

FACE DETECTION FOR TWO DIMENSIONAL IMAGES USING GABOR FEATURE EXTRACTION AND NEURAL NETWORKS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

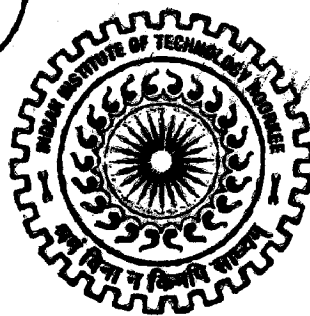
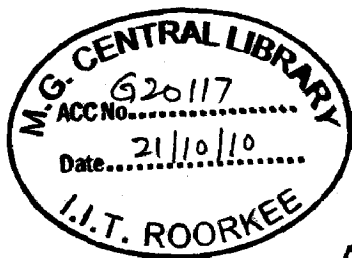
in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Control and Guidance)

By

S.LINGAMAIAH



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE**

ROORKEE -247 667 (INDIA)

JUNE, 2010

CANDIDATE'S DECLARATION

I hereby declare that the work, which is presented in this dissertation report entitled, **“Face Detection for two Dimensional images using Gabor feature extraction and Neural Networks”** towards the partial fulfillment of the requirements for the award of the degree of **Master of Technology** with specialization in **Control & Guidance**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee (India) is an authentic record of my own work carried out during the period from July 2009 to June 2010, under the guidance of **Dr.M.J.Nigam, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.**

I have not submitted the matter embodied in this dissertation for the award of any other Degree or Diploma.

Date: 28/06/10

Place: Roorkee

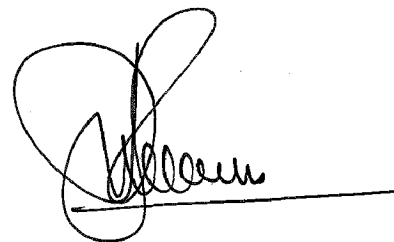
S. Lingamaiah
S.LINGAMAIH

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 28.06.2010

Place: Roorkee



Dr. M. J. NIGAM,

Professor, E&C Department,

IIT Roorkee,

Roorkee -247 667 (India).

ACKNOWLEDGEMENTS

I would like to extend gratitude and indebtedness to my guide, **Dr. M. J. NIGAM** for his guidance, attention and constant encouragement that inspired me throughout my dissertation work.

I would also like to thank the Lab staff of Control Systems Lab, Department of Electronics and Communication Engineering, IIT Roorkee for providing necessary facilities.

I gratefully acknowledge my sincere thanks to my family members for their inspirational impetus and moral support during course of this work.

I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample morale support and valuable suggestions. Finally, I would like to extend my gratitude to all those persons who directly or indirectly contributed towards this work.

S.LINGAMIAH

Images containing faces are essential to intelligent vision-based human computer interaction, and research efforts in face processing include face recognition, face tracking, pose estimation, and expression recognition. Given a single image, the goal of face detection is to identify all image regions which contain a face regardless of its position, orientation, and lighting conditions. Such a problem is challenging because faces are nonrigid and have a high degree of variability in size, shape, color, and texture. The faces in a given image are detected by using a neural network and Gabor filter features.

Numerous techniques have been developed to detect faces in a single image and are classified into four categories; Knowledge-based methods, Feature invariant approaches, Template matching methods and Appearance-based methods. In case of Appearance-based method the models are learned from a set of training images which should capture or identify the variations of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection.

The dissertation work consists of two parts; Image processing and Neural Network. The given image is processed using Gabor filters for extracting the facial features by using the image processing tool box. The extracted facial features are applied to the feed forward neural network by using neural network tool box. Considering the desirable characteristics of spatial locality and orientation selectivity of the Gabor filter, the filters had been designed for extracting facial features from the local image. The goal of facial feature extraction is to detect the presence and location of features such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc. The feature vector based on Gabor filters is used as the input of the classifier, which is a Feed Forward neural network. The given image convolved with Gabor filters by multiplying the image by Gabor filters in frequency domain.

Feature extraction algorithm for the method has two main steps; feature point localization and feature vector computation. Feature vectors are extracted from points with high information content on the face image. In most feature-based methods, facial features are assumed to be the eyes, nose and mouth. From the responses of the face image to Gabor filters, peaks are found by searching the locations in a window W_0 of size $W \times W$. Feature vectors are generated at the feature Points as a composition of Gabor wavelet transform

coefficients. This architecture was implemented using Matlab in a graphical environment allowing face detection in a database. It has been evaluated using the training data and test data of 150 images containing faces and non faces, on this test set I obtained a good detection which is shown in chapter 6.

Face detection and recognition has many applications in a variety of fields such as security system, videoconferencing and identification. The objective of this work is to implement a classifier based on neural networks and Gabor feature extraction for face detection. The ANN is used to classify face and non-face patterns.

LIST OF FIGURES

Figure No.	Figure Caption	Page No.
2.1	A typical face used in knowledge based top down method	5
2.2	Face and non face clusters	10
3.1	Gabor filter kernels with values of the wavelength parameter of 5, 10 and 15	14
3.2	Gabor filter kernels with values of the orientation parameter of 0, 45 and 90	15
3.3	Gabor filter kernels with values of the phase offset parameter of 0, 180, -90 and 90 degrees	16
3.4	Gabor filter kernels with values of the aspect ratio parameter of 0.5 and 1	16
3.5	Gabor filter kernels with values of the bandwidth parameter of 0.5, 1, and 2	17
3.6	Gabor filters correspond to 5 spatial frequencies and 8 orientations	21
3.7	Facial feature points found as the high-energized points of Gabor wavelet responses	22
3.8	Flowchart of the feature extraction stage of the facial images	23
3.9	Finding the center of each region	25
3.10	Filtering above pattern for values above threshold (xy)	25
3.11	Final Result of the detection	25
4.1	Single Neuron without Bias	28
4.2	Single Neuron with Bias	29
4.3	Multi-Layer Artificial Neural Network	30
4.4	Threshold Function	32
4.5	Linear Transfer Function	33
4.6	Sigmoid Transfer Function	34
4.7	Single Layer Network	35
4.8	Multi Layer Network	35
4.9	The neuron of supervised training	41
4.10	Architecture of proposed system	42

5.1	Main GUI executed by main.m	44
5.2	Training process	48
6.1	The training performance with learning rate 0.8 and scaled conjugate gradient training algorithm	51
6.2	The training performance with learning rate 0.1 and scaled conjugate gradient training algorithm	52
6.3	The training performance for the scaled conjugate gradient training algorithm with 0.8 learning rate, which shows that the target is reached at 127 Epochs.	54
6.4	The training performance for the Resilient Backpropagation training algorithm with 0.8 learning rate, which shows that the target is reached at 73 Epochs	55
6.5	The training performance for the Gradient descent with adaptive learning rate Backpropagation training algorithm with 0.8 learning rate, which shows that the target is not reached for 5000 Epochs	56
6.6	The training performance for the Gradient descent with momentum and adaptive learning rate Backpropagation training algorithm with 0.8 learning rate, which shows that the target is not reached for 5000 Epochs	57
6.7	Output obtained from the system which is taken from the digital camera	58
6.8	Output obtained from the system which is collected from the internet	59
6.9	Output obtained from the system	60
6.10	Output obtained from the system	61
6.11	Output obtained from the system in which false detections are present.	62
6.12	Output obtained from the existing method	63
6.13	Output obtained from the existing method with false detection	64

Candidate’s Declaration and Certificate.....	i
Acknowledgements.....	ii
Abstract.....	iii
List of Figures.....	v
Content.....	vii
1. Introduction.....	1
2. Face Detection in Image.....	3
2.1. Knowledge - Based Methods.....	4
2.2. Feature – Based Methods.....	6
2.2.1 Facial Features.....	6
2.2.2 Texture.....	7
2.2.3 Multiple Features.....	7
2.3. Template matching.....	8
2.3.1 Predefined Templates.....	8
2.4. Appearance Based Methods.....	9
2.4.1 Eigenfaces.....	9
2.4.2 Distribution Based Methods.....	10
2.4.3 Support Vector Machines.....	11
3. Digital Image Processing.....	12
3.1. Image Enhancement.....	12
3.1.1. Point Operation.....	12
3.1.2. Mask Operation.....	12
3.2. Gabor filter.....	12
3.2.1. Explanation of Parameters.....	13
3.3. 2D Gabor Wavelet Representation of Faces.....	20
3.3.1. Feature Extraction.....	21
3.3.2. Feature Point Localization.....	21
3.3.3. Feature Vector Generation.....	22
4. Neural Networks.....	26
4.1. Advantages of Artificial Neural Networks.....	27
4.2. Artificial Neural Networks for Image Processing.....	27
4.3. Feed-Forward Neural Network Model.....	28

4.4. Network Model for the System.....	29
4.5. Transfer Functions.....	32
4.6. Network Layers.....	34
4.7. Neural network architecture designing.....	36
4.8. Platform.....	37
4.9. Optimization of Neural Network Parameters.....	37
4.9.1. Weight Initialization.....	37
4.9.2. Weight Adaptation.....	38
4.9.3. Learning Constant.....	38
4.9.4. Inputs and Outputs.....	39
4.9.5. The Hidden Layer.....	39
4.9.6. Choice of Activation Function.....	40
4.9.7. Generalization Problems.....	40
4.9.8. Stopping Criterion.....	40
4.10. Neural Network Architecture.....	40
5. Implementation of the Neural Network Model.....	44
5.1. Program Components.....	44
5.2. Selection of Face Database.....	47
5.3. Image Resizing.....	47
5.4. Training Parameters and Weight Initialization.....	47
5.5. Training.....	48
6. Interpretation of Results.....	49
6.1. Image Allocation.....	49
6.2. Evaluation of parameters.....	50
6.3. Fixed and Variable Parameters.....	50
6.4. Learning Rate.....	51
6.5. Number of Hidden Neurons.....	52
6.6 Learning Algorithm.....	52
7. Conclusion.....	65
References.....	66
Appendix-1.....	71
Appendix-2.....	73

INTRODUCTION

The objective of the face detection problem described in this thesis is to identify the presence of a human face in a given image. In image analysis, there is always a need for a technique to model the human vision. An artificial neural network classifier is designed to solve the face detection problem similar to the human brain, which can differentiate between human faces and non-faces.

Pattern recognition covers a wide range of information processing problems of a great practical significance. Pattern recognition is a branch of artificial intelligence concerned with the identification of visual patterns by computers. For the computer to recognize the pattern, the patterns must be converted into digital signals and compared with patterns already stored in memory. Pattern recognition is an integral part of machine vision and image processing, and finds its applications in many fields such as biometric and biomedical image diagnostics, remote sensing, fault detection in machinery, and handwritten character recognition. These problems can quite often be solved by humans in a seemingly effortless fashion. However, designing a system for automatic image content recognition is a challenging task [16].

The objective of pattern recognition is to recognize objects in the scene from a set of measurements of the object. Each object is a pattern and measured values are the features of the pattern. Recognition is the concept of learning from sample patterns. A set of similar objects possessing more or less identical features are said to belong to a certain pattern class [17].

Techniques used to measure the features are known as feature extraction techniques. Patterns may be described by a set of features, all of which may not have enough discriminatory power to single out one class of patterns from another. The selection and extraction of appropriate features from patterns pose is the first major challenge in pattern recognition.

Numerous techniques have been developed to detect faces in a colour and gray scale images and they are classified into four categories [2]; Knowledge-based methods, Feature invariant approaches, Template matching methods and Appearance-based methods. In case of Appearance-based method the models are learned from a set of training images which should capture or identify the variations of facial appearance. These learned models are then used for detection.

In this work MATLAB is used to implement the algorithm to detect the presence of human faces in the given images. The results of this thesis work can be used for important applications such as automated security systems, indexing and retrieval of video images, and for face detection in crowded images. The main contribution of the work presented in this thesis is the implementation of neural network algorithm using scaled conjugate gradient training method and Gabor filter facial feature extraction.

Chapter II contains a brief description of the existing face detection techniques. Chapter III contains introduction to the digital image processing and also this chapter discusses about the Gabor filters and feature extraction of the images. Chapter IV explains the Artificial Neural Network, the architecture used, the threshold functions and also discusses the design of the feedforward neural network. Chapter V summarizes the implementation of the neural network model for the proposed method. Chapter VI contains the results and their interpretation. Conclusions and suggestions for future work are made in chapter VII.

FACE DETECTION IN IMAGE

In this section we review existing techniques to detect faces from a single intensity or colour image and classified into four categories [2];

1. Knowledge-based methods: It dependence on using the rules about human facial feature. It is easy to come up with simple rules to describe the features of a face and their relationships. For example, a face often appears in an image with two eyes that are symmetric to each other, a nose, and a mouth. , and features relative distance and position represent relationships between feature. After detecting features, verification is done to reduce false detection. This approach is good for frontal images but the difficulty of this method is how to translate human knowledge into known rules and to detect faces in different poses.

2. Feature based method: This approach depends on extraction of facial features that are not affected by variations in lighting conditions, pose, and other factors. These methods are classified according to the extracted features [18]. Feature-based techniques depend on feature derivation and analysis to gain the required knowledge about faces. Features may be skin colour, face shape, or facial features like eyes, nose, etc.... Feature based methods are preferred for real time systems where the multi-resolution window scanning used by image based methods are not applicable.

Human skin colour is an effective feature used to detect faces, although different people have different skin colour, several studies have shown that the basic difference based on their intensity rather than their chrominance. Textures of human faces have a special texture that can be used to separate them from different objects. Facial Features method depends on detecting features of the face. Some users use the edges to detect the features of the face, and then grouping the edges. Some others use the blobs and the streaks instead of edges. For example, the face model consists of two dark blobs and three light blobs to represent eyes, cheekbones, and nose. The model uses streaks to represent the outlines of the faces like, eyebrows, and lips .Multiple Features methods use several combined facial features to locate or detect faces. First find the face by using features like

skin colour, size and shape and then verifying these candidates using detailed features such as eye brows, nose, and hair.

3. Template matching methods: Template matching methods use the correlation between pattern in the input image and stored standard patterns of a whole face or face features to determine the presence of a face or face features. Predefined templates as well as deformable templates can be used. Several standard patterns of a faces are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection. These methods have been used for both face localization and detection.

4. Appearance-based methods: In this approach, there is a predefined standard face pattern which is used to match with the segments in the image to determine whether they are faces or not. It uses training algorithms to classify regions into face or non-face classes. In contrast to template matching, the models (or templates) are learned from a set of training images which should capture the representative variability of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection.

2.1 Knowledge-Based Methods

In this approach, face detection methods are developed based on the rules derived from the knowledge of human faces. It is easy to come up with simple rules to describe the features of a face and their relationships. For example, a face often appears in an image with two eyes that are symmetric to each other, a nose, and a mouth. The relationships between features can be represented by their relative distances and positions. Facial features in an input image are extracted first, and face candidates are identified based on the coded rules. A verification process is usually applied to reduce false detections. One problem with this approach is the difficulty in translating human knowledge into well-defined rules. If the rules are strict, they may fail to detect faces that do not pass all the rules. If the rules are too general, they may give many false positives. Moreover, it is difficult to extend this approach to detect faces in different poses.

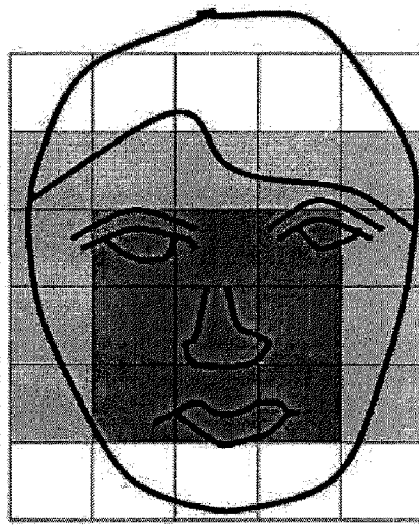


Fig 2.1: A typical face used in knowledge-based top-down methods: Rules are coded based on human knowledge about the characteristics (e.g., intensity distribution and difference) of the facial regions

The system consists of three levels of rules. At the highest level, all possible face candidates are found by scanning a window over the input image and applying a set of rules at each location. The rules at a higher level are general descriptions of what a face looks like while the rules at lower levels rely on details of facial features. A multiresolution hierarchy of images is created by averaging and subsampling. Examples of the coded rules used to locate face candidates in the lowest resolution include: “the center part of the face (the dark shaded parts which are shown in Fig. 2.1) has four cells with a basically uniform intensity,” “the upper round part of a face (the light shaded parts which are shown in Fig. 2.1) has a basically uniform intensity,” and “the difference between the average gray values of the center part and the upper round part is significant.” The lowest resolution (Level 1) image is searched for face candidates and these are further processed at finer resolutions. At Level 2, local histogram equalization is performed on the face candidates received from Level 2, followed by edge detection. Surviving candidate regions are then examined at Level 3 with another set of rules that respond to facial features such as the eyes and mouth.

2.2 Feature-Based Methods

In contrast to the knowledge-based approach, this method is based on finding invariant features of faces for detection. The observation is made based on the assumption that humans can effortlessly detect faces and objects in different poses and lighting conditions so, there must exist properties or features which are invariant over these variabilities. Numerous methods have been proposed to first detect facial features and then to infer the presence of a face. Facial features such as eyebrows, eyes, nose, mouth, and hair-line are commonly extracted using edge detectors. Based on the extracted features, a statistical model is built to describe their relationships and to verify the existence of a face. One problem with these feature-based algorithms is that the image features can be severely corrupted due to illumination, noise, and occlusion. Feature boundaries can be weakened for faces, while shadows can cause numerous strong edges which together render perceptual grouping algorithms useless.

2.2.1 Facial Features

This method is based on segmenting a face from a cluttered background for face identification. It uses an edge map and heuristics to remove and group edges so that only the ones on the face contour are preserved. An ellipse is then fit to the boundary between the head region and the background. This algorithm achieves 80 percent accuracy on a database of 48 images with cluttered backgrounds. Instead of using edges, blobs and streaks (linear sequences of similarly oriented edges) can be used for a simple face detection method. The face model consists of two dark blobs and three light blobs to represent eyes, cheekbones, and nose. The model uses streaks to represent the outlines of the faces, eyebrows, and lips. Two triangular configurations are utilized to encode the spatial relationship among the blobs. Next, the image is scanned to find specific triangular occurrences as candidates. A face is detected if streaks are identified around a candidate.

There is another method to locate facial features and faces in gray scale images. After band pass filtering, morphological operations are applied to enhance regions with high intensity that have certain shapes (e.g., eyes). The histogram of the processed image typically exhibits a prominent peak. Based on the peak value and its width, adaptive threshold values are selected in order to generate two binarized images. Connected components are identified in both binarized images to identify the areas of candidate facial features. Combinations of such areas are then evaluated with classifiers, to determine

whether and where a face is present. This method has been tested with head-shoulder images of 40 individuals and with five video sequences where each sequence consists of 100 to 200 frames. However, it is not clear how morphological operations are performed and how the candidate facial features are combined to locate a face.

2.2.2 Texture

Human faces have a distinct texture that can be used to separate them from different objects. This method is developed that infers the presence of a face through the identification of face-like textures. The textures are computed using second-order statistical features (SGLD) on sub images of 16x16 pixels. Three types of features are considered: skin, hair, and others. This method uses a cascade correlation neural network for supervised classification of textures and a self-organizing feature map to form clusters for different texture classes. To infer the presence of a face from the texture labels, they suggest using votes of the occurrence of hair and skin textures. However, only the result of texture classification is reported, not face localization or detection.

2.2.3 Multiple Features

Recently, numerous methods that combine several facial features have been developed to locate or detect faces. Most of them utilize global features such as skin colour, size, and shape to find face candidates, and then verify these candidates using local, detailed features such as eye brows, nose, and hair. Atypical approach begins with the detection of skin-like regions. Next, skin-like pixels are grouped together using connected component analysis or clustering algorithms. If the shape of a connected region has an elliptic or oval shape, it becomes a face candidate. Finally, local features are used for verification.

There is another method to detect faces in colour images based on fuzzy theory. The method uses two fuzzy models to describe the distribution of skin and hair colour in CIE XYZ colour space. Five (one frontal and four side views) head-shape models are used to abstract the appearance of faces in images. Each shape model is a 2D pattern consisting of $m \times n$ square cells where each cell may contain several pixels. Two properties are assigned to each cell: the skin proportion and the hair proportion, which indicate the ratios of the skin area (or the hair area) within the cell to the area of the cell. In a test image, each pixel is classified as hair, face, hair/face, and hair/background based on the distribution

models, thereby generating skin-like and hair-like regions. The head shape models are then compared with the extracted skin-like and hair-like regions in a test image. If they are similar, the detected region becomes a face candidate. For verification, eye-eyebrow and nose-mouth features are extracted from a face candidate using horizontal edges.

2.3 Template Matching

In template matching, a standard face pattern is manually predefined or parameterized by a function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently. The existence of a face is determined based on the correlation values. This approach has the advantage of being simple to implement. However, it has proven to be inadequate for face detection since it cannot effectively deal with variation in scale, pose, and shape. Multiresolution, multiscale, subtemplates, and deformable templates have subsequently been proposed to achieve scale and shape invariance.

2.3.1 Predefined Templates

The method uses several subtemplates for the eyes, nose, mouth, and face contour to model a face. Each subtemplate is defined in terms of line segments. Lines in the input image are extracted based on greatest gradient change and then matched against the subtemplates. The correlations between subimages and contour templates are computed first to detect candidate locations of faces. Then, matching with the other subtemplates is performed at the candidate positions. In other words, the first phase determines focus of attention or region of interest and the second phase examines the details to determine the existence of a face.

There is another method which is a two stage face detection method in which face hypotheses are generated and tested. A face model is built in terms of features defined by the edges. These features describe the curves of the left side, the hair-line, and the right side of a frontal face. The Marr-Hildreth edge operator is used to obtain an edgemap of an input image. A filter is then used to remove objects whose contours are unlikely to be part of a face. Pairs of fragmented contours are linked based on their proximity and relative orientation. Corners are detected to segment the contour into feature curves. These feature curves are then labeled by checking their geometric properties and relative positions in the neighborhood. Pairs of feature curves are joined by edges if their attributes are compatible.

The ratio of the feature pairs forming an edge is compared with the golden ratio and a cost is assigned to the edge. If the cost of a group of three feature curves (with different labels) is low, the group becomes a hypothesis. When detecting faces in newspaper articles, collateral information, which indicates the number of persons in the image, is obtained from the caption of the input image to select the best hypotheses. The system reports a detection rate of approximately 70 percent based on a test set of 50 photographs. However, the faces must be upright, unoccluded, and frontal.

2.4 Appearance-Based Methods

In contrast to the template matching methods where templates are predefined by experts, the “templates” in appearance-based methods are learned from examples in images. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and nonface images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection. Meanwhile, dimensionality reduction is usually carried out for the sake of computation efficiency and detection efficacy. Many appearance-based methods can be understood in a probabilistic framework. An image or feature vector derived from an image is viewed as a random variable x , and this random variable is characterized for faces and nonfaces by the class-conditional density functions $p(x|\text{face})$ and $p(x|\text{nonface})$. Bayesian classification or maximum likelihood can be used to classify a candidate image location as face or nonface. Unfortunately, a straightforward implementation of Bayesian classification is infeasible because of the high dimensionality of x , because $p(x|\text{face})$ and $p(x|\text{nonface})$ are multimodal, and because it is not yet understood if there are natural parameterized forms for $p(x|\text{face})$ and $p(x|\text{nonface})$. Hence, much of the work in an appearance-based method concerns empirically validated parametric and nonparametric approximations to $p(x|\text{face})$ and $p(x|\text{nonface})$.

2.4.1 Eigenfaces

A simple neural network is demonstrated to perform face recognition for aligned and normalized face images. The neural network computes a face description by approximating the eigenvectors of the image’s autocorrelation matrix. These eigenvectors are later known as Eigenfaces. Given a collection of $n \times m$ pixel training images represented as a vector of size $m \times n$, basis vectors spanning an optimal subspace are

determined such that the mean square error between the projection of the training images onto this subspace and the original images is minimized. They call the set of optimal basis vectors eigenpictures since these are simply the eigenvectors of the covariance matrix computed from the vectorized face images in the training set. Experiments with a set of 100 images show that a face image of 91x50 pixels can be effectively encoded using only 50 eigenpictures, while retaining a reasonable likeness (i.e., capturing 95 percent of the variance).

2.4.2 Distribution-Based Methods

Distribution-based system for face detection demonstrates how the distributions of image patterns from one object class can be learned from positive and negative examples (i.e., images) of that class. The system consists of two components, distribution-based models for face/nonface patterns and a multilayer perceptron classifier. Each face and nonface example is first normalized and processed to a 19x19 pixel image and treated as a 361 dimensional vector or pattern. Next, the patterns are grouped into six face and six nonface clusters using a modified k-means algorithm, as shown in Fig. 2.2.

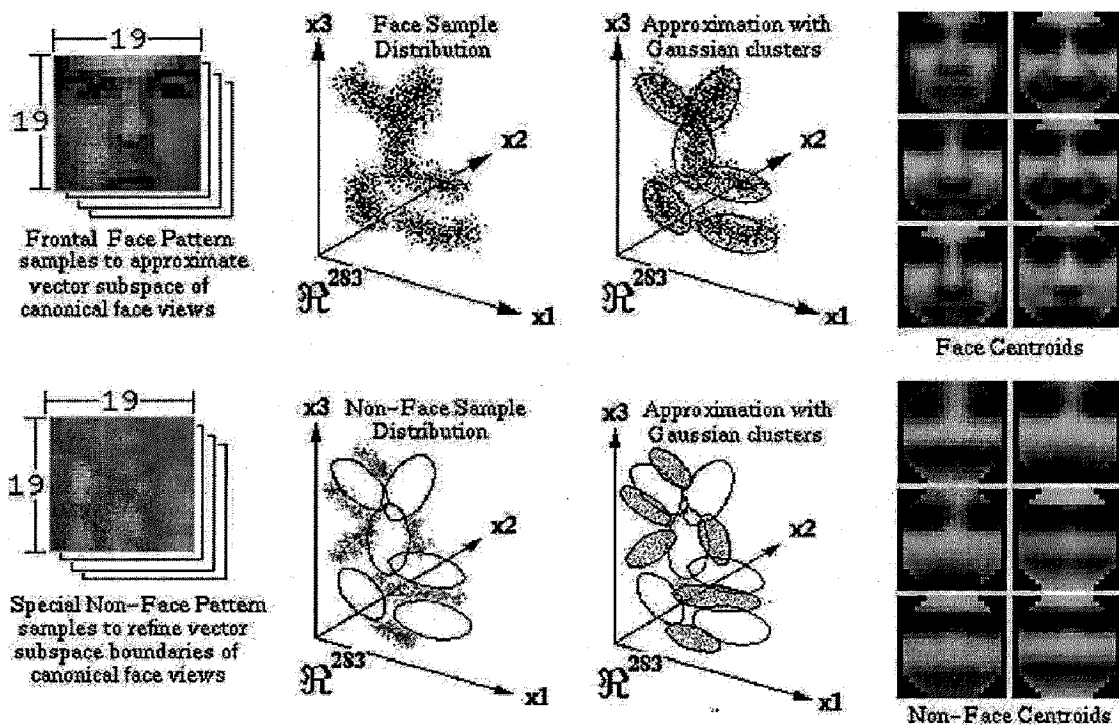


Fig 2.2: Face and nonface clusters

Each cluster is represented as a multidimensional Gaussian function with a mean image and a covariance matrix. Two distance metrics are computed between an input image pattern and the prototype clusters. The first distance component is the normalized distance between the test pattern and the cluster centroid, measured within a lower-dimensional subspace spanned by the cluster's 75 largest eigenvectors. The second distance component is the Euclidean distance between the test pattern and its projection onto the 75-dimensional subspace. This distance component accounts for pattern differences not captured by the first distance component. The last step is to use a multilayer perceptron (MLP) network to classify face window patterns from nonface patterns using the twelve pairs of distances to each face and nonface cluster. The classifier is trained using standard backpropagation from a database of 47,316 window patterns. There are 4,150 positive examples of face patterns and the rest are nonface patterns. Note that it is easy to collect a representative sample face patterns, but much more difficult to get a representative sample of nonface patterns. This problem is alleviated by a bootstrap method that selectively adds images to the training set as training progress. Starting with a small set of nonface examples in the training set, the MLP classifier is trained with this database of examples. Then, they run the face detector on a sequence of random images and collect all the nonface patterns that the current system wrongly classifies as faces. These false positives are then added to the training database as new nonface examples.

2.4.3 Support Vector Machines

Support Vector Machines (SVMs) can be considered as a new paradigm to train polynomial function, neural networks, or radial basis function (RBF) classifiers. While most methods for training a classifier (e.g., Bayesian, neural networks, and RBF) are based on minimizing the training error, i.e., empirical risk, SVMs operates on another induction principle, called structural risk minimization, which aims to minimize an upper bound on the expected generalization error. An SVM classifier is a linear classifier where the separating hyperplane is chosen to minimize the expected classification error of the unseen test patterns. This optimal hyperplane is defined by a weighted combination of a small subset of the training vectors, called support vectors. Estimating the optimal hyperplane is equivalent to solving a linearly constrained quadratic programming problem.

DIGITAL IMAGE PROCESSING

The purpose of face detection is to examine and extract information from a set of images and try to find the exact location of the face in an image. For such a system to detect well, any extracted feature has to be accurate. Image processing is therefore used to eliminate unwanted information and extract useful features from an image. Machine vision systems make use of image processing technique to carry out object identification and categorization.

3.1. Image enhancement

Image enhancement [19] improves the detect-ability of important image details or objects. It is normally performed in the first stage of digital image processing. Image enhancement operations transform an input image into another image, which is an improved version of the input. Examples of such operations include histogram stretching, convolution, noise reduction, smoothing, and edge enhancement.

3.1.1. Point operations

Point operations are based on histogram modification techniques [19]. Common histogram operations are sliding, stretching, and equalization.

3.1.2. Mask operations

Discrete convolution is used to build any linear and shift invariant filter. According to (20), the equation for the convolution $g(x)$ of the sequence $f(x)$ with the convolution mask $h(x)$ is

$$G(x) = f(x) * h(x) = \sum h(x-k) * f(k) \quad (3.1)$$

3.2 Gabor filter

A Gabor filter is a linear filter used in image processing for edge detection. Frequency and orientation representations of Gabor filter are similar to those of human visual system, and it has been found to be particularly appropriate for texture

representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. The Gabor filters are self-similar - all filters can be generated from one mother wavelet by dilation and rotation.

Its impulse response is defined by a harmonic function multiplied by a Gaussian function. Because of the multiplication-convolution property (Convolution theorem), the Fourier transform of a Gabor filter's impulse response is the convolution of the Fourier transform of the harmonic function and the Fourier transform of the Gaussian function [6].

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (3.2)$$

Where

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

And

$$y' = -x \sin \theta + y \cos \theta$$

In this equation, λ represents the wavelength of the cosine factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function, ψ is the phase offset, σ is the sigma of the Gaussian envelope and γ is the spatial aspect ratio, and specifies the ellipticity of the support of the Gabor function.

3.2.1 Explanation of parameters

Gabor filtering

This block implements one or multiple convolutions of an input image with a two-dimensional Gabor function:

$$g_{\lambda, \theta, \psi, \sigma, \gamma}(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (3.3)$$

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

$$y' = -x \sin \theta + y \cos \theta$$

To visualize a Gabor function select the option "Gabor function" under "Output image". The Gabor function for the specified values of the parameters "wavelength",

"orientation", "phase offset", "aspect ratio", and "bandwidth" will be calculated and displayed as an intensity map image in the output window. The image in the output window has the same size as the input image: select, for instance, input image octagon.jpg to get an output image of size 100 by 100. If lists of values are specified under "orientation(s)" and "phase offset(s)", only the first values in these lists will be used.

Two-dimensional Gabor functions were proposed by Daugman to model the spatial summation properties (of the receptive fields) of simple cells in the visual cortex. They are widely used in image processing, computer vision, neuroscience and psychophysics.

Wavelength (λ)

This is the wavelength of the cosine factor of the Gabor filter kernel and herewith the preferred wavelength of this filter. Its value is specified in pixels. Valid values are real numbers equal to or greater than 2. The value $\lambda=2$ should not be used in combination with phase offset $\varphi=-90$ or $\varphi=90$ because in these cases the Gabor function is sampled in its zero crossings. In order to prevent the occurrence of undesired effects at the image borders, the wavelength value should be smaller than one fifth of the input image size.

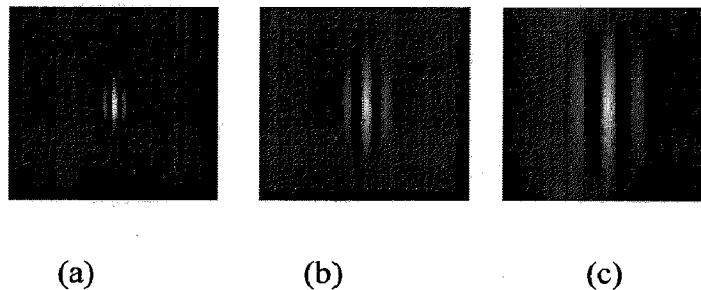


Fig 3.1: The images (of size 100 x 100) on the above shows Gabor filter kernels with values of the wavelength parameter of 5, 10 and 15, from left to right, respectively. The values of the other parameters are as follows: orientation 0, phase offset 0, aspect ratio 0.5, and bandwidth 1.

Orientation(s) (θ)

This parameter specifies the orientation of the normal to the parallel stripes of a Gabor function. Its value is specified in degrees. Valid values are real numbers between 0 and 360.

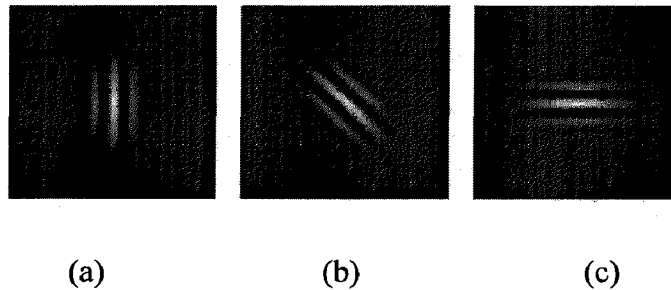


Fig 3.2: The images (of size 100 x 100) on the above shows Gabor filter kernels with values of the orientation parameter of 0, 45 and 90, from left to right, respectively. The values of the other parameters are as follows: wavelength 10, phase offset 0, aspect ratio 0.5, and bandwidth 1.

For one single convolution, enter one orientation value and set the value of the last parameter in the block "number of orientations" to 1. If "number of orientations" is set to an integer value N , $N \geq 1$, then N convolutions will be computed. The orientations of the corresponding Gabor functions are equidistantly distributed between 0 and 360 degrees in increments of $360/N$, starting from the value specified under "orientation(s)". An alternative way of computing multiple convolutions for different orientations is to specify under "orientation(s)" a list of values separated by commas (e.g. 0,45,110). In this case, the value of the parameter "number of orientations" is ignored.

Phase offset(s) (ϕ)

The phase offset ϕ in the argument of the cosine factor of the Gabor function is specified in degrees. Valid values are real numbers between -180 and 180. The values 0 and 180 correspond to center-symmetric 'center-on' and 'center-off' functions, respectively, while -90 and 90 correspond to anti-symmetric functions. All other cases correspond to asymmetric functions.

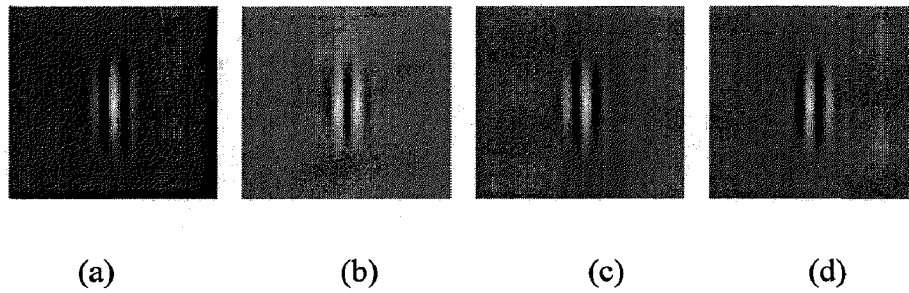


Fig 3.3: The images (of size 100 x 100) on the above shows Gabor filter kernels with values of the phase offset parameter of 0, 180, -90 and 90 degrees, from left to right, respectively. The values of the other parameters are as follows: wavelength 10, orientation 0, aspect ratio 0.5, and bandwidth 1.

If one single value is specified, one convolution per orientation will be computed. If a list of values is given (e.g. 0,90 which is default), multiple convolutions per orientation will be computed, one for each value in the phase offset list.

Aspect ratio (γ)

This parameter, called more precisely the spatial aspect ratio, specifies the ellipticity of the support of the Gabor function. For $\gamma = 1$, the support is circular. For $\gamma < 1$ the support is elongated in orientation of the parallel stripes of the function. Default value is $\gamma = 0.5$.

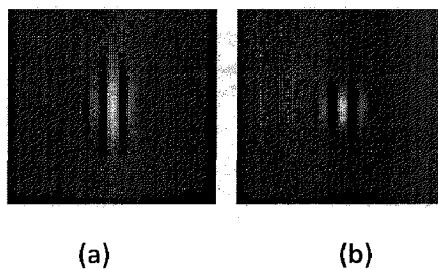


Fig 3.4: The images (of size 100 x 100) on the above shows Gabor filter kernels with values of the aspect ratio parameter of 0.5 and 1, from left to right, respectively. The values of the other parameters are as follows: wavelength 10, orientation 0, phase offset 0, and bandwidth 1.

Bandwidth (b)

The half-response spatial frequency bandwidth b (in octaves) of a Gabor filter is related to the ratio σ / λ , where σ and λ are the standard deviation of the Gaussian factor of the Gabor function and the preferred wavelength, respectively, as follows:

$$b = \log_2 \frac{\frac{\sigma}{\lambda} \pi + \sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda} \pi - \sqrt{\frac{\ln 2}{2}}}, \quad \frac{\sigma}{\lambda} = \frac{1}{\pi} \sqrt{\frac{\ln 2}{2}} \cdot \frac{2^b + 1}{2^b - 1} \quad (2.4)$$

The value of σ cannot be specified directly. It can only be changed through the bandwidth b . The bandwidth value must be specified as a real positive number. Default is 1, in which case σ and λ are connected as follows: $\sigma = 0.56 \lambda$. The smaller the bandwidth, the larger σ , the support of the Gabor function and the number of visible parallel excitatory and inhibitory stripe zones.

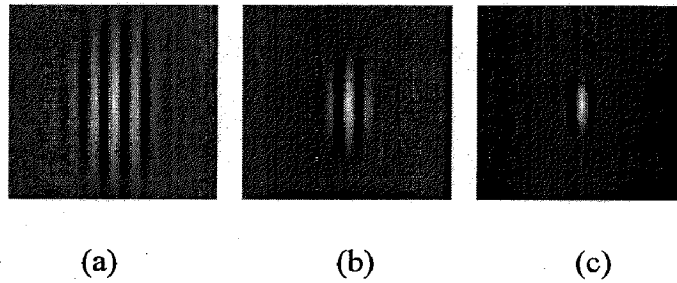


Fig 3.5: The images (of size 100 x 100) on the above shows Gabor filter kernels with values of the bandwidth parameter of 0.5, 1, and 2, from left to right, respectively. The values of the other parameters are as follows: wavelength 10, orientation 0, phase offset 0, and aspect ratio 0.5.

Number of orientations

Default value is 1. If an integer value N , $N \geq 1$, is specified then N convolutions will be computed. The orientations of the corresponding Gabor functions are equidistantly distributed between 0 and 360 degrees, with increments of $360/N$, starting from the value specified in "orientation(s)". For this option to work, one single value must be specified for the parameter "orientation(s)".

Half-wave rectification (HWR)

Enable HWR

If this option is enabled, all values in the convolution results below a certain threshold value will be set to zero (HWR is disabled by default).

HWR threshold (%)

The threshold value can be specified as a percentage of the maximum value in a given convolution result. If this percentage is set to 0, all negative values in that convolution result will be changed to 0.

Superposition of phases

If a list of multiple values is entered under parameter "Phase offset(s)" of the "Gabor filtering" block, multiple convolutions will be computed for each orientation value specified, one convolution for each phase offset value in the list. The convolution results for the different phase offset values of a given orientation can be combined in one single output image for that orientation. This combination can be done in different ways, using the L2, L1 or L-infinity norms. If the L2 norm is used, the squared values of the convolution results for the concerned orientation will be added together pixel-wise and followed by a pixel-wise square root computation to produce the combined result. The L1 and the L-infinity norms correspond to the pixel-wise sum and maximum of the absolute values, respectively. Default is the L2 norm. This choice, together with the default (0,90) of the "Phase offset(s)" of the "Gabor filtering" block, implements the Gabor energy filter that is widely used in image processing and computer vision. One can also choose not to apply superposition of phases ("None").

Surround inhibition

The Gabor filter can be augmented with surround inhibition which suppresses texture edges while leaving relatively unaffected the contours of objects and region boundaries. This biologically motivated mechanism is particularly useful for contour-based object recognition. In that case, texture edges play the role of noise that obscures object contours and region boundaries and should preferably be eliminated. One can best observe the effect of surround inhibition on different types of oriented features, such as

edges in texture vs. isolated edges and lines, by taking the default input image "synthetic1.png".

Select inhibition type

Default is "no surround inhibition".

If "isotropic surround inhibition" is selected, edges in the surroundings of a given edge have a suppression effect on that edge. The relative orientation of these edges has no influence on the suppression effect.

If "anisotropic surround inhibition" is selected, the suppression effect of edges surrounding a given edge depends on their relative orientation: edges parallel to the considered edge have stronger suppression effect than oblique edges, and orthogonal edges have no such effect.

Superposition for isotropic inhibition

If "isotropic inhibition" is selected, a superposition of the convolution results for all used orientations is computed and deployed for surround suppression. Different types of superposition can be used: L1, L2 and L-infinity norms (see the explanations of these terms under "Superposition of phases" in the "Gabor filtering" block).

Alpha (α)

This parameter controls the strength of surround suppression. Default is 1 but one may need larger values in order to completely suppress texture edges.

K_1 and K_2

The surround that has a suppression effect on an edge in a given point has annular form with inner radius controlled by the combination of values of the parameters K_1 and K_2 . The contribution of points in the annular surround is defined by a weighting function which is a half-wave rectified difference of Gaussian functions with standard deviations of $K_1\sigma$ and $K_2\sigma$ where σ is the standard deviation of the Gaussian factor of the Gabor

function(s) used. One can visualize the weighting function by selecting option "inhibition kernel" under parameter "Output image".

The inner radius of the annular surround increases with K_1 . The size of the annual surround which has substantial contribution to the suppression increases with K_2 .

Default values are $K_1 = 1$ and $K_2 = 4$.

Thinning and thresholding

These are post-processing techniques standardly used in image processing.

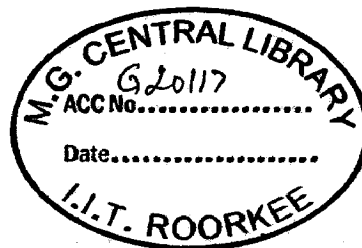
Thinning thins edges in the output to one-pixel wide edges by non-maxima suppression.

Hysteresis thresholding results in a binary output image. If it is enabled, two threshold values must be specified: T-high and T-low. These are given as fractions (between 0 and 1) of the maximum response value.

Pixels with responses higher than T-high are assigned the binary value 1 in the output, while pixels with responses below T-low are assigned the binary value 0. Pixels with responses between T-low and T-high are assigned the value 1 in the binary output if they can be connected to any pixel with a response larger than T-high through a chain of other pixels with responses larger than T-low.

3.2.2 2D GABOR WAVELET REPRESENTATIONS OF FACES

Since face recognition is not a difficult task for human beings, selection of biologically motivated Gabor filters is well suited to this problem. Gabor filters, modelling the responses of simple cells in the primary visual cortex, are simply plane waves restricted by a Gaussian envelope function.



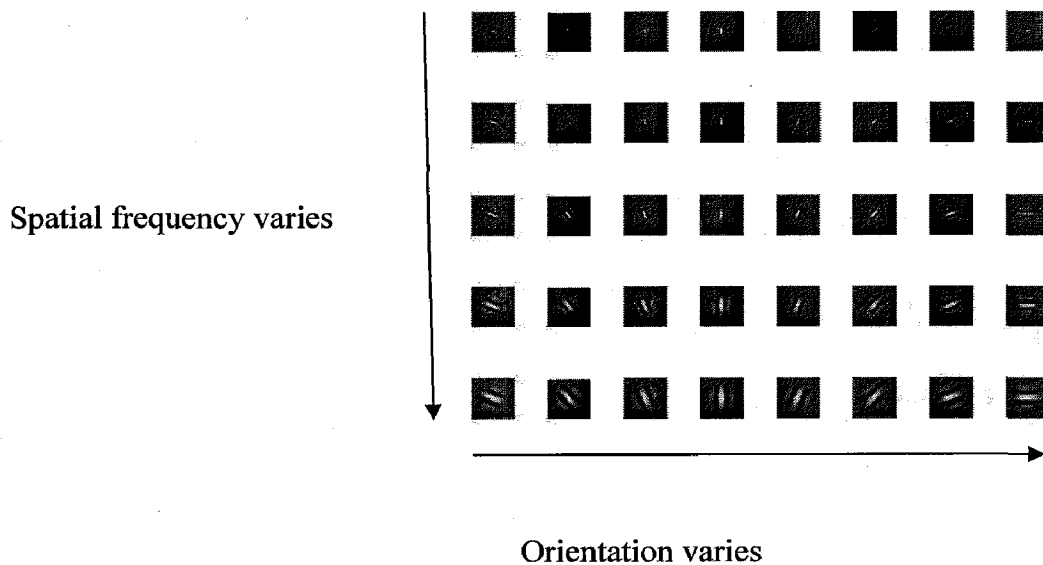


Fig 3.6: Gabor filters correspond to 5 spatial frequencies and 8 orientations

Orientation (Gabor filters in Time domain) an image can be represented by the Gabor wavelet transform allowing the description of both the spatial frequency structure and spatial relations. Convolution of the image with complex Gabor filters with 5 spatial frequency ($\nu = 0, \dots, 4$) and 8 orientation ($\mu = 0, \dots, 7$) captures the whole frequency spectrum, both amplitude and phase Fig 3.6.

One of the techniques used in the literature for Gabor based face recognition is based on using the response of a grid representing the facial topography for coding the face. Instead of using the graph nodes, high-energized points can be used in comparisons which form the basis of this work. This approach not only reduces computational complexity, but also improves the performance in the presence of occlusions.

3.3.1. Feature extraction

Feature extraction algorithm for the proposed method has two main steps in Fig 8; (1) Feature point localization, (2) Feature vector computation.

3.3.2. Feature point localization

In this step, feature vectors are extracted from points with high information content on the face image. In most feature-based methods, facial features are assumed to be the

eyes, nose and mouth. However, we do not fix the locations and also the number of feature points in this work. The number of feature vectors and their locations can vary in order to better represent diverse facial characteristics of different faces, such as dimples, moles, etc., which are also the features that people might use for recognizing faces Fig 7.



Fig 3.7: Facial feature points found as the high-energized points of Gabor wavelet responses

From the responses of the face image to Gabor filters, peaks are found by searching the locations in a window $W0$ of size $W \times W$ by the following procedure:

A feature point is located at (x_0, y_0) , if

$$R_j(x_0, y_0) = \max_{(x,y)=w_0} (R_j(x, y)) \quad (2.5)$$

$$R_j(x_0, y_0) > \frac{1}{N_1 N_2} \sum_{x=1}^{N_1} \sum_{y=1}^{N_2} R_j(x, y), \quad (2.6)$$

$$j=1, \dots, 40$$

Where R_j is the response of the face image to the j_{th} Gabor filter $N_1 N_2$ is the size of face peaks of the responses. In our experiments a 9×9 window is used to search feature points on Gabor filter responses. A feature map is constructed for the face by applying above process to each of 40 Gabor filters.

3.3.3. Feature vector generation

Feature vectors are generated at the feature Points as a composition of Gabor wavelet transform coefficients. K_{th} feature vector of i_{th} reference face is defined as,

$$v_{i,k} = \{x_k, y_k, R_{i,j}(x_k, y_k) | j = 1, \dots, 40\} \quad (2.7)$$

While there are 40 Gabor filters, feature Vectors have 42 components. The first two components represent the location of that feature point by storing (x, y) coordinates. Since we have no other information about the locations of the feature vectors, the first two components of feature vectors are very important during matching process. The remaining 40 components are the samples of the Gabor filter responses at that point. Although one may use some edge information for feature point selection, here it is important to construct feature vectors as the coefficients of Gabor wavelet transform.

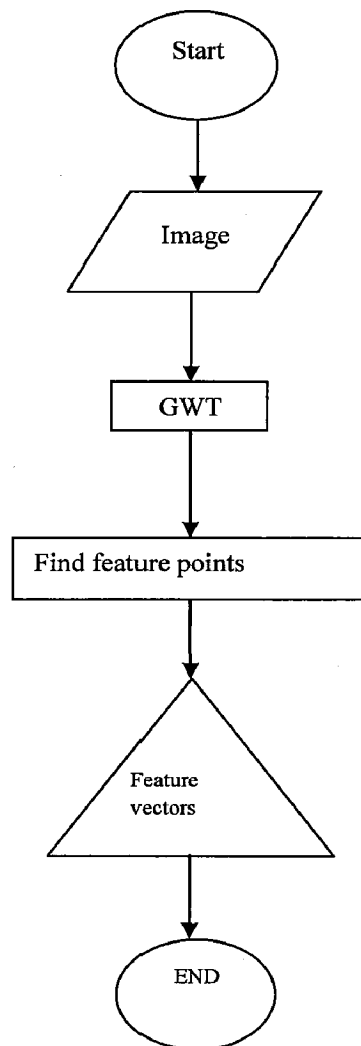


Fig 3.8: Flowchart of the feature extraction stage of the facial images.

Feature vectors, as the samples of Gabor wavelet transform at feature points, allow representing both the spatial frequency structure and spatial relations of the local image region around the corresponding feature point.

In this section the algorithm will check all potential face contained windows and the windows around them using neural network. The given image is convolved with the templates then the locations of the peaks of both the templates are saved. Then make a list of all the centre of the windows that should be checked. Yellow pixels shown in the fig 3.7 has to be tested for the presence of the face. The centres of the each window are cut and send it to Neural Network. If the answer of the NN (neural network) is near -1, it means that no face is near this Black status location, if the answer of the NN is near 1, it really contains a face around the Blue status location and all the neighbours of this location not to be checked because there is already a face cantered on this location. The Green status indicates that the face is detected in this location and marks the neighbours to be checked. Then convert the yellow to green on this location. Then finally draw a rectangle around the green status location which indicates the presence of the face.

This architecture was implemented using Matlab in a graphical environment allowing face detection in a database. It has been evaluated using the test data of 50 images containing faces on this test set a good detection faces is obtained.

The proposed method has good detection quality even if there are sunglasses in the input image. This algorithm compares faces in terms of mouth, nose and any other features rather than eyes. This method also has a simple matching procedure, low computational cost, robust to illumination changes as a property of Gabor wavelets and faster than existing methods. A new facial image can also be simply added to database by attaching new feature vectors to training images.



Fig 3.9: Finding the centre of each region

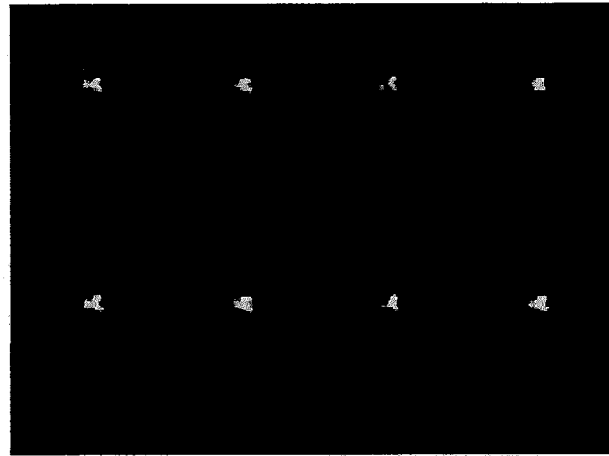


Fig 3.10: Filtering above pattern for values above threshold

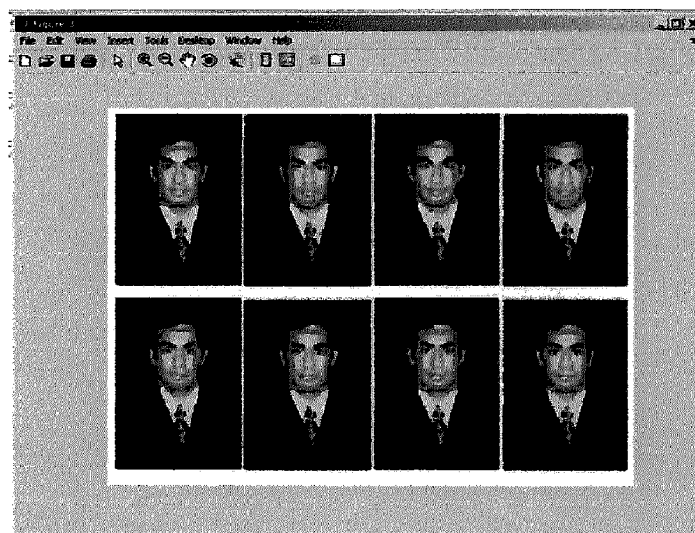


Fig 3.11: Final Result will be like this

NEURAL NETWORKS

According to Haykins [23], “A Neural Network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use”. It resembles the brain in two respects.

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The term artificial neural networks also referred to in literature as neurocomputers, connectionist network, and parallel distributed processors is used to describe various topologies of highly interconnected simple processing elements that offer an alternative to traditional approaches to computing[23]. The topic of neural networks has received much attention in the past decade. This fact is reflected in the range of publications containing related articles. Researchers from such diverse area as neuroscience, mathematics, psychology, engineering, and computer science are attempting to relate underlying models for pattern recognition, the computation that is desired, the potential parallelism that emerges, and the operation of biological neural systems.

The idea of neural networks was inspired by the structure of the human brain. Biological systems, such as the human brain, implement pattern or speech recognition computations through interconnections of physical cells called neurons. A neuron is the most basic component of a neural network [23]. There have been several models proposed to describe the behaviour of neurons in actual nervous systems [16, 23].

Artificial Neural Network (ANN), is a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. The behaviour of a neural network is defined by the way its individual computing elements are connected and by the strength of those connections or weights. ANNs are mainly applicable to problems requiring a nonlinear solution such as high-level tasks in image

processing chain (for example object recognition) rather than low-level tasks [24, 25]. Multilayer perceptron neural networks are good tools for classification purposes. These networks can approximate almost any regularity between input and output [26].

4.1 Advantages of Artificial Neural Networks

A neural network has the ability to learn through its massively parallel distributed structure, and, therefore, to generalize. Generalization refers to the neural network producing reasonable outputs for inputs not encountered during training [23]. These capabilities help the neural network to solve complex problems that are currently intractable. The use of neural networks [16, 23] offers the following useful properties and capabilities:

- **Nonlinearity:** An artificial neuron can be linear or nonlinear. Since the neural network is made up of nonlinear neurons, nonlinearity is distributed throughout the network.
- **Input-output mapping:** The neural network model consists of input signal and a corresponding desired response. The network is presented with a random set, and the synaptic weights of the network are modified to minimize the difference between the desired response and actual response of the network produced by the input signal in accordance with an appropriate statistical criterion.
- **Adaptivity:** It is the ability of a network to perform efficiently in any environment. A neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions.

4.2 Artificial Neural Networks for Image Processing

ANNs are very useful tools for nonlinear image processing, nonlinearity still being a major problem in image processing. Nonlinearity issues can be listed as follows:

- The network input can consist of pixels or measurements in images; the output can contain pixels, decisions, labels, etc., as long as all these can be coded numerically.
- Instead of designing an algorithm, one could construct an example data set and an error criterion, and train ANNs to perform the desired input-output mapping.
- ANNs can themselves be highly nonlinear

In this thesis, ANN was used to identify the faces in the images as those containing human faces and other objects. ANN solution was chosen because of the non linear nature of the problem. Many learning procedures have been proposed for neural networks over time. All the procedures, try to adjust the weights in the network model so that the models performance improves over time [27]. Currently two classes of learning procedures exist, namely, supervised and unsupervised.

In supervised learning [23], also referred to as learning with a teacher, network models are presented with a set of training patterns one by one. The outputs, generated by the networks based on the current inputs, are then compared with the desired output for that particular training pattern. The difference between the actual outputs and desired outputs, also called error, is used to update the weights so that the error will be minimized. In unsupervised learning, also known as learning without a teacher, there is no teacher to observe the learning process. That is to say, there are no labelled examples of the function to be learned by the network.

For the work described in this thesis, a supervised learning algorithm was implemented; since the label of the training set images (images with human faces vs. images without human faces) were known.

4.3 Feed-Forward Neural Network

As suggested earlier, neural networks are composed of simple elements operating in parallel. The network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the weights between elements [28]. The model that I have chosen for training images is a feed-forward network because the problem at hand is a two class problem, where the output is a single image that indicates the faces in the image.

A simple neuron consists of scalar input, p , which is transmitted through a connection that multiplies its strength by the scalar weight, w , to form the product, wp , which is again a scalar as shown in Fig 4.1 below.

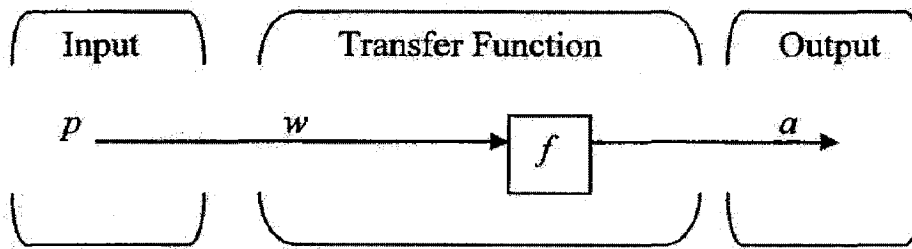


Fig 4.1: Single Neuron without Bias

The weighted input, wp along with the transfer function, f , produces the scalar output, a .

The neuron may also have a scalar bias, b , which is simply being added to the product, wp , as shown by the summing function or as shifting the function, f , to the left by an amount b . The bias is much like a weight, except that it has a constant input (weight) of 1. A neuron with bias is shown below in Fig 4.2.

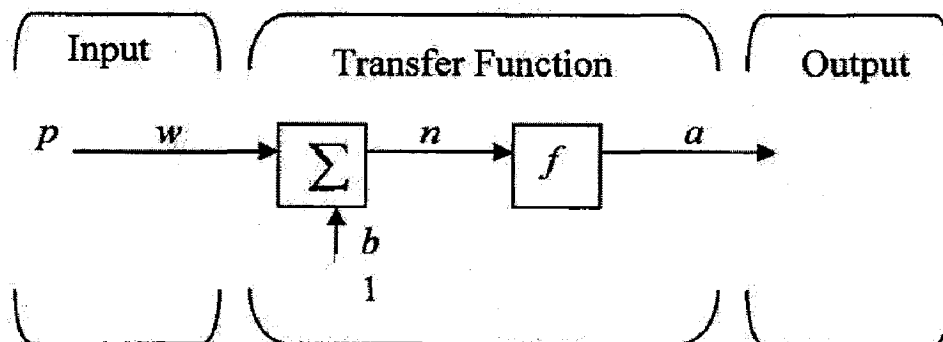


Fig 4.2: Single Neuron with Bias

The w and b are both adjustable scalar parameters of the neuron and f is a transfer function, typically a step function or a sigmoid function, which takes the argument n and produces the output a .

4.4 Network Model for the System

To perform the face detection, artificial neural network as shown in Fig 4.3, is used. The inputs to the model are the features from the given image, and the output is an image indicating the presence or absence of human faces. The parameters of this model were optimized by trial and error method [33].

$$\left\{ \begin{array}{l} > 0.9 \implies X \text{ contains a human face} \\ < 0.9 \implies X \text{ does not contains a human face} \\ = 0.9 \implies \text{unclear if } X \text{ contains a human face} \end{array} \right.$$

A network with one hidden layer consists of 100 nodes is chosen in order to represent the interrelationship among input features from images. The sigmoid function $\sigma(t)$, is chosen as the output threshold function because the sigmoid function takes the input, which may have any value between plus and minus infinity, and squashes the output into the range zero to one.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Inputs entering a neuron not only get multiplied by the weights, but they also get multiplied by the neurons characteristic equation, or transfer function. The sigmoid function is a typical neuronal non-linear transfer function that helps make outputs reachable. In other words, the sigmoid function helps a system reach desired outputs. The sigmoid transfer function is used as a suitable transfer function in this case, since the output has to be between zero and one.

The training algorithm chosen was scaled conjugate gradient algorithm, which minimizes the error $\|Y_{\text{TARGET}} - Y_{\text{ACTUAL}}\|_2$ by multi-dimensional steepest descent. On function approximation problems, for networks that contain up to a few hundred weights, the scaled conjugate gradient algorithm will have the fastest convergence. This advantage is especially noticeable if very accurate training is required.

The training data set, as shown in Appendix 1 & 2 consists of 68 face images and 55 non face images. The face images chosen represent a variety of ages, gender, with glasses, without glasses. All the non face images were random objects taken from the internet that were for free usage. Under ideal circumstances each face would have a corresponding output $Y = 1$, while non face images would have output $Y = 0$; however, as mentioned before, $|Y| > 0.9$ was considered to represent a face image, and $|Y| < 0.9$ meant a non face image.

4.5 Transfer Functions

Three basic types of transfer functions commonly used are [23]:

- **Threshold function:** The threshold function, as shown in the Fig 4.4, limits the output of the neuron to either 0, if the net input argument n is less than 0 or 1, if n is greater than or equal to 0 having the same name.

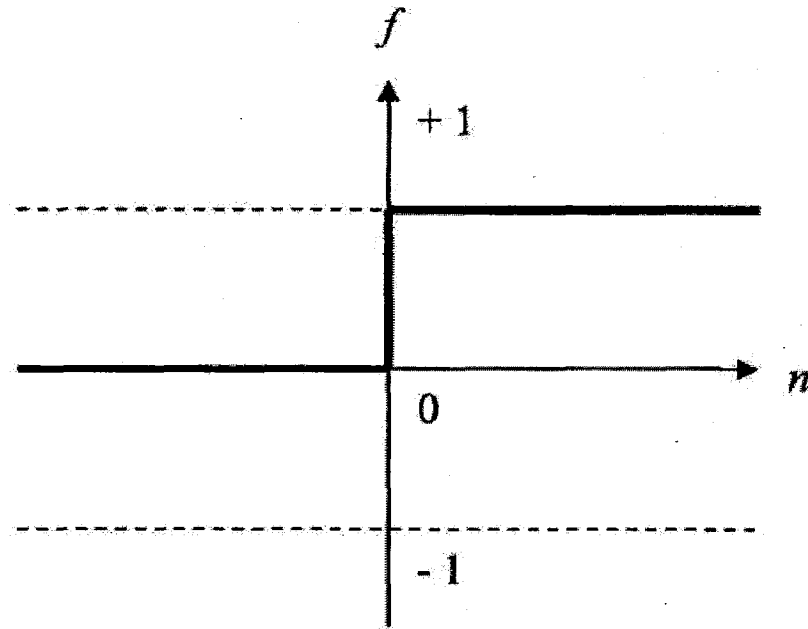


Fig 4.4: Threshold Function

- **Linear Transfer function:** Neurons of this type, as shown in Fig 4.5, are used as linear approximations in linear filters. The network output can take any value by using linear output neurons.

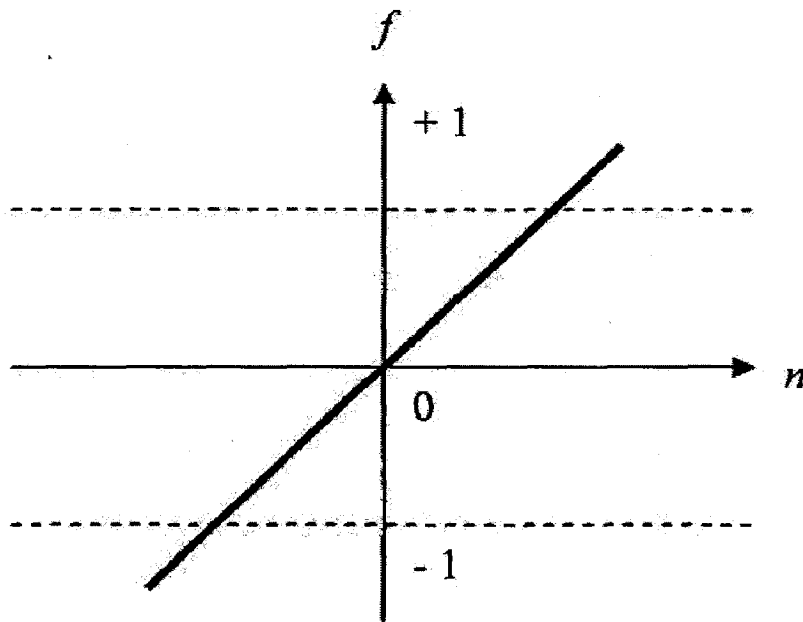


Fig 4.5: Linear Transfer Function

- **Sigmoid Transfer Function:** This type of function, as shown in Fig 4.6, takes the input, which may have any value between plus and minus infinity and squashes the output into the range 0 to 1.

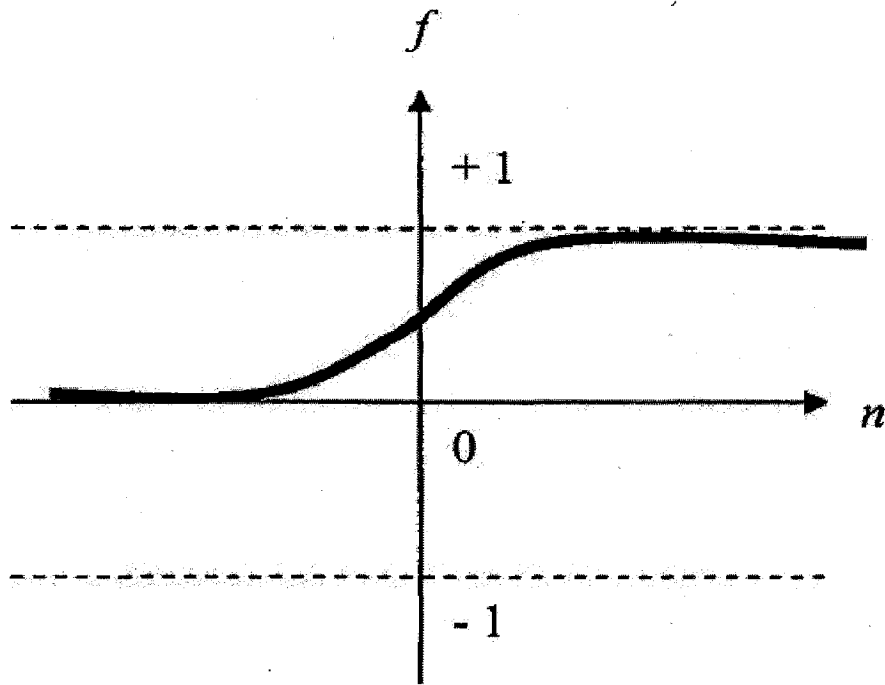


Fig 4.6: Sigmoid Transfer Function

The central idea of a neural network is that it can be adjusted so that the network exhibits desired behaviour. Thus, we can train the network itself will adjust these parameters to achieve some desired output [29]. In this thesis, tan sigmoid function was used as the transfer function, as the problem at hand was a two-class problem involving the presence (1) or absence (0) of a human face in an image.

4.6 Network Layers

In the neural network terminology, a layer is defined as a group of neurons arranged at various hierarchies [33]. So the Fig 4.7 would represent a single-layer network and Fig 4.8 would represent a two-layer network, with the middle layer called a hidden layer. There can be any number of hidden layers, and each hidden layer can have any number of nodes. Each layer has a number of properties, the most important being the transfer functions of the neurons in that layer, and the function that defines the net input of each neuron gives its weights and the output of the previous layer.

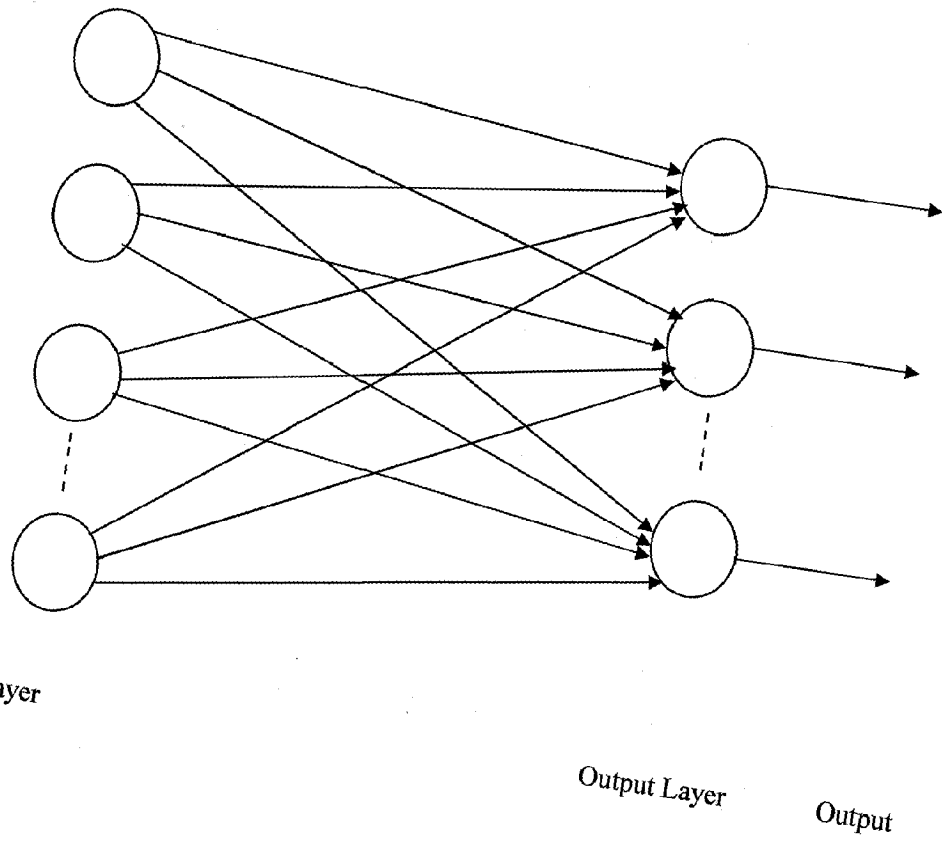


Fig 4.7: Single Layer Network

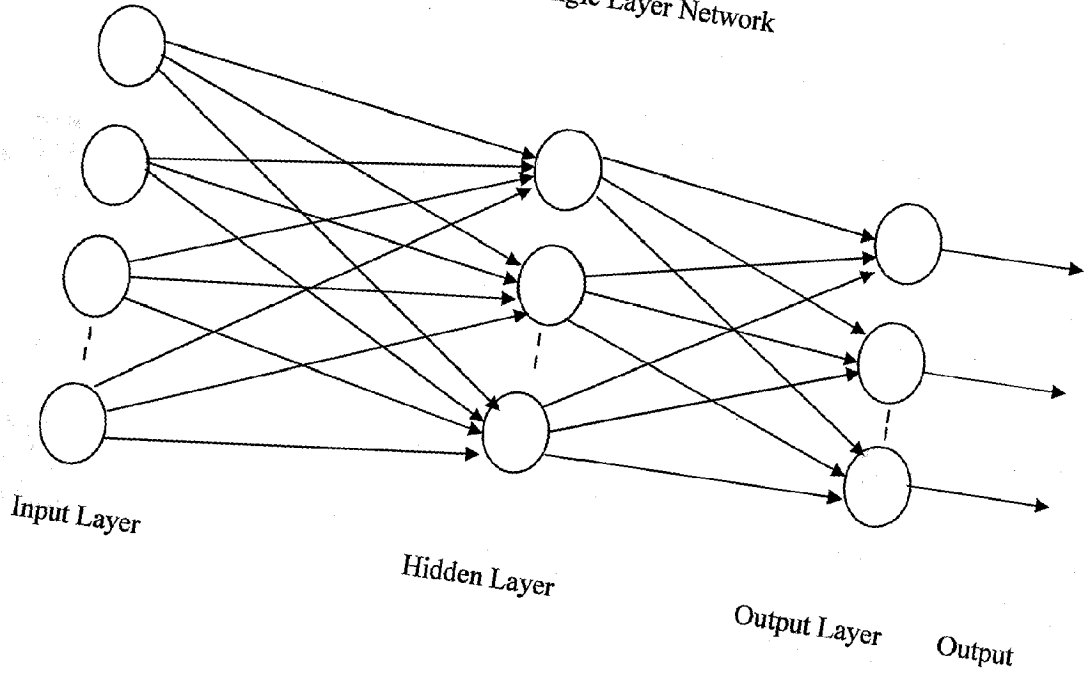


Fig 4.8: Multi Layer Network

4.7 Neural network architecture designing

Requirements

During planning stage, following requirements for the software was laid.

- a. It should have a GUI through which the user can execute each task;
- b. The interface should be simple, clear, and systematic: one button, one function;
- c. It should allow the user to select the test image;
- d. Each subprogram should be straightforward and should not contain functions that overlap;
- e. It should display both the input image and the detected image at the end of the detection process;
- f. It should display the training process for observation purposes;
- g. It should display detection results so that we are able to evaluate and analyse.

The things considered next were the image processing tasks. Internally, all pattern recognition systems have the following processes. Each operation must complete its task before the next one can begin:

1. Image acquisition
2. Image enhancement
3. Feature extraction
4. Neural network and classification
5. Detection

Since the output of each operation is the input to the next, the functional parts (1-5) must execute in sequence. The size of image (input and output) is to be kept standard so that there is better control and accuracy during matrix computation and parameter training.

4.8 Platform

MATLAB (30) is used in project because it has an integrated technical computing environment that is suitable for algorithm design and development. It is a high-level programming language that includes hundreds of functions.

4.9 Optimization of the Neural Network Parameters

Every supervised training algorithm involves the reduction of an error value. For assessing the quality and success of training, the cumulative error for the entire batch of training patterns must be computed. Mentioned in the previous chapter, the two most common performance measures used are [11].

1. Mean square error

$$\text{MSE} = 0.5 \sum \sum (d-o)^2$$

2. Root Mean Square Error

$$E_{\text{RMS}} = \text{SQRT} [(\sum \sum (d-o)^2) / PK]$$

Where P stand for all the training patterns, and K stands for all the neuron outputs. For a simple network using the gradient descent, the plotting of errors (MSE or RMS) against weight vectors will us the error surface of the network. This error profile will provide information on the network's behaviour during training. One major problem faced by error minimization is the entrapment of weights under a local of learning parameters should guarantee a good quality solution within a reasonable amount of computing time. Insertion of noise and randomness to the training set may be able to pull the process out of a local minimum.

Training a network is not an easy task. Each choice of any parameter will affect the other. An appropriate choice of any parameter will affect the other. An appropriate choice of learning parameters should guarantee that a good quality solution is found within a reasonable period of computing time. Generally, it is important to pay attention to a few aspects [32]:

4.9.1 Weight Initialization

Initialization strongly affects the classifying solution. There are now many new methods of doing it using statistics. It is necessary to reset the weights if an unsuccessful training occurs. A good convergence is also determined by the values of weights that are initialized: that is, whether they are randomly initialized or zero-initialized. The choice of

random or zero weights for the hidden and output layers affects the network's performance.

4.9.2 Weight Adaptation

Cumulative weight adjustment refers to the implementation of weight adjustments at the conclusion of a complete learning cycle. During incremental training, the weights continue to be modified as each is computed. If the network is capable and the learning rate is set correctly, the error will eventually be driven to zero. In batch mode, the weights and biases of the network are updated only after the entire training set has been applied to the network. The gradients calculated at each training examples are added together to determine the change in the weights and biases. For on-line operations, pattern-by-pattern updating rather than batch updating should be used for weight adjustment.

4.9.3 Learning Constant

Its optimum value depends on the problem to be solved and is normally chosen experimentally (with values ranging from 0.1 to 0.8). Only small learning rate guarantees a true smooth gradient descent. Too large a value leads to fast convergence but poor stability. Too small a value results in slow convergence. An adaptive rate may be more suited for exploratory work. The addition of a momentum will also accelerate convergence. It is done by supplementing the current weight adjustment parameter with a momentum term $\lambda\Delta w(t-1)$:

$$\Delta w(t) = \eta\delta y + \lambda\Delta w(t-1)$$

Where λ is the momentum constant (0.1 to 0.9) during the t^{th} change in weight. The momentum constant is greater than 0 and to ensure convergence problems because the learning-rate parameter is maintained constant throughout the computation. One may use the search-then-converge schedule defined in

$$\eta(n) = \eta(0)/(1+n/\tau)$$

Where $\eta(0)$ and τ are user selected constants. In the early stages of adaptation involving a number of iterations n , the learning-rate parameter $\eta(n)$ is small compared to the search time constant τ , and is approximately equal to $\eta(0)$. Hence, a high value for $\eta(0)$ within permissible range will help find a set of weights that hovers about a 'good' set of values. One common modification to the algorithm is to gradually reduce the value of η as the number of gradient descent steps grows. This stochastic approximation is described by

$\eta(n) = c / n$, where c is a constant. Such an adaptive learning rate is sufficient to guarantee convergence.

4.9.4 Inputs and Outputs

The training set must contain enough information to reveal the mapping structure. Its size depends on the user's decision on the number of inputs, hidden neurons, and output neurons. The number of inputs is usually determined by the dimension or the size of the data to be classified. The size of an input vector often corresponds to the number of features extracted from the previous stage. The number of output neurons is generally less than the input size. Training examples presented to the network should be randomized from one epoch to another for faster convergence. In our algorithm, a random permutation of the face input vector is fed into the backpropagation network each time. All the vectors are presented once, but their order is random in every iteration. This technique encourages greater diversity of possible paths across the error-weight surface, which tends to favour escape from local minima.

4.9.5 The hidden layer

In most situations, the best number of hidden units is determined by experimenting with several network settings and estimating the generalization error of each. The size of the hidden layer is always worth some level of consideration. Using too underfitting and high statistical bias. Using too many will increase training time. An excessive number of hidden neurons may cause overfitting, which means the network will learn the insignificant aspects of the training set, resulting in high generalization error. A rule of thumb, known as the Baum-Haussler rule (thesis 16), may be used to determine the number of hidden neurons:

$$N_{\text{hidden}} < V (N_{\text{train}} - E_{\text{tolerance}}) / (N_{\text{points}} + N_{\text{output}})$$

Where N_{hidden} is the number of hidden neurons; N_{train} is the number of training example; $E_{\text{tolerance}}$ is the error tolerance; N_{points} is the number of data points (or pixels) Per training example; and N_{output} is the number of output neurons. This rule generally ensures that the neural network generalizes, rather than memorizes. The number of hidden layers is determined by trial and error. Usually, a network with one hidden layer is sufficient for face detection. For large images, more hidden layers are needed to extract more information. Excessive use of layers may also result in overtraining.

4.9.6 Choice of Activation Function

It is important to choose an activation function that is suitable for the nature of the input pattern. A discrete function will perform poorly in a nonlinear pattern classification. The combination of the types of transfer functions in individual layers also affects the quality of training. It is advisable to try several combinations interchangeably.

4.9.7 Generalization problems

A network is said to generalize well when the input-output mapping computed by the network is correct for test data that have not been used in creating or training the network, or when the input is slightly different from the examples used to train the network. However, when the network learns too many input-output examples, the network may end up memorizing the training data. It may do so by finding a feature (due to noise, for example) that is present in the training data but not true of the underlying function that is to be modelled. When a network is overtrained, it loses the ability to generalize weight similar input-output patterns. Controlling the number of epochs and finding an optimum learning rate are good ways to avoid it.

4.9.8 Stopping Criterion

A variety of termination criteria can be used to halt the training process:

1. Stop after a fixed number of iterations;
2. Stop when the mean square error falls below some threshold;
3. Stop when the error on a separate validation set of examples meets some criterion;
4. Stop when the rate of change of error surface becomes shallower, so the change in error at each epoch becomes even smaller;
5. Stop when the generalization error is acceptable.

4.10 Neural Network Architecture

The MLP neural network has feed forward architecture within input layer, a hidden layer, and an output layer. The input layer of this network has N units for an N dimensional input vector. The input units are fully connected to the I hidden layer units, which are in turn, connected to the J output layers units, where J is the number of output classes. A Multi-Layers Perceptron (MLP) is a particular of artificial neural network. We

will assume that we have access to a training dataset of l pairs (x_i, y_i) where x_i is a vector containing the pattern, while y_i is the class of the corresponding pattern. In our case a 2-class task, y_i can be coded 1 and -1.

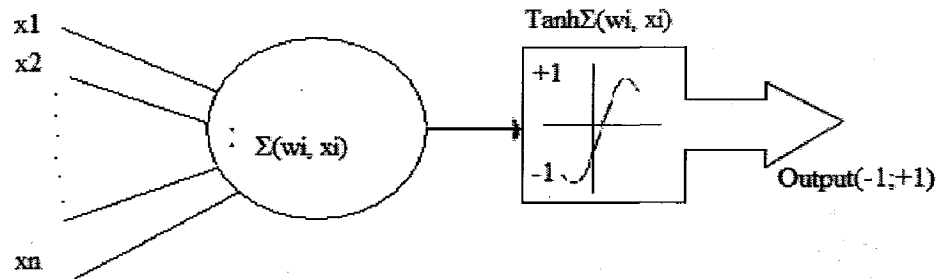


Fig 4.9: The neuron of supervised training

The Multi-Layers Perceptron consists of 3 layers, the input layer is a vector constituted by n^2 units of neurons ($n \times n$ pixel input images). The hidden layer has n neurons, and the output layer is a single neuron which is active to 1 if the face is presented and to otherwise. The activity of a particular neuron j in the hidden layer is written by

$$S_j = \sum_{i \in \text{input}} w_{ji} x_i, \quad x_i = f(s_j)(1), \quad f \quad (4.1)$$

a sigmoid function. Where W_{1i} is the set of weights of neuron i , $b_1(i)$ is the threshold and x_i is an input of the neuron. Similarly the output layer activity is

$$S_j = \sum_{i \in \text{input}} w_{ji} x_i \quad (4.2)$$

In our system, the dimension of the retina is 27×18 pixels represent human faces and non-face, the input vector is constituted by 2160 neurons, the hidden layer has 100 neurons.

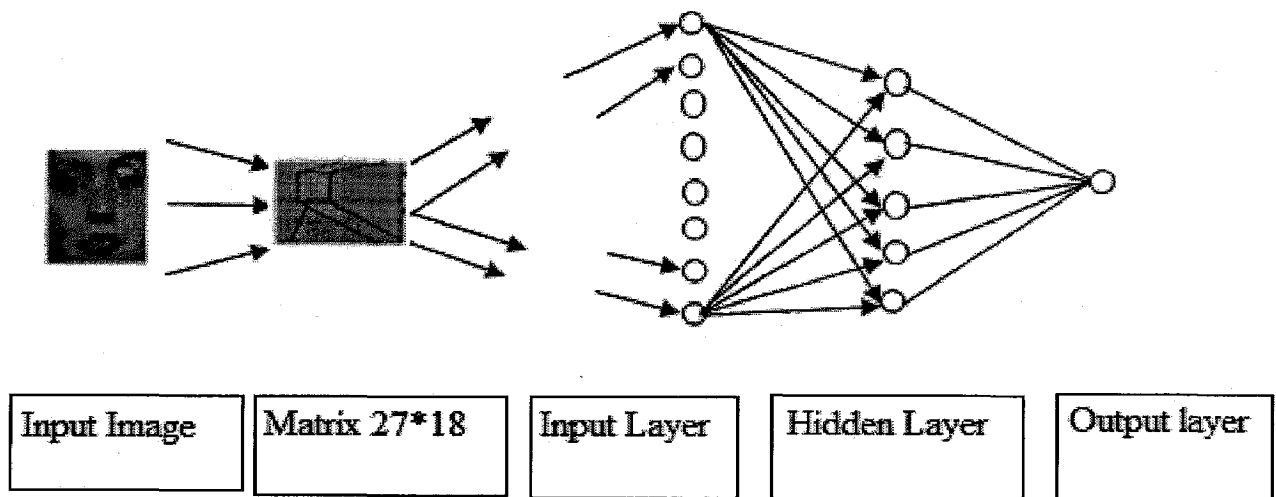


Fig 4.10: Architecture of detection system

The feed forward neural network is designed with one hundred neurons in the hidden layer and one neuron in the output layer. All data from both “face” and “non-face” folders will be gathered in a large cell array. Each column will represent the features of an image, which could be a face, or not. Rows are as follows:

Row 1: File name

Row 2: Desired output of the network corresponded to the feature vector.

Row 3: Prepared vector for the training phase

We will adjust the histogram of the image for better contrast. Then the image will be convolved with Gabor filters by multiplying the image by Gabor filters in frequency domain. To save time they have been saved in frequency domain before Features is a cell array contains the result of the convolution of the image with each of the forty Gabor filters. These matrices have been concatenated to form a bif 135x144 matrix of complex numbers. We only need the magnitude of the result. That is why “abs” is used. 135x144 has 10,400 pixels. It means that the input vector of the network would have 19,400 values, which means a large amount of computation. So we have reduced the matrix size to one-third of its original size by deleting some rows and columns. Deleting is not the best way

but it save more time compare to other methods like PCA. We should optimize this function as possible as we can.

First training the neural network and then it will return the trained network. The examples were taken from the Internet database. The MLP will be trained on 70 face and 60 non-face examples.

The Neural Network Architecture obtained from the MATLAB which consists of one input layer, one output layer and one hidden layer. The hidden layer consists of 100 neurons.

IMPLEMENTATION OF THE NEURAL NETWORK MODEL

This chapter is dedicated to description of program execution. The entire face detection system is composed of eleven-files. In the following sections, the GUI and the software structure of the program are shown, including a list of all the subprograms together with a description of their tasks. How the face images were acquired, how certain decisions were taken for training, and most importantly, how this system works.

5.1 Program Components

Fig. 6.1 is the main menu of our face detection system. The program consists of eleven component files, each performing a different task. The lists labelled “functions” and “sub-functions” are the m-files created using MATLAB. As a lot of investigations and frequent testing are conducted, the program is designed in such a way that one can perform each operation step by step. A complex program that links every function together is going to give many problems; therefore, each execution is kept simple. In this particular GUI, one button executes one process. A description of all the functions is given in below. The complete source codes are given in Appendix 1.

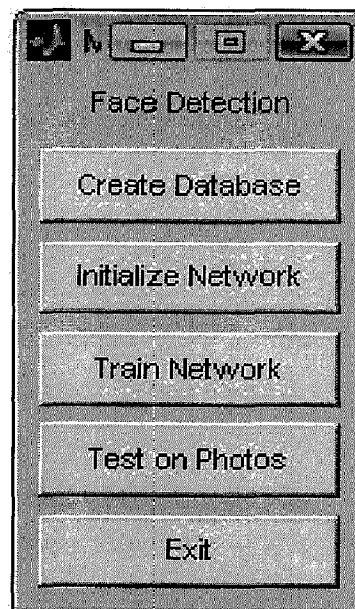


Fig 5.1: Graphical User Interface (GUI) executed by main.m

- a. **Create Database:** The images containing faces and non faces had been created for training the network.
- b. **Initialize Network:** The Neural Network is initialized with input, output and hidden layers and also with training parameters such as learning rate, training algorithm, number of epochs and performance function.
- c. **Train Network:** The Neural Network is trained with the training face and non face images as per the Network initializations.
- d. **Test on Photos:** The input images are selected from the folders for detection of the faces.
- e. **Exit:** After completion of the detection process the system will exit.

The description of program components is given below

1. main.m

User interface for accessing all operations.

2. loadimages.m

This function prepares images for training phase. All data from both “face” and “non-face” folders will be gathered in a large cell array. Each column represents the features of an image which could be a face or not. Rows are as follows: Row 1: File name
Row 2: Desired output of the network corresponded to the feature vector. Row 3: Prepared vector for the training phase, Also this script saves the database to a file named “imgdb.mat”. So we do not need to create the database more than once unless we add or delete some photos to/from “face” and “non-face” folders. Every time we do this, after recreating a database, we should initialize and train the network again. This script uses “im2vec.m” to extract features from images and vectorize them for the database.

3. createffnn.m

This function creates a feed forward neural network with one hundred neurons in the hidden layer and one neuron in the input and output layer. The network will be saved in “net.mat” for further use.

4. trainnet.m

This function trains the neural network with the face and non face images and then returns the trained network

5. create_gabor.m

This script uses gabor.m to generate forty 32×32 gabor filters and save them in a cell array matrix called "G" and in a file named "gabor.mat". This script will be invoked only once unless we delete "gabor.mat".

6. im2vec.m

This function takes a 27×18 image. It adjusts the histogram of the image for better contrast. Then the image will be convolved with gabor filters by multiplying the image by gabor filters in frequency domain. Gabor filters are stored in "gabor.m". To save time they have been saved in frequency domain before. Features135x144 is a cell array contains the result of the convolution of the image with each of the forty gabor filters. These matrixes will be concated to form a 135×144 matrix of complex numbers. we only need the magnitude of the result. That is why "abs" is used. 135×144 has 10,400 pixels. It means that the input vector of the network should have 19,400 values which mean a large amount of computation. So we reduce the matrix size to one-third of its original size by deleting some rows and columns. Deleting is not the best way but it save more time compare to other methods like PCA.

7. imscan.m

In the first section the given image is convolved with the predefined template. Then imregionimax is used to detect the centre of potential face contained window. In the second section the algorithm checks all potential face-contained windows and the windows around them using neural network. The result will be the output of the neural network for checked regions.

8. drawrec.m

This script is used to draw the rectangle around the face in the detected images which indicates that the presence of faces in the given images.

5.2 Selection of Face Database

During planning, some research was carried out on the Internet and a few face and non face images were found. Most were freely available and had a rich variety of faces and non faces, but in accordance to the objective to test the system passport size photos with 27x18 are required for training the network. We are clear that training and analyzing too many images is a waste of precious time, so 68 face images and 55 non face images were taken for training. Each of them varies in orientation and expression, some with glasses, and some without glasses. For testing the system different images are collected. Some of them are taken from the internet and some of them are taken from the digital camera.

5.3 Image Resizing

The original images are found with different dimensions such as 112x92 pixels, which mean each has a total pixel size of 10304 if it is placed in a single array row. This size is not efficient for normal training; especially there are large training sets. To avoid increased computation and long training times, it is necessary to resize the face images. The images are therefore resized to 27x18 which means each has a total pixel of 486. The input image given for testing also resized such that the faces in the images are must be approximately in the size of 27x18.

5.4 Training Parameters and Weight Initialization

A good set of training parameters ensures the learnability of the network. Network learnability relates to the ability of a learning algorithm to find a set of weights that can give the accuracy needed for a good mapping approximation. In other words, the network must find weights that can produce good generalization. An optimum set of parameters, was found, for the network after several tests. The validation set was used to perform cross validation with the training set. The first thing tested for was the maximum training epoch. Started out with 500 epochs, but the network seemed to be overtrained. Hence, the epochs were reduced and the performance goal met at 177 epochs. The performance was relatively

fine after that. The network was stable and convergence was smooth. Next, proceeded to determine the learning rate. Started out with 0.1, but it is showed signs of undertraining, continued with other values, each time increasing the learning constant by 0.1. It was finally decided on 0.8, as this value produced outputs that were stable even after the set of weights were reinitialized.

A range of hidden neuron sizes on the network was tried out and found that a neuron size weight 100 to 106 is best. The network uses the least mean square method.

5.5 Training

This program displays real-time training parameters as they go through each iteration. Knowledge of the current epoch and the corresponding error gradient enables us to keep check on the training progress. If the training error suddenly drops or increases in value, it is known that the network is facing problems and can terminate the training process immediately. The network training process is shown below.

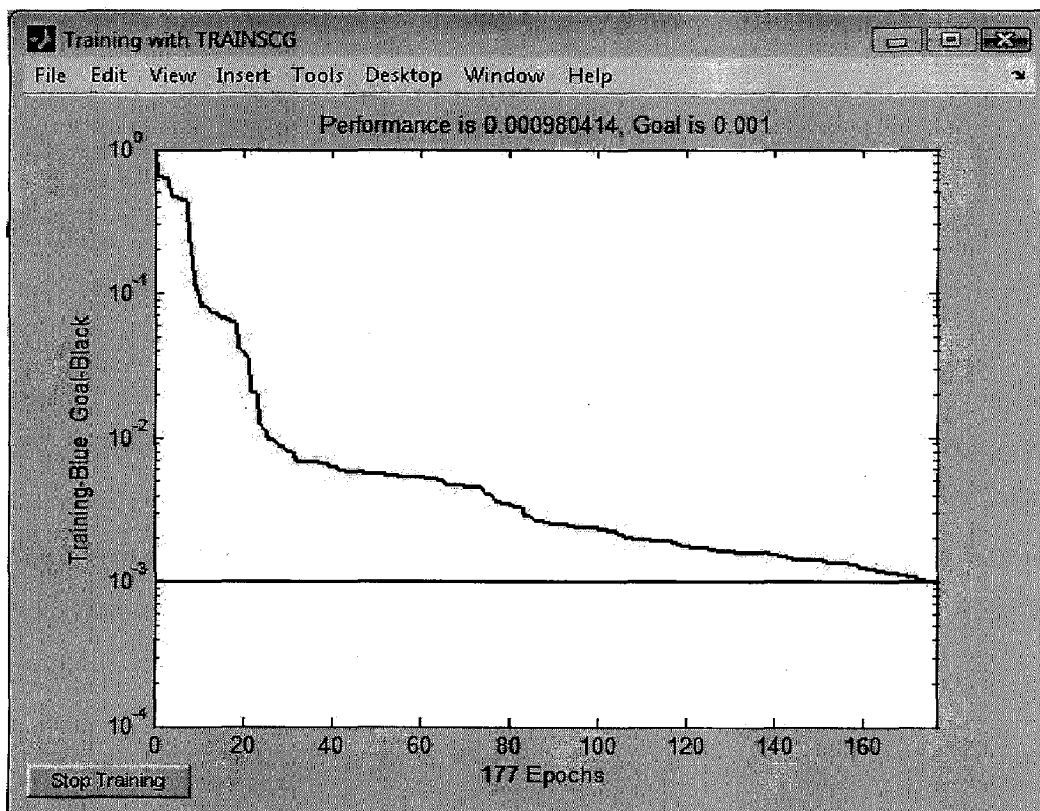


Fig 5.2: Training process

INTERPRETATION OF RESULTS

In this chapter Experimental results of the existing methods and the proposed method are presented, which are implemented using Matlab Version 7.0.1. The output images of the existing methods and the proposed method are shown in the fig 6.12 and fig 6.7 respectively. The results show that the proposed face detection system has high detection accuracy as compared to the existing methods. From the fig 6.12 & 6.13 of the existing method we can observe that some of the faces are not detected. The output images of the proposed method are shown in fig 6.7 and have very good detection quality but in fig 6.11 the detection is poor since the selected parameters of Neural Network are not optimized. The training images (face and non face images) are initially pre-processed by Gabor filters for feature extraction. Later the extracted features of the images are applied for training the neural network. Then the unknown test image is also pre-processed and Gabor features are extracted. Now the extracted features are fed and tested with the trained neural networks of the dataset of images and classified to see if the part containing a face or not.

There are virtually no tools to select an appropriate architecture and learning parameters for a neural network. In most cases, learning parameters are determined by experience or based on the trial and error method. In this chapter the learning rate, optimal number of neurons for the hidden layer and the learning algorithm are decided for training the neural network.

6.1. Image Allocation

In this experiment, 68 face images and 55 non face images are used for the training of the feed forward Neural Network shown in the Appendix1. All the training images are resized to have a dimension of 27x18 for minimizing the learning time. Different sizes of the images were used for testing the system. Some of the testing images were collected from the digital camera, web camera and some of them are collected from internet.

6.2. Evaluation of the parameters

In this section the experiments are conducted to investigate the feasibility and effectiveness of the appearance-based face detection system, which are shown in table 6.1.

Table 6.1: Areas of investigation

Evaluation Areas	Variable Parameters
Learning rate	0.1 to 0.8
Number of Hidden Neurons	100 to 106
Sigmoid function	Tansig, logsig
Training algorithm	Scg, rp, gda

6.3. Fixed and Variable Parameters

The network is structured with one hidden layer of 100 neurons it is decided by trial and error method. The sigmoid function is used in both the hidden and output layers. Initial weights for the hidden layer are set at 0. The optimized parameters for Neural Network are shown in table 6.2.

Table 6.2: Regularized Parameters

Parameters	Standardized and Optimized Value
Hidden Layer	1
Hidden Neurons	100
Sigmoid function	logistic
Training Algorithm	Scaled Conjugate Algorithm

6.4. Learning Rate

The performance of the system is very sensitive to the proper setting of the learning rate. The learning rate is held constant throughout training. If the learning is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training. Several trainings should be performed using a variety of learning rates before determining the optimum value.

Experiments were conducted with learning rates ranging from 0.1 to 0.8, each time increasing by 0.1. Detection accuracy increases with the learning constant until it reaches full detection at 0.6 or 0.8, after or before which the performance starts to deteriorate. The training with TRAINSCG and 0.1 learning rate is shown in the fig 6.1 and the training with 0.8 learning rate is shown in fig 6.2.

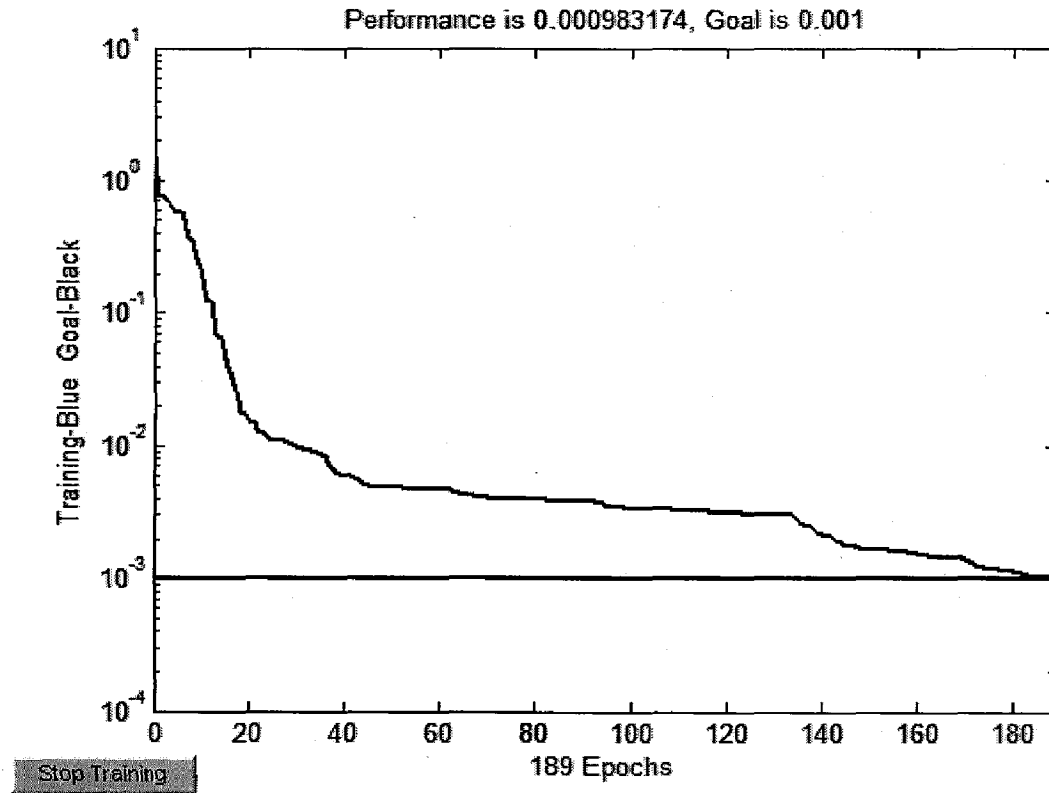


Fig 6.1: The training performance with learning rate 0.8 and scaled conjugate gradient training algorithm

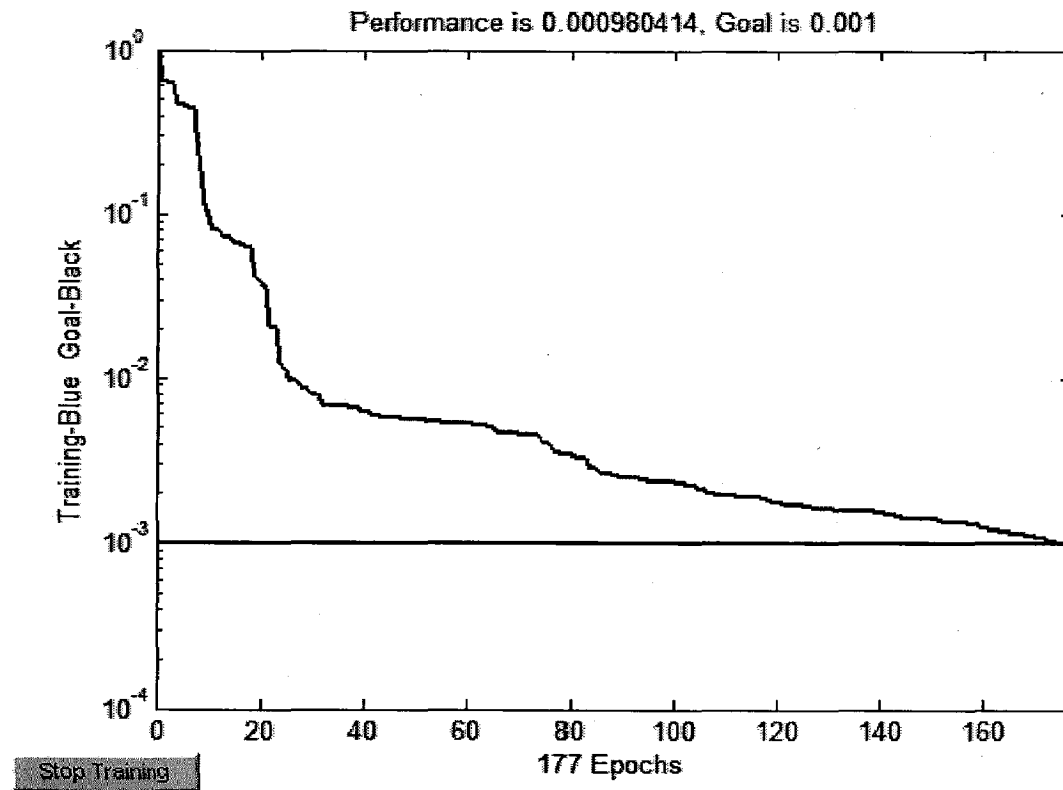


Fig 6.2: The training performance with learning rate 0.1 and scaled conjugate gradient training algorithm

6.5. Number of Hidden Neurons

How to determine the number of hidden neurons is always a discussion topic in neural networks. Baum and Haussler rule was discussed in the previous chapter, but that does not mean the equation will work for every network. But the trial and error is the best way to work out the optimal number of hidden units. The Baum and Haussler rule should be used as an estimator. In this experiment, the network was trained with different number of hidden units (90 to 150) and verified their detection performance. For a network with 100 to 106 hidden neurons, the recognition accuracy is 99%. Any number of hidden units beyond 106 will experience a gradual decrease in performance.

6.6 Learning Algorithm

Once the network weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of

examples of proper network behaviour - network inputs p and target outputs t . During training the weights and biases of the network are iteratively adjusted to minimize the network performance function. The default performance function for feedforward networks is mean square error mse - the average squared error between the networks output a and the target outputs t .

There are several different training algorithms for feedforward networks. All of these algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance. Gradient Descent (`traingd`), Gradient Descent with Momentum (`traingdm`), Variable Learning Rate (`traingda`, `traingdx`), Resilient Backpropagation (`trainrp`) and Scaled Conjugate Gradient (`trainscg`) are the different training algorithms used for feed forward networks. In this experiment different training algorithms are used and their training rate and detection rate is recorded. For this system scaled conjugate gradient algorithm will give the best performance and the results are shown in the fig 6.7 which shows the complete detection. The Resilient Back Propagation will take the less time for training as compared to scaled conjugate algorithm but the detection is poor as shown in fig 6.11. The training of the network for different training algorithm with 0.8 learning rate is shown in the figures below.

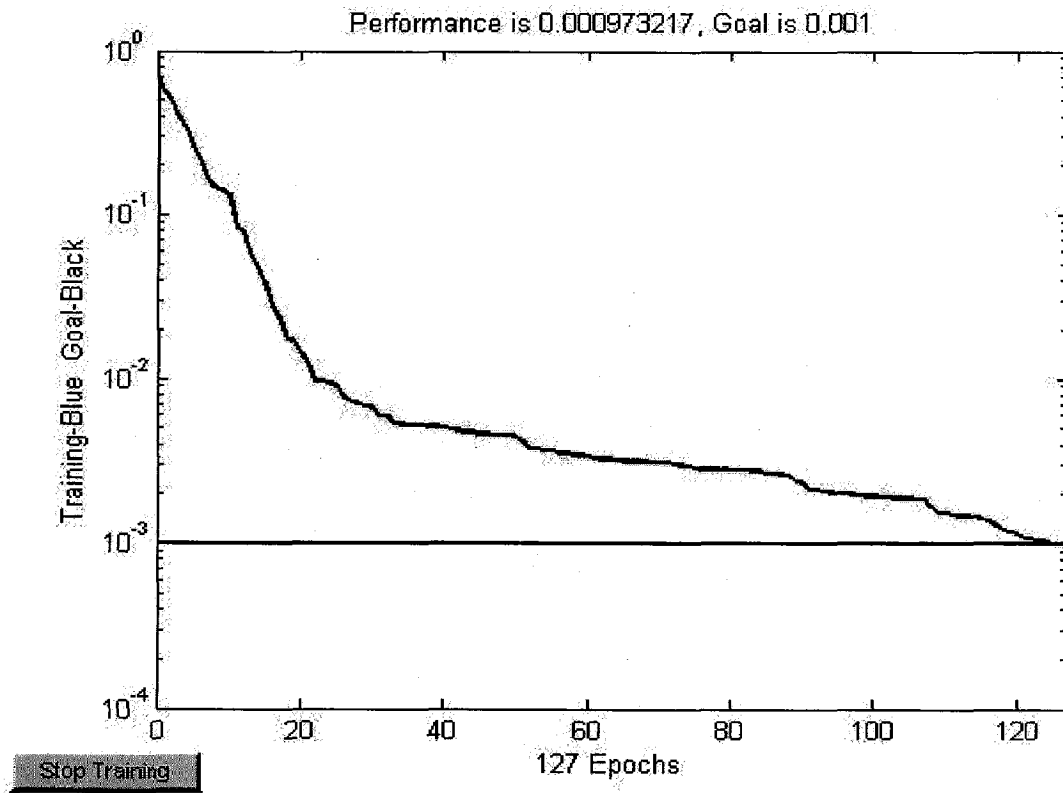


Fig 6.3: The training performance for the scaled conjugate gradient training algorithm with 0.8 learning rate, which shows that the target is reached at 127 Epochs.

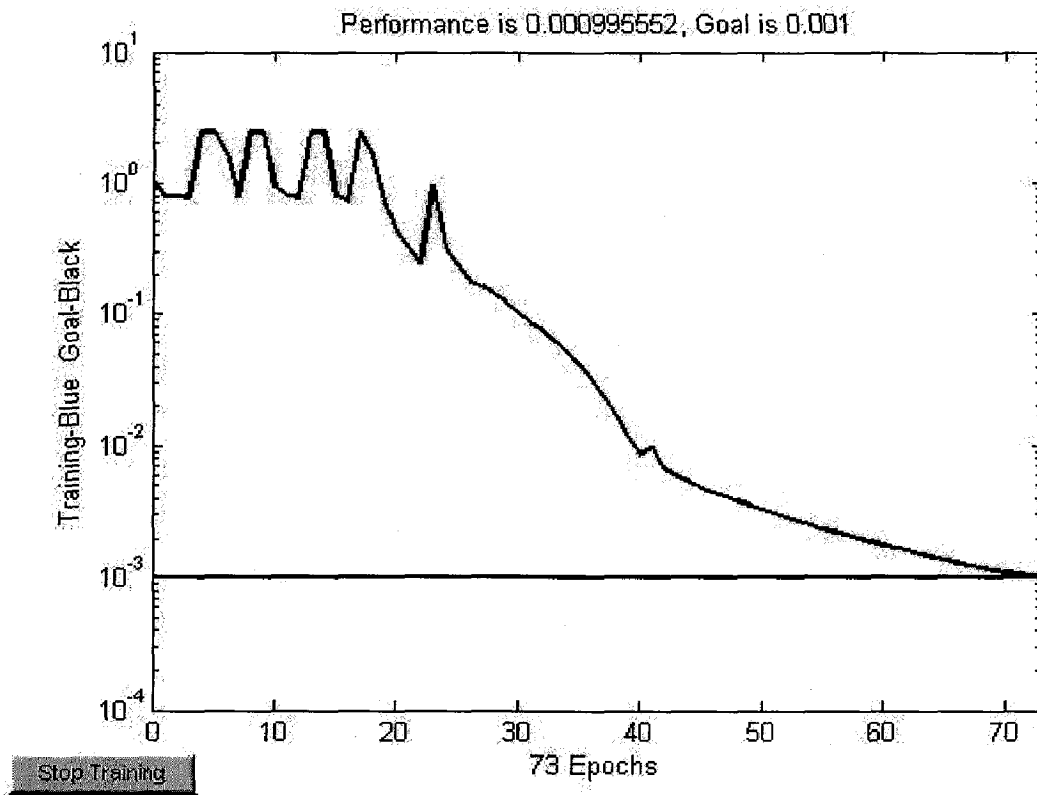


Fig 6.4: The training performance for the Resilient Backpropagation training algorithm with 0.8 learning rate, which shows that the target is reached at 73 Epochs.

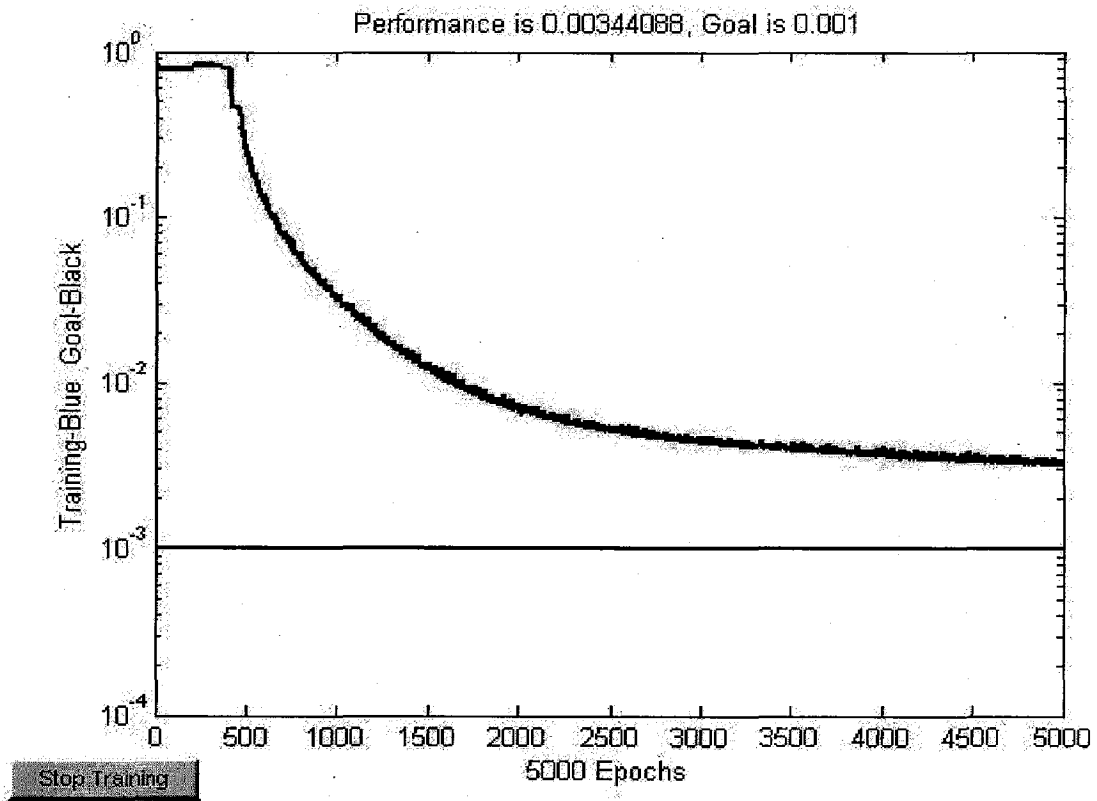


Fig 6.5: The training performance for the Gradient descent with adaptive learning rate Backpropagation training algorithm with 0.8 learning rate, which shows that the target is not reached for 5000 Epochs.

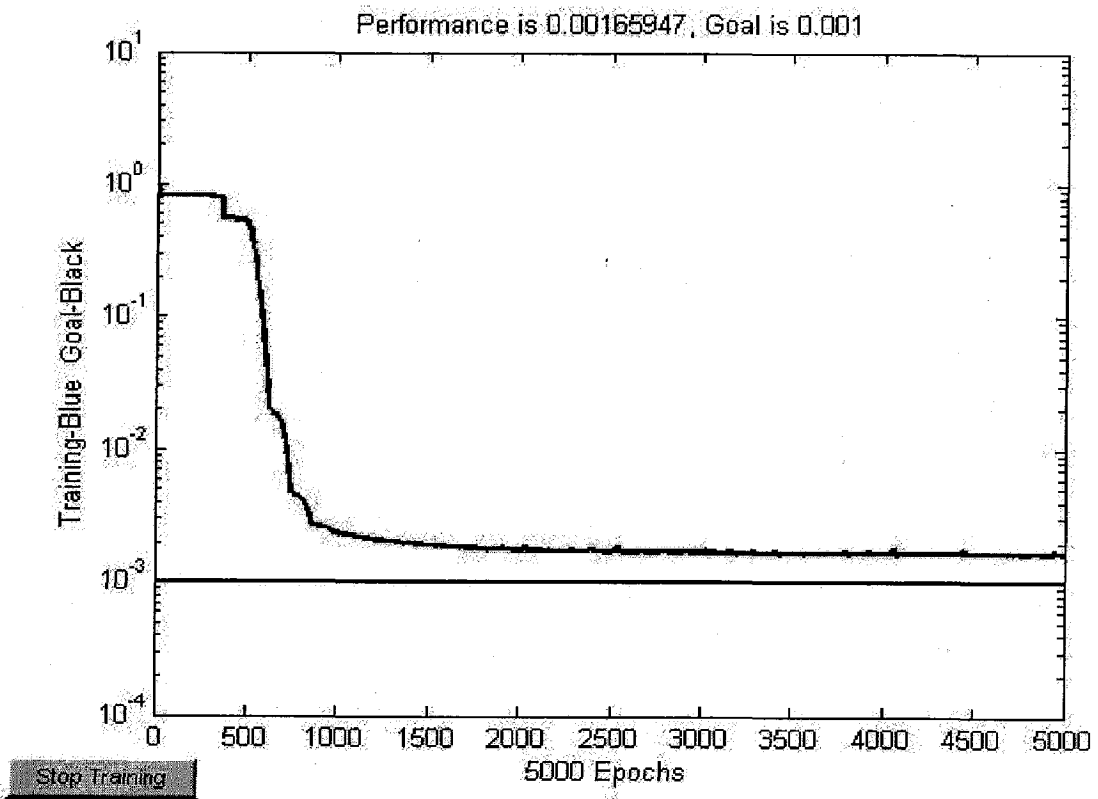
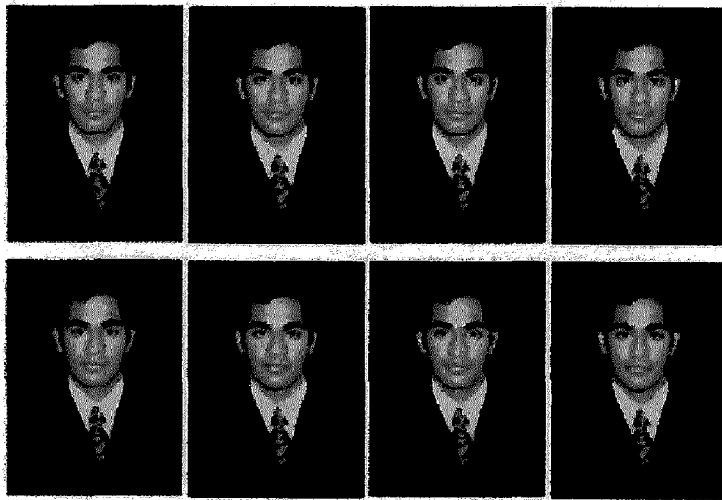
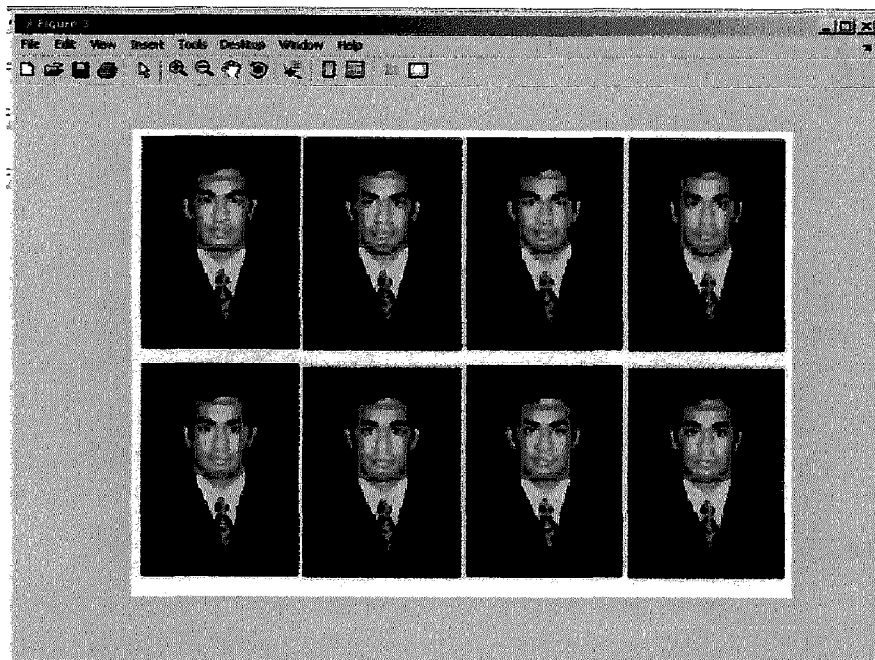


Fig 6.6: The training performance for the Gradient descent with momentum and adaptive learning rate Backpropagation training algorithm with 0.8 learning rate, which shows that the target is not reached for 5000 Epochs.

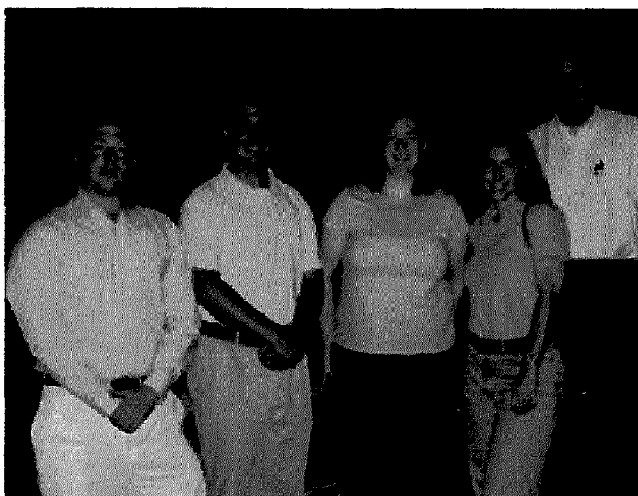


(a)

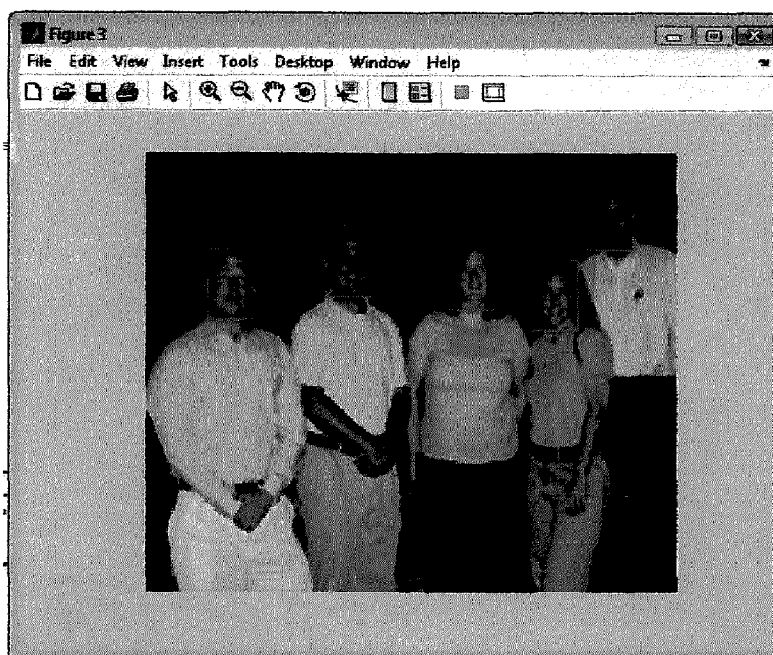


(b)

Fig 6.7: (a) Original image which is taken from the digital camera, (b) Output obtained from the system. The total faces in the image are six and the detected faces are six hence no false detection.



(a)



(b)

Fig 6.8: (a) Original image which is taken from the internet, (b) Output obtained from the system in which there is no false detection.



(a)

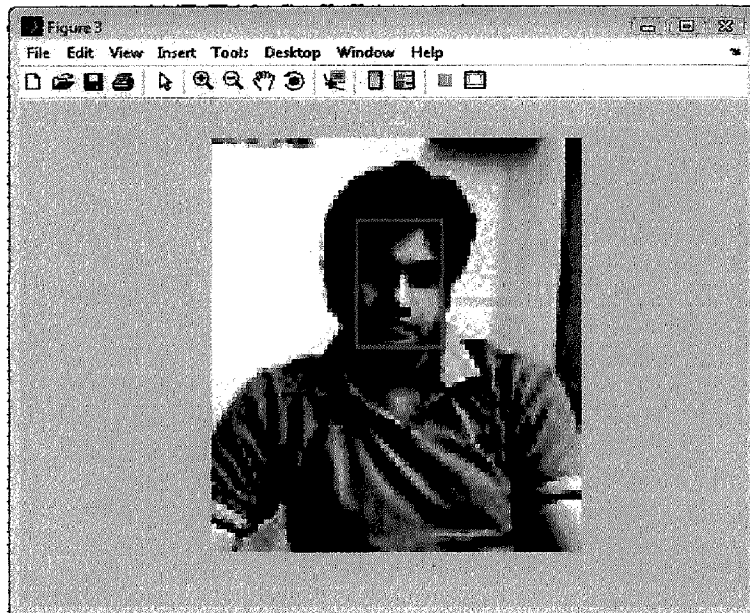


(b)

Fig 6.9: (a) Original image, (b) Output obtained from the system. Totally there are eight faces but only seven faces are detected due to blurriness.



(a)

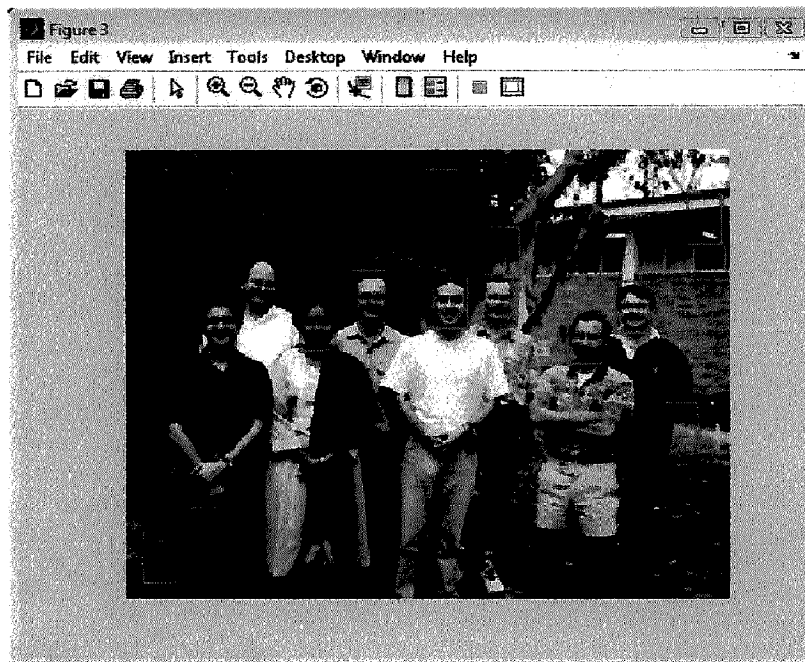


(b)

Fig 6.10: (a) Original image taken from the web camera, (b) Output obtained from the system



(a)

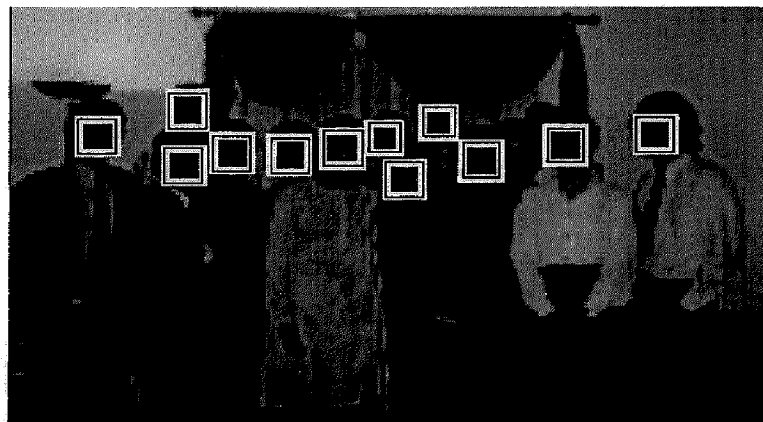


(b)

Fig 6.11: (a) Original image, (b) Output obtained from the system in which false detections are present.



(a)

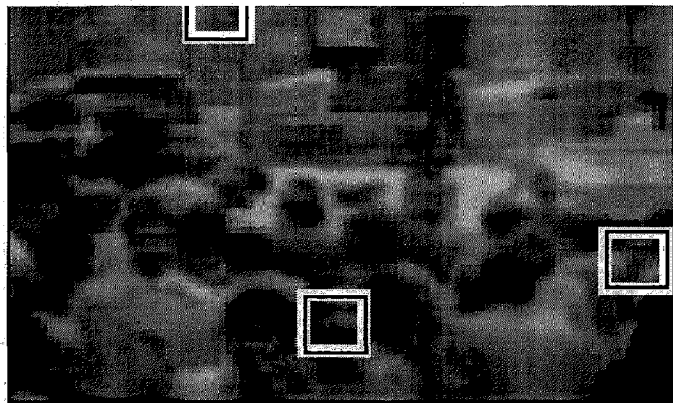


(b)

Fig 6.12: (a) Original image, (b) output image from the system which has false detections



(a)



(b)

Fig 6.13: (a) Original image, (b) Output image from the system which has maximum false detection due to blurriness.

CONCLUSION AND FUTURE WORK

In this thesis, a new approach to face detection with Gabor filters & feed forward neural network is presented. The method uses Gabor feature extraction & feed forward neural network for both finding feature points and extracting feature vectors. From the experimental results, it is seen that proposed method achieves better results compared to the graph matching and eigenface methods. Feature points are obtained from the special characteristics of each individual face automatically, instead of fitting a graph that is constructed from the general face idea. In the proposed algorithm, since the facial features are compared locally, instead of using a general structure, it allows us to make a decision from the parts of the face.

Future Work

Although detection performance of the proposed method is very good, the main limitation of the system is that it only detects upright faces looking at the camera. So further it can be improved by training the separate versions of the system for each head orientation, and the results could be combined to get the detection of faces for different poses and orientations.

REFERENCES

1. H. A. Rowley, S. Baluja, T. Kanade, "Neural Network-Based Face Detection", *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol.20, No. 1, Page(s). 39-51, 1998
2. Y. Ming-Hsuan, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 34-58, 2002.
3. H. Rein-Lien, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 696-706, 2002.
4. Lamiaa Mostafa, Sharif Abdelazeem " Face Detection Based on Skin Color Using Neural Networks" in *GVIP 05 Conference*, pp19-21 ,Dec 2006, CICC, Cairo, Egypt
5. F.Smach, M.Atri, J.Miteran and M.Abid " Design of a Neural Networks Classifier for Face Detection" in *Journal of computer Science* 2(3):pp257- 260,2006
6. Xuewen Wang, Xiaoqing Ding, Changsong Liu "Gabor filters-based feature extraction for character recognition" in *journal of pattern recognition* 38 (2005) 369 – 379.
7. Antonio J. Colmenarez and Thomas S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, 1997.
8. Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

9. Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *Fifth International Conference on Computer Vision*, pages 786–793, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.
10. S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decisionbased neural network. *IEEE Transactions on Neural Networks, Special Issue on Artificial Neural Networks and Pattern Recognition*, 8(1), January 1997.
11. Gilles Burel and Dominique Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.
12. Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition*, pages 130–136, 1997.
13. R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4), August 1994.
14. Gaungzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.
15. Kin Choong Yow and Roberto Cipolla. Feature-based human face detection. Technical Report CUED/F-INFENG/TR 249, Department of Engineering, University of Cambridge, England, 1996.
16. C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford United Press, NewYork, 1995.
17. T. Acharya and A. Ray, *Image Processing: Principles and Applications*, Wiley Publications, 2005.

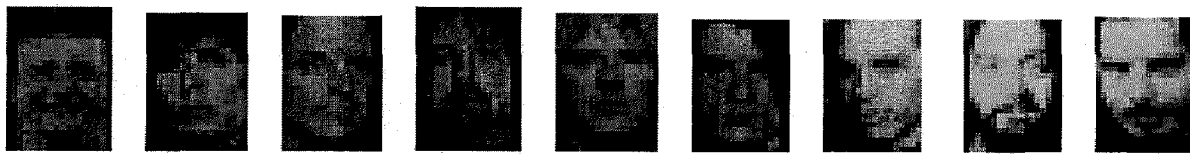
18. Lamiaa Mostafa, Sharif Abdelazeem " Face Detection Based on Skin Color Using Neural Networks" in GVIP 05 Conference, pp19-21 ,Dec 2006, CICC, Cairo, Egypt.
19. R. Gonzalez and R. Woods, "Digital Image Processing" (Second Edition), Pearson Education.,2005
20. F. Lin, X, Yu, S. Gregor, and R. Irons, "Time Series Forecasting with Neural Networks", Central Queensland University, Australia, 1995. Scientific Papers, Complexity International, Monash University, Australia, 1996.
21. Zhang ZhenQiu, Zhu Long, S.Z. Li, Zhang Hong Jiang, "Real-time multi-view face detection" Proceeding of the Fifth IEEE International Conference on automatic Face and Gesture Recognition, Page(s): 142-147, 20-21 May 2002.
22. M. Sadiku and M. Mazzara, "Computing with Neural Networks," *IEEE Potentials*, October 1993, pp. 14
23. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Pearson Education 1994.
24. M. Ibnkkahla, "Application of Neural Networks to digital communications- a survey," *Signal Processing*, Published by Elsevier Science, pp. 1185-1215, November 2000.
25. M. H. Ahmad Fadzil and H. Abu Bakar, "Human Face Recognition using Neural Networks," IEEE, 0-8186-6950-0, 1994.
26. M. Zhang and J. Fulcher, "Face Recognition using Artificial Neural Network Group-Based Adaptive Tolerance (GAT)," *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, may 1996.

27. J. Dalton and A. Deshmane, "Artificial neural networks," *IEEE Potentials*, pp. 34, April 1991.
28. A. Doulamis, N. Doulamis, and S. Koffias, "On-line Retractable Neural Networks: Improving the performance of Neural Networks in Image Analysis Problem," *IEEE Transactions on Neural Networks*, Vol. II, No. 1, January 2000.
29. R.P. Lippmann, "An Introduction to computing with neural nets," *IEEE ASSP Magazine*, Vol.4, pp. 4-22, 1987.
30. Howard Demuth, Mark Beale, "Neural Network Toolbox for use with MATLAB," The Mathworks Inc.
31. Fan Yang and Michel Paindavoine, "Prefiltering for pattern Recognition Using Wavelet Transform and Neural Networks", *Advances in imaging and Electron physics*, Vol. 127, 2003.
32. Sqn Ldr Prashant Srinivastava, "A Neural Network Based Face Recognition System" a dissertation report at IIT Roorkee, 2007.
33. Rajesh Reddy Kalwakuntla, "Face Recognition of Color Images Using Artificial Neural Networks" a thesis report at Texas A&M University, Kingsville, 2005

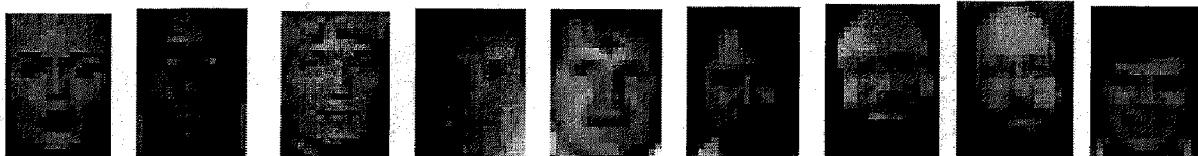
Paper submitted for conference:

S.Lingamiah, M.J.Nigam,"Face detection for two dimensional images using Gabor feature extraction and Neural Networks" *International conference on Neural Networks*, Amsterdam The Netherlands. Sept 28-30. 2010. (under review)

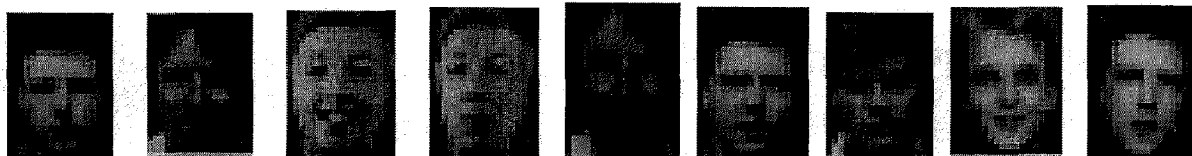
Appendix-1: Face Data Base for Training



1 2 3 4 5 6 7 8 9



10 11 12 13 14 15 16 17 18



19 20 21 22 23 24 25 26 27



28 29 30 31 32 33 34 35 36



37 38 39 40 41 42 43 44 45



46



47



48



49



50



51



52



53



54



55



56



57



58



59



60



61



62



63



64



65



66



67



68

Appendix-2: Non Face Data Base for Training





41



42



43



44



45



46



47



48



49



50



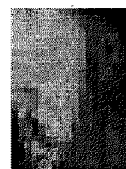
51



52



53



54



55