean=92% and Hamming = 90%.
- From table-6.9, for same set of network parameters, *features collected after threshold* proves better among the others.

Chapter-7

Discussion and Conclusion

DISCUSSION AND CONCLUSION

7.1. Discussion

In this research, an attempt was made to recognize Human Faces using different features and methods. Three recognition methods namely Neural Networks, Hamming and Euclidean distance measurements were used with emphasis on Neural Networks. The formation and performance of artificial neural network was studied in chapter 3. A neural network was trained and tested for this recognition scheme. The behavior of the net with variations in its different parameters was studied. Results are shown in table-6.1 to 6.5. The parameters were two spread parameters *spr* and *spread*, two learning rates η_1 and η_2 and number of *hidden layers* used. As seen in table-6.1, the net gives better performance for the following values:

spr = 0.4, spread = 0.25, $\eta_1 = 0.1$, $\eta_2 = 0.2$ *and No. of Hidden Layer = 20*

For each set of parameters, the network cycles through the input patterns for 30,000 times. The network input pattern and output target pattern effects the values of the network parameters and hence the success rate. Various combinations of these two types of patterns were used to find the best combination of patterns and parameters. The results in tables 6.1 to 6.5 shows that the same success rate occurred for a number of combinations. The first occurrence of highest success rate was considered to make a summary table for behavior study of this network. As seen in table-6.7, the overall highest success rate achieved for this network is 92% and this rate is for Unit matrix as output target and features

collected after threshold as input pattern. During all these tests, It was also found that success rate also varies with initial random weights.

Three recognition methods namely, Neural Network, Hamming and Euclidean distance measurement have been discussed in chapter-3. Comparative results were shown in table-6.8. As seen in table-6.8, the two methods Euclidean distance measurement and neural network produce the same result, but Hamming distance measurement produces slightly less success. The distance measurements were very simple in computation but neural networks were somewhat more complex. So the simple decision is that the Euclidean distance measurement should be accepted in recognition of human face recognition. But today's recognition systems avoid the use of this method due to its less possibility of further improvement. On the other side, Neural Networks are rapid developing recognition tool and there is strong possibility to improve recognition result using different pattern and parameter setting. In this prospect, the rest of the tests were carried out using Artificial Neural Network.

Three types of features were collected from the face image. They were

I) Features extracted direct from the bitmap of the face image.

II) Features extracted from the thresholded face image.

III) Features extracted from the edge detected face image.

These feature extraction methods have been discussed in chapter-4. The result of the experiment with various features is given in table-6.9. The result shows that

the features extracted from thresholded image produces better results (92%). The effect of thresholding have the effect of filtering and in our experiment, a threshold of 200 was used. The resulting thresholded face images are given in Appendix-D. It was seen from these images that the regions of lip, eyes, nose, hair and face boundary are clearly marked. From Appendix-C, it is also seen that the edge-detected face has the same effect. The difference is that in the former image, the locations are marked by less data. But the latter has a gradient of gray levels from the edges. So with edge detection, the original image information was lost and the localization of face regions was not better. For these two simultaneous effects, features after edge detection produce less success rate. On the other hand, though the thresholding method also losses original image information, it locates face organs clearly and correctly. So the localization of face organs plays an important roll in searching a better feature.

In this research 100% percent recognition was not possible in any case. Errors ranging from 8% to 10% were observed in finally accepted results. The following were major as the source of this error. During photograph the width and height (dimension) of face was not equal in all scenes (Appendix-C). On the other hand, the face part was separated manually from the photo by Photofinish software. So it was not possible to separate the faces with accurately equal dimension. In feature extraction, a sum and average algorithm was adapted for each 10x10 matrix of image data. For each face image a 10x9 feature data was extracted. So a total of 10x10x10x9 data was included in this process. Thus the image with higher

dimension losses some data in the feature extraction process. Instead of manual separation, using a face separation algorithm and a scaling algorithm may eliminate this problem. As seen in table-6.7, the training phase of neural network produces some errors. These errors were effective in recognition using neural network. Proper settings of parameter, pattern and random weight may reduce the errors.

7.2 Conclusion

A face recognition system must be able to recognize a face in many different imaging situations. The appearance of a face in a 2D image is not only influenced by its identity but also by other variables such as brightness, dimension and direction of face in photo. This thesis has described a method for recognizing the face despite these variations.

1. From our study, it is now clear that the use of edge detection in recognition of human face is less suitable than other methods. The features collected from thresholded face are proved stronger in recognition of human faces.

2. Neural Network and Template Matching by Hamming and Euclidean distance measurement was used as recognition tools. The recognition of human faces is not so dependent on these methods. But there are various opportunities to moderate the Neural Networks for better recognition result. So the Neural Network may be accepted as a better tool then the others used in this research.

3. The recognition results are satisfactory and the developed system may be used in any human or object identification system where reference database is not so large.

7.3 Suggestions for future work

1. This research finds best features by sum and average a block of thresholded image. But with this feature also, there was 8% error in the recognition result that may not be tolerable in some application. So future research workers should find much better feature extraction method for recognition.

2. All the experiments were done using only faces of ten people. The recognition rate may decrease with large reference database. So future work should give emphasis on the large size of reference database without any compromise to recognition rate.

3. As discussed in section-7.1, for manual separation of face parts from photographs make differences in the dimensions of faces. So future research workers should include an automatic face extraction system including face-scaling facility to eliminate this problem.

4. In solution to the above problems Motion Field Histogram[40], Eigenspace Modeling[41] or other suitable features may be used and Convolutional Neural

Network [42], Hidden Markov Model (HMM) or any other suitable recognition tool should be used.

References

REFERENCES

1. M. Kirby and L. Sirovich, *Application of the karhunen-loeve procedure for the characterization of human faces*, *IEEE Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, 1990.

2. M. Turk and A. Pentland, *Eigenfaces for recognition*,*J. Cog. Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

3. Kah-Kay Sung, Tomaso Poggio, *Example-Based learning for View-Based Human Face Detection*, Proceedings of 23rd Image Understanding Workshop, Menlonay, California, USA, 1994.

4. Tony S. Jebara , *3D Pose Estimation and Normalization for Face Recognition*, MIT Media Laboratory Home page, http://www.media.mit.edu, Cambridge, MA 02139, 2002.

5. R. et al. Chellappa. *Human and machine recognition of faces: A survey. Technical Report CAR-TR-731*, Univeristy of Maryland Computer Vision Laboratory, 1994.

6. T. Kanade. *Picture Processing System by Computer Complex and Recognition of Human Faces, PhD. Thesis*. PhD thesis, Kyoto University, Japan, 1973.

7. I. Craw, D. Tock, and A. Bennett. Finding face features. In *Second European Conference on Computer Vision*, pages 92-96, 1992.

8. M. Nixon. *Eye spacing measurement for facial recognition.* In SPIE Proceedings, pages 279-285, 1985.

9. I.J. Cox, J. Ghosn, and P.N. Yianilos. *Feature-based face recognition using mixture-distance.* Technical Report TR-95-09, NEC Research Institute, 1995.

10. N. Roeder and X. Li. *Experiments in analyzing the accuracy of facial feature detection.* In *Vision Interface '95*, pages 8-16, 1995.

11. R. Rao and D. Ballard. *Natural basis functions and topographics memory for face recognition.* In International Joint Conference on Artificial Intelligence, pages 10-17, 1995.

12. B.S. Manjunath, R. Chellappa, and C.V.D. Malsburg. *A feature based approach to face recognition.* In Proceedings, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 373-378, 1992.

13. M.A. Turk and A.P. Pentland, *Face recognition using eigenfaces.* In IEEE Computer Society Conference Computer Vision and Pattern Recognition, pages 586-591, 1991.

14. S. Akamatsu, T. Saski, H. Fukamachi, N. Masui, and Y. Suenaga. *Robustface identification scheme.* In Proceedings of SPIE - The International Society for Optical Engineering, pages 71-84, 1992.

15. B. Moghaddam, C. Nastar, and A. Pentland. Bayesian *Face Recognition using Deformable Intensity Surfaces.* Technical Report 371, MIT Media

Laboratory Perceptual Computing Section, 1996.

16. Maria Petrou, Panagiota Bosdogianni, *Image Processing – The Fundamentals*, John Wiley & Sons Ltd, England, 1999.

17. Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Addision-Wesely, 1993.

18. Fu, Limin, *Neural Networks in Computer Intelligence*, McGraw-Hill International Edition, 1994.

19. Anil K. Jain, *Fundamental of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

20. A.Maher Sid-Ahmed, *Image Processing-Theory, Algorithms, and Architecture*, McGraw-Hill, Inc., 1995.

21. Efford Nick, *Digital Image Processing-a practical introduction using Java,* Pearson Education Limited, England, 2000.

22. Philips Dwayne, *Image Processing*, BPB Publications, B-14, Connaught Place, New Delhi-11001, 1995.

23. Lee Nelson, *Commercialising Robust Face Recognition Capability, Polariod and Quebec Vision Start-up*, Advance Imaging, Feb. 1998, pp. 72-79.

24. Leigh Strither-Vien, Mugshot Recognition Meets Witness Composite Sketches in L.A., Advance Imaging, Jan. 1998, pp. 22.

25. H. A. Rowley, S. Baluja, T. Kanade, *Neural Network Based Face Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, pp. 23-38, 1998.

26. Bose N.K., P. Liang, *Neural Network Fundamentals with Graphs, Algorithms and Applications*, Tata McGraw-Hill, Inc., 1998.

27. Steve Lawrence, C.L. Giles, A.C. Tsoi, and A.D. Back. *Face recognition: A hybrid neural network approach*. Technical Report UMIACS-TR-96-16, Institute for Advanced Computer Studies, University of Maryland, http://www.neci.nj.nec.com , 1996.

28. Gyu-tae Park, Zeungnam Bein, *Neural network-based fuzzy observer with application to facial analysis*, Pattern Recognition Letters 21 (2000) 93-105, www.elsevier.nl/locate/patrec, 2000.

29. Rietman Edward, *Exploring Parallel Processing*, Windcrest Books, U.S.A., 1990.

30. Adam Blum, *Neural Networks in c++*, John Wiley & Sons Ltd.

31. R. Beale and T. Jackson, *Neural Computing: An Introduction*, IOP Publishing Ltd. 1990.

32. Md. Farukuzzaman Khan, *Computer Recognition of Bangla Speech*, M.Phil. Thesis, Islamic University, Kushtia, Bangladesh, 2003.

33. Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, MIT press, Cambridge, MA, 1986.

34. Widrow, B. and Hoff, M.E., *Associative Storage and Retrieval of Digital Information in Networks of Adaptive Neurons*, Biological Prototypes and Synthetic Systems, Vol. 1, Plelnum Press,, New York, 1962.

35. Widrow, B. and Hoff, M.E., *Adaptive Switching Circuits*, IRE WESCON, Convention Record IRE, New York, pp 96-104, 1960.

36. LeCun, Y., *Learning Processes in an Asymmetric Threshold Network*, Disordered Systems and Biological Organization, NATO ASI Series F, Vol. 20, Springer-Verlag, Berlin, 1986.

37. Stamatios V. Kartalopoulos, *Understanding Neural Networks and Fuzzy Logic: Basic concepts and Application*, IEEE, New Delhi, 1996.

38. Rakib Ahmed, *Pattern Recognition by Neural Network*, M.Sc. Thesis, Department of Applied Physics & Electronics, Rajshahi University, Bangladesh, 1994.

39. Yuille, D. Cohen, and P. Hallinan. *Feature extraction from faces using deformable templates.* In IEEE Computer Society Conference on Computer Vision and Templates, pages 104-109, 1989.

40. Tanzeem Choudhury, Alex Pentland, *Motion Field Histograms for Robust Modeling of Facial Expressions*, MIT Media Laboratory Home page,

http://www.media.mit.edu, Cambridge, MA 02139, 2002.

41. Tanzeem Choudhury, Brian Clarkson, Tony Jebara, Alex Pentland, *Multimodal Person Recognition using Unconstrained Audio and Video*, MIT Media Laboratory Home page, http://www.media.mit.edu, Cambridge, MA 02139, 2002.

42. Steve Lawrence, C. Lee Giles, A h Chung Tsoi and Andrew D. Back, *Face Recognition: A Convolutional Neural Network Approach*, IEEE Transaction on Neural Networks, Special issue on Neural Networks and Pattern Recognition, Volume 8, Number 1, pp. 98-113, 1997.

Apendices

APPENDIX-A

The prototypes of functions used in the human face recognition system

Functions	Header Files
void direct(FILE *in, FILE *out)	feature.h
void edge(FILE *f1, FILE *f2)	feature.h
void threshold(FILE *in, FILE *out)	feature.h
void bmpread(FILE *f1)	feature.h
void min_image(FILE *in, FILE *out)	feature.h
int neu_test(char *filen, int file_no)	neural.h
void neural(int type1)	neural.h
double uc_dis(float xdata[], float gdata[], int n)	rec.h
double ham_dis(float xdata[], float gdata[], int n)	rec.h
void rec_result(int type1)	Main Program
int recognition(int file_no, char *filename, int type1)	Main Program
void process_image(int type1)	Main Program

APPENDIX B

BitMap Image File Format

Bmp file format

Windows bitmap files are stored in a device-independent bitmap (DIB) format that allows windows to display the bitmap on any type of display device. The term "device independent" means that the bitmap specifies pixel color in a form independent of the method used by a display to represent color. The default filename extension of a Windows DIB file is BMP.

Bitmap file structures: Windows bitmap files contain the following sequence of data structures,

A file header

A bitmap information header

A color table

An array of bites that defines the bitmap bits.

The file has the following form:

```
BITMAPFILEHEADER bmpFileHeader;
BITMAPINFOHEADER bmpInfoHeader;
RGBQUAD aColors[ ];
BYTE aBitmapBits[ ];
```

Bitmap File Header: The bitmap-file header contains information about the type, size, and layout of a device-independent bitmap file. The header is defined as a BITMAPFILEHEADER data structure.

```
typedef struct tagBITMAPFILEHEADER{
        unsigned int bfType;
        unsigned long bfSize;
        unsigned int bfReversed1;
        unsigned int bfReversed2;
        unsigned long bfOffbits;
}BITMAPFILEHEADER;
```

The following table describes the fields of the structure,

Field	Description
bfType	Specifies the file type. This member must be BM(WORD value 0×4D42)
bfSize	Specifies the file size in bytes.
bfReserved1	Reserved. Must be set to zero.
bfReserved2	Reserved. Must be set to zero.
bfOffBits	Specifies the byte offset from the BITMAPFILEHEADER structure to the actual bitmap data in the file.

Bitmap Information Header: The bitmap information header, defined as compression type, and color format for the bitmap.

```
typedef strutct tagBITMAPINFOHESDER{
        unsigned long biSize;
        unsigned long biWidth;
        unsigned long biHeight;
        unsigned int biPlanes;
        unsigned int biBitCount;
        unsigned long biCompression;
        unsigned long biSizeImage;
        unsigned long biXPelsPerMeter;
        unsigned long biYpelsPerMeter;
        unsigned long biClrUsed;
        unsigned long biClrImportant;
}BITMAPINFOHEADER;
```

The following table describes the fields of the structure,

Field	Description
biSize	Specifies the number of bytes required by the BITMAPINFOHEADER structure.
biWidth	Specifies the width of the bitmap, in pixels.
biHeight	Specifies the height of the bitmap, in pixels.

biPlanes Specifies the number of planes for the target device. This field must be set to 1.

biBitCount Specifies the number of bits per pixel. It can have any of the following values,

Value	Meaning
1	Bitmap is monochrome and the color table contains two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the color table. If the bit is set, the pixel has the color of the second entry in the table.
4	Bitmap has a maximum of 16 colors. Each pixel in the bitmap is represented by a 4-bit index into the color table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8	Bitmap has a maximum of 256 colors. Each pixel in the bitmap is represented by a 1-byte index into the color table. For example, if the first byte in the bitmap is 0xlF, the first pixel has the color of the thirty-second table entry.
24	Bitmap has a maximum of 2^24 colors. The bmiColors (or bmciColors) member is NULL, and each 3-byte sequence in the bitmap array represents the relative intensities of red green and blue respectively, for a pixel.

biCompression Specifies the type of compression for a compressed bitmap. It can be one of the following values.

Value	Meaning
BI_RGB	Specifies the bitmap is not compressed.
BI_RLE8	Specifies a run-length-encoding format with 8 bits per pixel.
BI_RLE4	Specifies a run-length-encoding format with 4 bits per pixel.

biSizeImage Specifies the image size in bytes.

biXPelsPerMeter Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap.

biYPelsPerMeter Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.

biClrUsed Specifies the number of color indexes in the color table actually used by the bitmap. Possible values for this field are

Value	Result
0	Bitmapuses the maximum number of colors corresponding to the value of the bitBitCount field.
Less than 24	Bitmap uses the actual number of colors that the graphics engine or device driver will access.
24	The size of the color table is used to optimize the performance of windows color pallate.

BiClrImportant Specifies the number of color indexes that are considered important for displaying the bitmap. If this field is set to zero, then all the colors are important.

Bitmap Color Table

Thecolor table, defined as an array of RGBQUAD structures, contains as many elements as there are colors in the bitmap.

```
tydef struct tagRGBQUAD{
                    unsigned char rgbBlue;
                    unsigned char rgbGreen;
                    unsigned char rgbred;
                    unsigned char rgbReserved;
                   }RGBQUAD;
```

The following table describes the fields of the structure,

Field	Description
rgbBlue	Specifies the intensity of blue in the color.
rgbGreen	Specifies the intensity of green in the color.
rgbRed	Specifies the intensity of red in the color.
rRgbReserved	Not used. Must be set to zero.

The color table is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area. The colors in the table should appear in order of importance. This helps a display driver render a bitmap on a device that cannot display as many colors as there are in the bitmap. If the DIB is in windows version 3.0 or later format, the driver can use the biClrImportant member of the BITMAPINFOHEADER structure to determine which colors are important.

Bitmap Example:

The following example is a text dump of a 16-color bitmap (4 bits per pixel):

```
Win3DIBFile
        BitmapFileHeader
            Type            19778
            Size            3118
           Reserved1        0
           Reserved2        0
           Offset Bits      118
        BitmapInfoHeader
            Size            40
            Weidth          80
            Height          75
            Planes          1
```

BitCount	4
Compression	0
SizeImage	3000
XpelsPerMeter	0
YpelsPerMeter	0
ColorsUsed	16
ColorsImportant	16

Win3ColorTable

	Blue	Green	Red	Unused
[00000000]	84	252	84	0
[00000001]	252	252	84	0
[00000002]	84	84	252	0
[00000003]	252	84	252	0
[00000004]	84	252	252	0
[00000005]	252	252	252	0
[00000006]	0	0	0	0
[00000007]	168	0	0	0
[00000008]	0	168	0	0
[00000009]	168	168	0	0
[0000000A]	0	0	168	0
[0000000B]	168	0	168	0
[0000000C]	0	168	168	0
[0000000D]	168	168	168	0
[0000000E]	84	84	84	0
[0000000F]	252	84	84	0

.

.

. Image

.

.

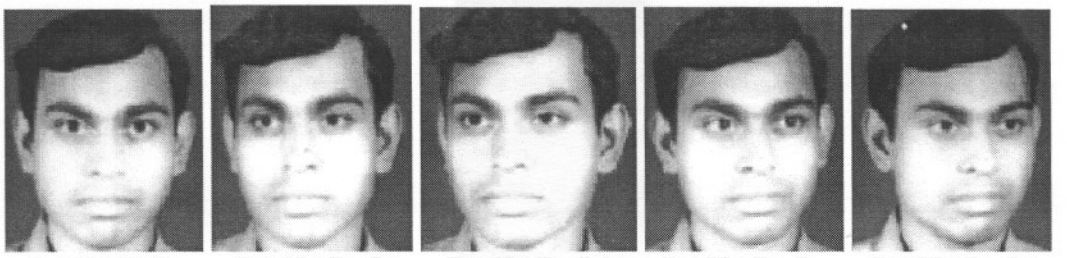
. Bitmap data

APPENDIX-C

Original Face Image

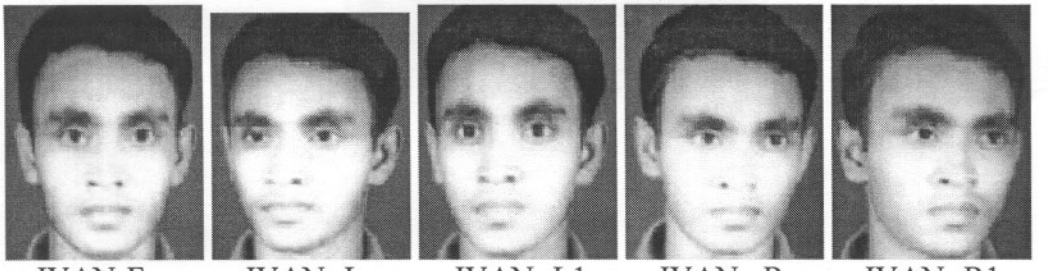
MASUM-L MASUM-L1 MASUM –R MASUM -R1 MASUM –F

HASAN-F HASAN -L HASAN -L1 HASAN –R HASAN -R1

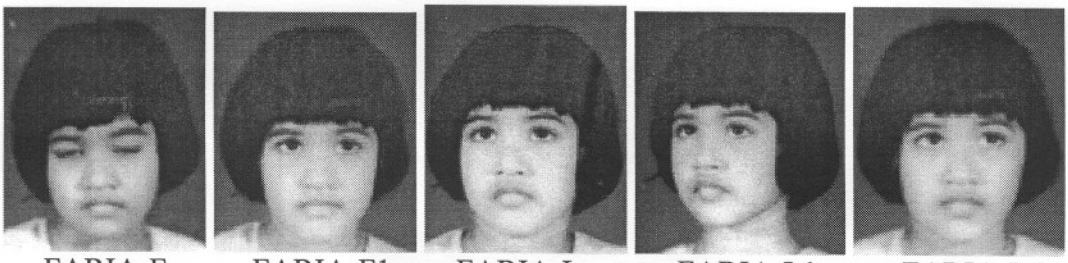


RAHAD-F RAHAD -L RAHAD -L1 RAHAD -R RAHAD -R1

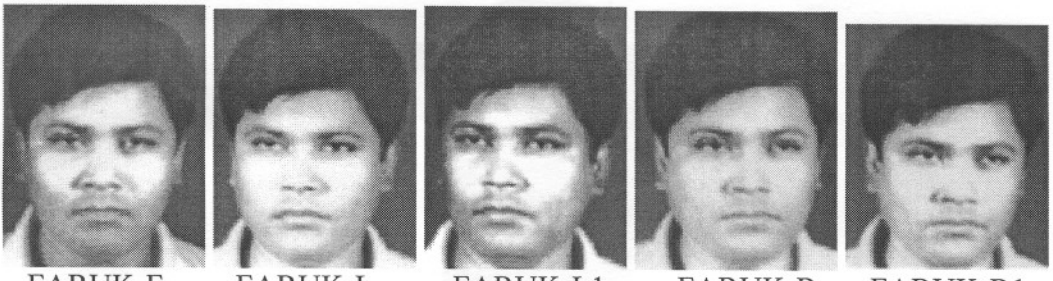
ARIF-F ARIF -L ARIF -L1 ARIF –R ARIF -R1



IVAN-F IVAN -L IVAN -L1 IVAN –R IVAN -R1

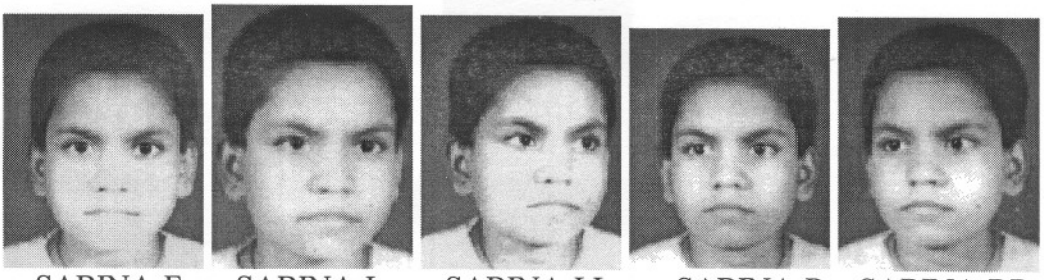


FARIA-F FARIA-F1 FARIA-L FARIA-L1 FARIA-R



FARUK-F FARUK-L FARUK-L1 FARUK-R FARUK-R1

LISA-F LISA-L LISA-L1 LISA-R LISA-R1

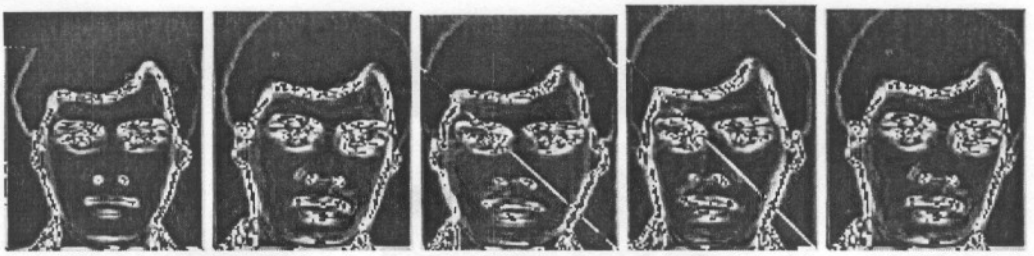


SABINA-F SABINA-L SABINA-LL SABINA-R SABINA-RR

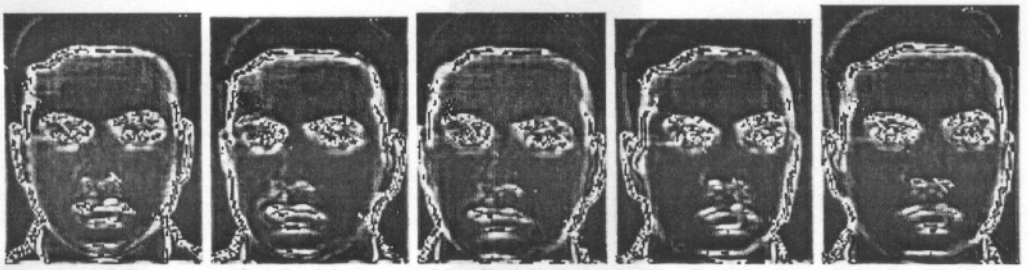
PRISLEY-F PRISLEY-L PRISLEY-LL PRISLEY-R PRISLEY-RR

APPENDIX-D

Edge Detected Face Images



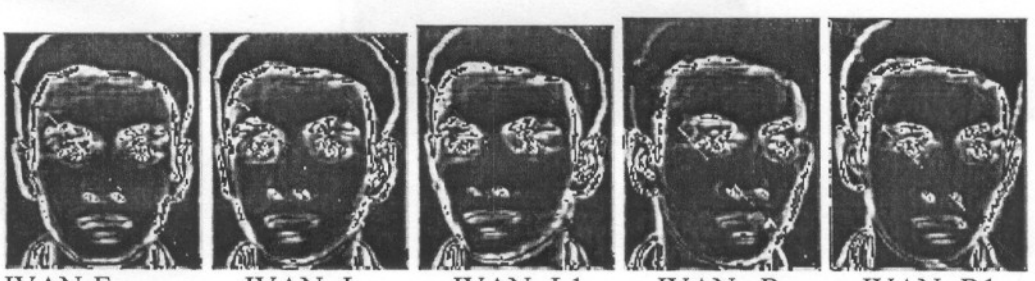
MASUM-L MASUM-L1 MASUM –R MASUM -R1 MASUM –F



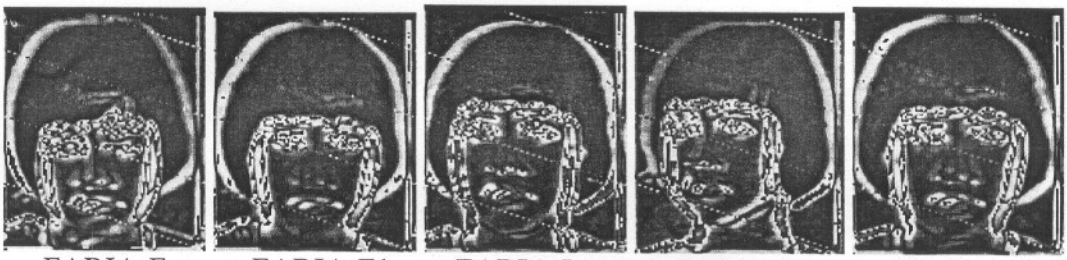
HASAN-F HASAN -L HASAN -L1 HASAN –R HASAN -R1

RAHAD-F RAHAD -L RAHAD -L1 RAHAD -R RAHAD -R1

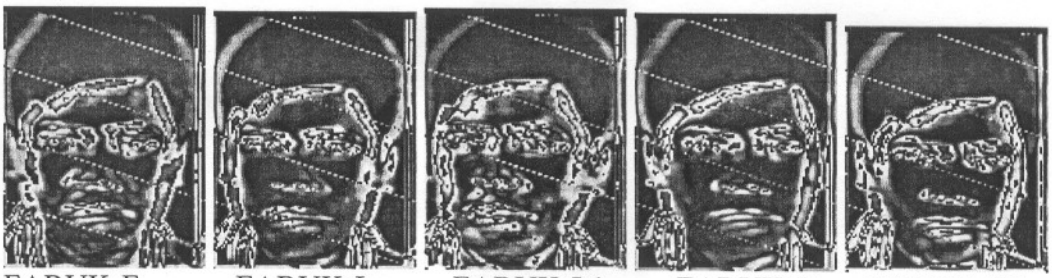
ARIF-F ARIF -L ARIF -L1 ARIF –R ARIF -R1



IVAN-F IVAN -L IVAN -L1 IVAN –R IVAN -R1

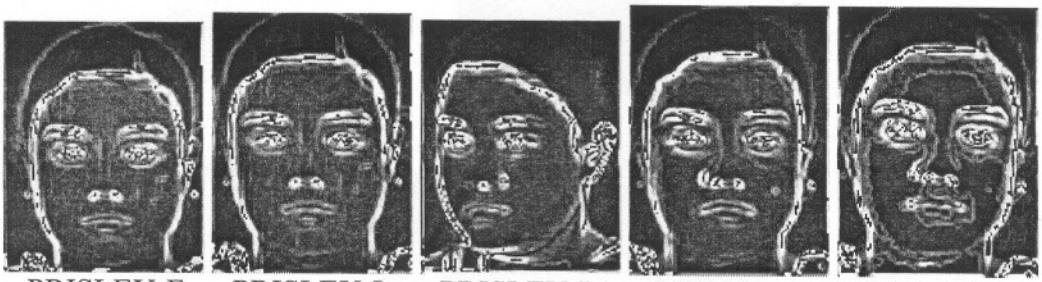


FARIA-F FARIA-F1 FARIA-L FARIA-L1 FARIA-R

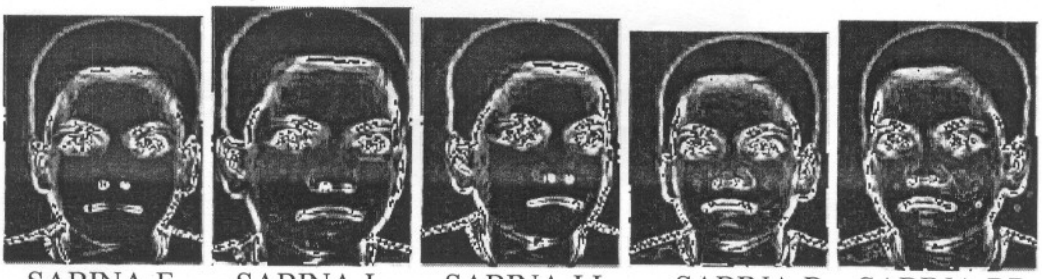


FARUK-F FARUK-L FARUK-L1 FARUK-R FARUK-R1

LISA-F LISA-L LISA-L1 LISA-R LISA-R1



PRISLEY-F PRISLEY-L PRISLEY-L1 PRISLEY-R PRISLEY-R1

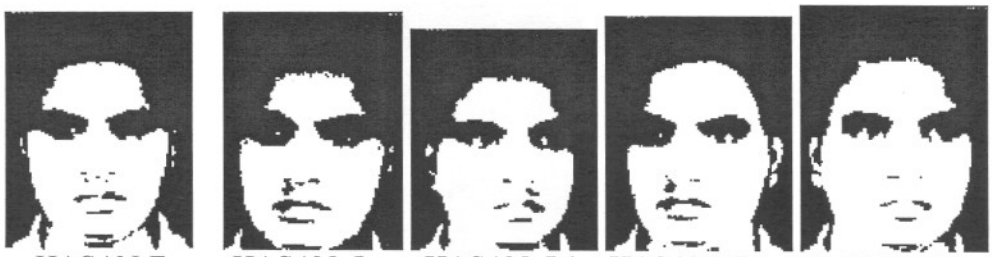


SABINA-F SABINA-L SABINA-LL SABINA-R SABINA-RR

APPENDIX-E

Face Images After Thresholding

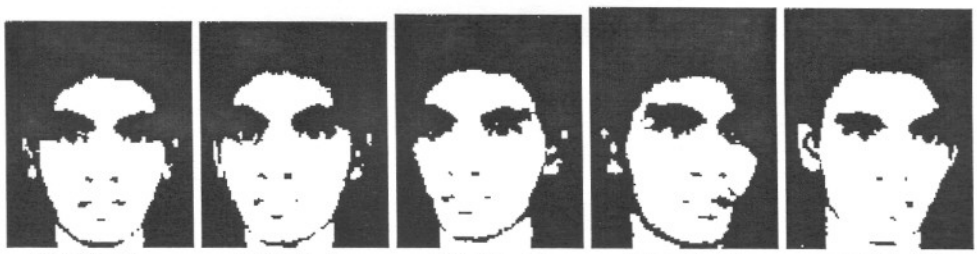
MASUM-L MASUM-L1 MASUM –R MASUM -R1 MASUM –F



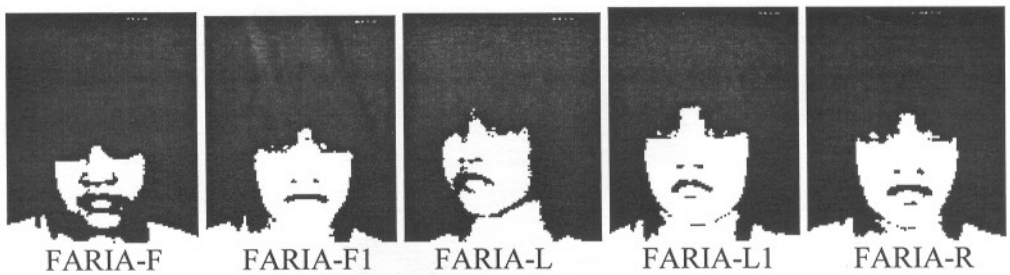
HASAN-F HASAN -L HASAN -L1 HASAN –R HASAN -R1

RAHAD-F RAHAD -L RAHAD -L1 RAHAD –R RAHAD -R1

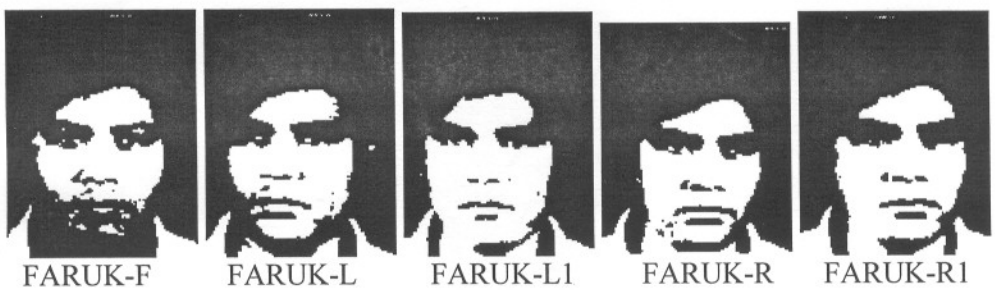
ARIF-F ARIF -L ARIF -L1 ARIF –R ARIF -R1



IVAN-F IVAN -L IVAN -L1 IVAN –R IVAN -R1

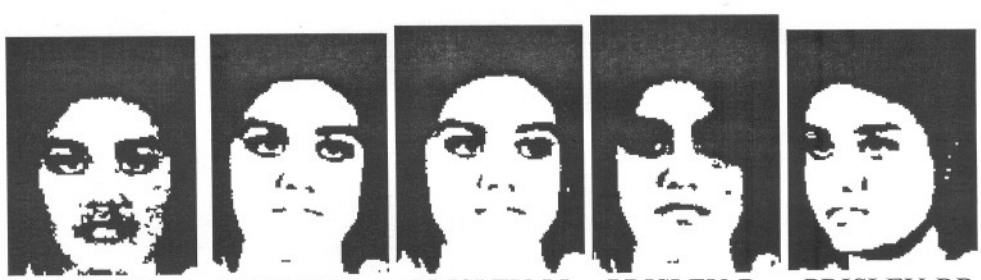


FARIA-F FARIA-F1 FARIA-L FARIA-L1 FARIA-R



FARUK-F FARUK-L FARUK-L1 FARUK-R FARUK-R1

LISA-F LISA-L LISA-L1 LISA-R LISA-R1



PRISLEY-F PRISLEY-L PRISLEY-LL PRISLEY-R PRISLEY-RR

SABINA-F SABINA-L SABINA-LL SABINA-R SABINA-RR

Abbreviations used in Appendices

- F: Face Images with 0^0 Rotation (Front Pose)
- R: Face Images with 22.5^0-Right Rotation (Right Pose)
- R1: Face Images with 45^0-Right Rotation (Right Pose)
- L: Face Images with 22.5^0-Left Rotation (Left Pose)
- L1: Face Images with 45^0-Left Rotation (Left Pose)