CHAPTER-I

INTRODUCTION

1.1 BIO-METRIC AUTHENTICATION

Biometrics is automated method of identifying a person or verifying the identity of a person based on a physiological or behavioral characteristic. Examples of physiological characteristics include hand or finger images, facial characteristics.

Biometric authentication requires comparing a registered or enrolled biometric sample (biometric template or identifier) against a newly captured biometric sample (for example, captured image during a login). During, as shown in the picture below, a sample of the biometric trait is captured, processed by a computer, and stored for later comparison.

Biometric recognition can be used in mode, where the biometric system identifies a person from the entire enrolled population by searching a database for a match based solely on the biometric. Sometime identification is called "one-to-many" matching.A system can also be used in mode, where the biometric system authenticates a person’s claimed identity from their previously enrolled pattern. This is also called “one-to-one” matching. In most computer access or network access environments, verification mode would be used.

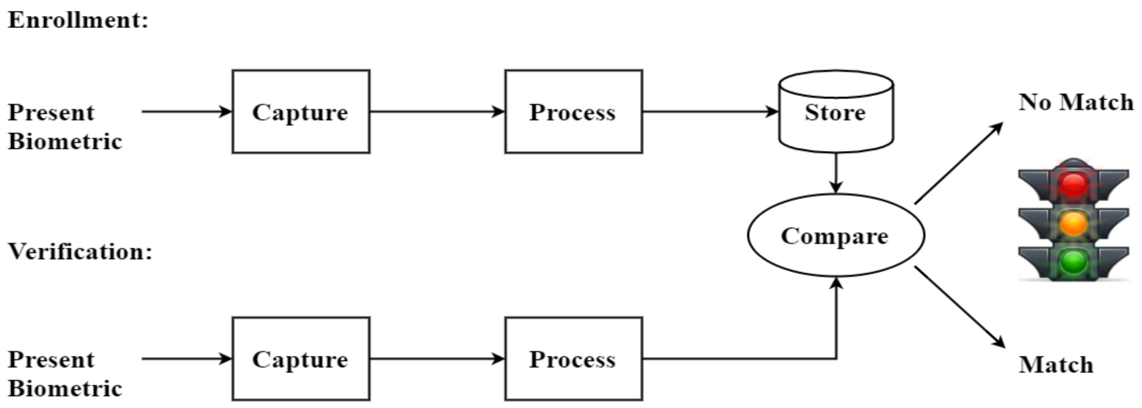


Fig 1.1: Sample process for Bio metric recognition

1.2 TYPES OF BIO-METRICS

1.2.1 FACE RECOGNITION

The identification of a person by their facial image can be done in a number of different ways such as by capturing an image of the face in the visible spectrum using an inexpensive camera or by using the infrared patterns of facial heat emission. Facial recognition in visible light typically model key features from the central portion of a facial image Using a wide assortment of cameras, the visible light systems extract features from the captured image that do not change over time while avoiding superficial features such as facial expressions or hair. Several approaches to modeling facial images in the visible spectrum are Principal Component Analysis, Local Feature Analysis, neural networks, elastic graph theory, and multi-resolution analysis.

Some of the challenges of facial recognition in the visual spectrum include reducing the impact of variable lighting and detecting a mask or photograph. Some facial recognition systems may require a stationary or posed user in order to capture the image, though many systems use a real-time process to detect a person's head and locate the face automatically. Major benefits of facial recognition are that it is non-intrusive, hands-free, and continuous and accepted by most users.

1.2.2 FINGERPRINTS

Fingerprints are unique for each finger of a person including identical twins. One of the most commercially available biometric technologies, fingerprint recognition devices for desktop and laptop access are now widely available from many different vendors at a low cost. With these devices, users no longer need to type passwords – instead, only a touch provides instant access.

Fingerprint systems can also be used in identification mode. Several states check fingerprints for new applicants to social services benefits to ensure recipients do not fraudulently obtain benefits under fake names.

1.2.3 IRIS RECOGNITION

This recognition method uses the iris of the eye, which is the colored area that surrounds the pupil. Iris patterns are thought unique. The iris patterns are obtained through a video-based image acquisition system. Iris scanning devices have been used in personal authentication applications for several years. Systems based on iris recognition have substantially decreased in price and this trend is expected to continue.

The technology works well in both verification and identification modes (in systems performing one-to-many searches in a database). Current systems can be used even in the presence of eyeglasses and contact lenses. The technology is not intrusive. It does not require physical contact with a scanner. Iris recognition has been demonstrated to work with individuals from different ethnic groups and nationalities.

1.2.4 SIGNATURE VERIFICATION

This technology uses the dynamic analysis of a signature to authenticate a person. The technology is based on measuring speed, pressure and angle used by the person when a signature is produced. One focus for this technology has been e-business applications and other applications where signature is an accepted method of personal authentication.

1.2.5 SPEAKER RECOGNITION

Speaker recognition has a history dating back some four decades, where the outputs of several analog filters were averaged over time for matching. Speaker recognition uses the acoustic features of speech that have been found to differ between individuals.

These acoustic patterns reflect both anatomy (e.g., size and shape of the throat and mouth) and learned behavioral patterns (e.g., voice pitch, speaking style). This incorporation of learned patterns into the voice templates (the latter called "voiceprints") has earned speaker recognition its classification as a "behavioral biometric." Speaker recognition systems employ three styles of spoken input: text-dependent, text-prompted and text independent.

Most speaker verification applications use text-dependent input, which involves selection and enrollment of one or more voice passwords. Text-prompted input is used whenever there is concern of imposters. The various technologies used to process and store voiceprints include hidden Markov models; pattern matching algorithms, neural networks, and matrix representation and decision trees. Some systems also use "anti-speaker" techniques, such as cohort models, and world models.

Ambient noise levels can impede both collections of the initial and subsequent voice samples. Performance degradation can result from changes in behavioral attributes of the voice and from enrollment using one telephone and verification on another telephone. Voice changes due to aging also need to be addressed by recognition systems. Many companies market speaker recognition engines, often as part of large voice processing, control and switching systems.

Capture of the biometric is seen as non-invasive. The technology needs little additional hardware by using existing microphones and voice-transmission technology allowing recognition over long distances via ordinary telephones (wire line or wireless).

In this project we concentrate on face recognition approach out of these biometric approaches.

CHAPTER - II

LITARACY OF SURVEY

2.1 IMPROVING THE SPEED

In this section we briefly discuss some methods to improve the speed of the system. The work described is preliminary, and is not intended to be an exhaustive exploration of methods to optimize the execution time.

The dominant factor in the running time of the system described thus far is the number of 20x20 pixel windows which the neural networks must process. Applying two networks to a 320x240 pixel image (246766 windows) on a 200 MHz R4400 SGI Indigo 2 takes approximately 383 seconds. The computational cost of the arbitration steps is negligible in comparison, taking less than one second to combine the results of the two networks over all positions in the image.

Recall that the amount of position invariance in the pattern recognition component of our system determines how many windows must be processed. In the related task of license plate detection, this was exploited to decrease the number of windows that must be processed . The idea was to make the neural network be invariant to translations of about 25% of the size of the license plate. Instead of a single number indicating the existence of a face in the window, the output of Umezaki's network is an image with a peak indicating the location of the license plate. These outputs are accumulated over the entire image, and peaks are extracted to give candidate locations for license plates.

The same idea can be applied to face detection. The original detector was trained to detect a 20x20 face centered in a 20x20 window. We can make the detector more flexible by allowing the same 20x20 face to be off-center by up to 5 pixels in any direction. To make sure the network can still see the whole face, the window size is increased to 30x30 pixels. Thus the center of the face will fall within a 10x10 pixel region at the center of the window.

As before, the network has a single output, indicating the presence or absence of a face. This detector can be moved in steps of 10 pixels across the image, and still detect all faces that might be present.

The scanning method is illustrated in Fig. which shows the input image pyramid and which of the 10x10 pixel regions are classified as containing the centers of faces. An architecture with an image output was also tried, which yielded about the same detection accuracy, but required more computation. The network was trained using the same bootstrap procedure as described earlier. The windows are preprocessed with histogram equalization before they are passed to the network.

As can be seen from the figure, this network has many more false detections than the detectors described earlier. To improve the accuracy, we treat each detection by the 30x30 detector as a candidate, and use the 20x20 detectors described earlier to verify it. Since the candidate faces are not precisely located, the verification network's 20x20 indow must be scanned over the 10x10 pixel region potentially containing the center of the face.

A simple arbitration strategy, ANDing, is used to combine the outputs of two verification networks. The heuristic that faces rarely overlap can also be used to reduce computation, by first scanning the image for large faces, and at smaller scales not processing locations which overlap with any detections found so far.

With these modifications, the processing time for a typical 320x240 image is about 7.2 seconds on a 200 MHz R4400 SGI Indigo 2. To examine the effect of these changes on the accuracy of the system, it was applied to the test sets used in the previous section.

Further performance improvements can be made if one is analyzing many pictures taken by a stationary camera. By taking a picture of the background scene, one can determine which portions of the picture have changed in a newly acquired image, and analyze only those portions of the image. Similarly, a skin color detector like the one presented in can restrict the search region.

These techniques, taken together, have proven useful in building an almost real-time version of the system suitable for demonstration purposes, which can process a 320x240 image in 2 to 4 seconds, depending on the image complexity.

2.2 COMPARISION TO THE OTHER SYSTEMS

Sung and Poggio developed a face detection system based on clustering techniques. Their system, like ours, passes a small window over all portions of the image, and determines whether a face exists in each window. Their system uses a supervised clustering method with six “face” and six “non-face” clusters. Two distance metrics measure the distance of an input image to the prototype clusters, the first measuring the distance between the test pattern and the cluster’s 75 most significant eigenvectors, and the second measuring the Euclidean distance between the test pattern and its projection in the 75 dimensional subspace.

The last step in their system is to use either a perception or a neural network with a hidden layer, trained to classify points using the two distances to each of the clusters. Their system is trained with 4000 positive examples and nearly 47500 negative examples collected in the bootstrap manner. In comparison, our system uses approximately 16000 positive examples and 9000 negative examples.

In, 149 faces were labeled in this test set, while we labeled 155. Some of these faces are difficult for either system to detect. Assuming that Sung and Poggio were unable to detect any of the six additional faces we labeled, the number of faces missed by their system is six more than listed in their paper.

Osuna, Freund, and Girosi have recently investigated face detection using a framework similar to that used in and in our own work. However, they use a “support vector machine” to classify images, rather than a clustering-based method or a neural network. The support vector machine has a number of interesting properties, including the fact that it makes the boundary between face and non face images more explicit. The result of their system on the same 23 images used in is given in Table 4.1; the accuracy is currently slightly poorer than the other two systems for this small test set.

As with Sung and Poggio’s work, Moghoddam and Pentland’s approach uses a two component distance measure, but combines the two distances in a principled way based on the assumption that the distribution of each cluster is Gaussian [7]. The clusters are used together as a multi-modal Gaussian distribution, giving a probability distribution for all face images. Faces are detected by measuring how well each window of the input image fits the distribution, and setting a threshold. This detection technique has been applied to faces, and to the detection of smaller features like the eyes, nose, and mouth.

Moghaddam and Pentland’s system, along with several others, was tested in the FERET evaluation of face recognition methods [9,10]. Although the actual detection error rates are not reported, an upper bound can be derived from the recognition error rates. The recognition error rate, averaged over all the tested systems, for frontal photographs taken in the same sitting is less than 2% (see the rank 50 results in Figure 4 of . This means that the number of images containing detection errors, either false alarms or missing faces, was less than 2% of all images.

Anecdotally, the actual error rate is significantly less than 2%. Our system using the configuration of System 11 achieves a 2% error rate on frontal faces. Given the large differences in performance of our system on Test Set 1 and the FERET images, it is clear that these two test sets exercise different portions of the system. The FERET images examine the coverage of a broad range of face types under good lighting with uncluttered backgrounds, while Test Set 1 tests the robustness to variable lighting and cluttered backgrounds.

The candidate verification process used to speed up our system, described in Section 4, is similar to the detection technique presented in. In that work, two networks were used. The first network has a single output, and like our system it is trained to produce a positive value for centered faces, and a negative value for non-faces. Unlike our system, for faces that are not perfectly centered, the network is trained to produce an intermediate value related to how far off-center the face is. This network scans over the image to produce candidate face locations.

The network must be applied at every pixel position, but it runs quickly because of the its architecture: using retinal connections and shared weights, much of the computation required for one application of the detector can be reused at the adjacent pixel position. This optimization requires the preprocessing to have a restricted form, such that it takes as input the entire image, and produces as output a new image. The nonlinear window-by-window preprocessing used in our system cannot be used.

A second network is used for precise localization it is trained to produce a positive response for an exactly centered face, and a negative response for faces which are not centered[2]. It is not trained at all on non faces. All candidates which produce a positive response from the second network are output as detections. One possible problem with this work is that the negative training examples are selected manually from a small set of images (indoor scenes, similar to those used for testing the system).

It may be possible to make the detectors more robust using the bootstrap training technique described here and in.

In recent work, Colmenarez and Huang presented a statistically based method for face detection[3]. Their system builds probabilistic models of the sets of faces and non-faces, and compares how well each input window compares with these two categories. When applied to Test Set 1, their system achieves a detection rate between 86.8% and 98.0%, with between 6133 and 12758 false detections, respectively, depending on a threshold. These numbers should be compared to Systems 1 through 4 in Table 1, which have detection rates between 90.9% and 92.1%, with between 738 and 945 false detections. Although their false alarm rate is significantly higher, their system is quite fast. It would be interesting to use this system as a replacement for the candidate detector.

CHAPTER-III

NEURAL NETWORK BASED FACE RECOGNITION

3.1 INTRODUCTION

In this paper, we present a neural network-based algorithm to recognize upright, frontal views of faces in gray-scale images. The algorithm works by applying one or more neural networks directly to portions of the input image, and arbitrating their results. Each network is trained to output the presence or absence of a face. The algorithms and training methods are designed to be general, with little customization for faces.

Many face recognition researchers have used the idea that facial images can be characterized directly in terms of pixel intensities. These images can be characterized by probabilistic models of the set of face images or implicitly by neural networks or other mechanisms . The parameters for these models are adjusted either automatically from example images (as in our work) or by hand. A few authors have taken the approach of extracting features and applying either manually or automatically generated rules for evaluating these features.

Training a neural network for the face recognition task is challenging because of the difficulty in characterizing prototypical “non face” images. Unlike face recognition, in which the classes to be discriminated are different faces, the two classes to be discriminated in face recognition are “images containing faces” and “images not containing faces”. It is easy to get a representative sample of images which contain faces, but much harder to get a representative sample of those which do not.

We avoid the problem of using a huge training set for non faces by selectively adding images to the training set as training progresses. This “bootstrap” method reduces the size of the training set needed. The use of arbitration between multiple networks and heuristics to clean up the results significantly improves the accuracy of the detector.

3.2 Description of the System

Our system operates in two stages: It first applies a set of neural network-based filters to an image, and then uses an arbitrator to combine the outputs. The filters examine each location in the image at several scales, looking for locations that might contain a face. The arbitrator then merges detections from individual filters and eliminates overlapping detections.

Stage One: A Neural Network-Based Filter

The first component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To recognize faces anywhere in the input, the filter is applied at every location in the image. To recognize faces larger than the window size, the input image is repeatedly reduced in size (by sub sampling), and the filter is applied at each size. This filter must have some invariance to position and scale. The amount of invariance determines the number of scales and positions at which it must be applied. For the work presented here, we apply the filter at every pixel position in the image, and scale the image down by a factor of 1.2 for each step in the pyramid.

The filtering algorithm is shown in Fig. 3.1. First, a preprocessing step, adapted from is applied to a window of the image. The window is then passed through a neural network, which decides whether the window contains a face. The preprocessing first attempts to equalize the intensity values in across the window.

We fit a function which varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval (shown in Fig. 3.1) may represent the background, so those intensity values are ignored in computing the lighting variation across the face. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions.

Then histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases. The preprocessing steps are shown in Fig. 3.2.

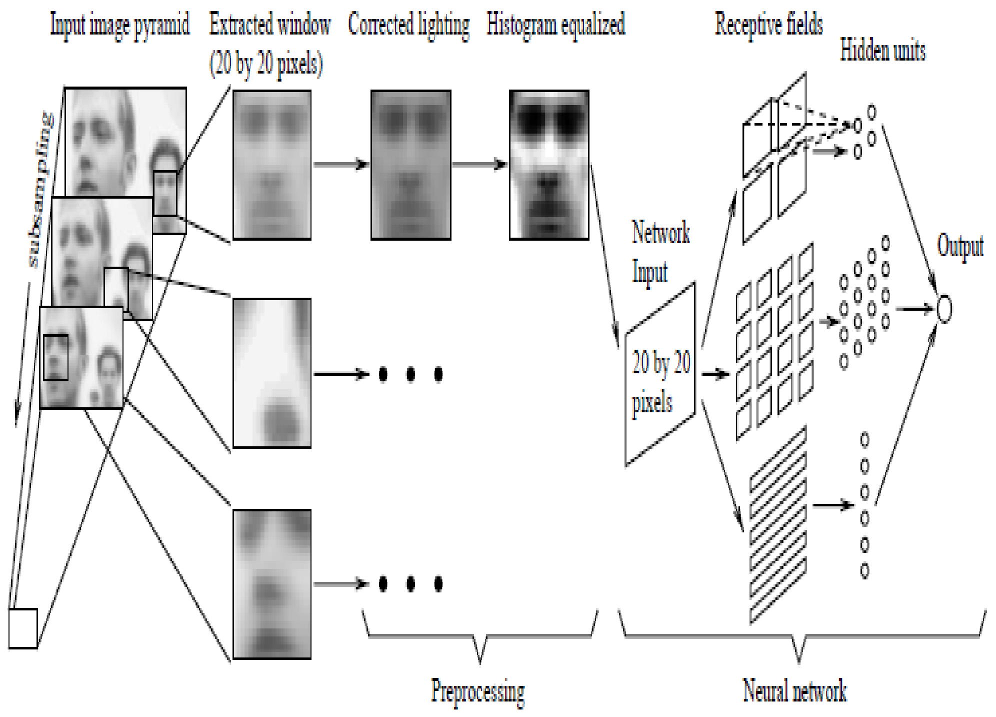


Fig 3.1: The Basic algorithm for Face Recognition System using Neural Networks

The preprocessed window is then passed through a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Fig 3.1.

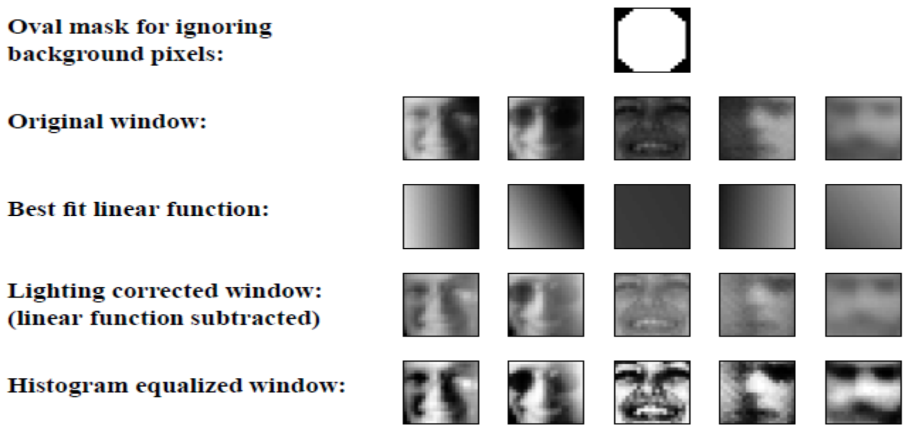

Oval mask for ignoring background pixels:
Original window:
Best fit linear function:
Lighting corrected window: (linear function subtracted)
Histogram equalized window:

Fig 3.2 : The Steps in Pre Processing a window

There are three types of hidden units: 4 which look at 10x10 pixel sub regions, 16 which look at 5x5 pixel sub regions, and 6 which look at overlapping 20x5 pixel horizontal stripes of pixels. Each of these types was chosen to allow the hidden units to detect local features that might be important for face detection.

In particular, the horizontal stripes allow the hidden units to detect such features as mouths or pairs of eyes, while the hidden units with square receptive fields might detect features such as individual eyes, the nose, or corners of the mouth. Although the figure shows a single hidden unit for each sub region of the input, these units can be replicated. For the experiments which are described later, we use networks with two and three sets of these hidden units. Similar input connection patterns are commonly used in speech and character recognition tasks[11] .The network has a single, real-valued output, which indicates whether or not the window contains a face.

Examples of output from a single network are shown in Fig. 3.3. In the figure, each box represents the position and size of a window to which the neural network gave a positive response.

The network has some invariance to position and scale, which results in multiple boxes around some faces. Note also that there are some false detections; they will be eliminated by methods presented in Section 3.2.

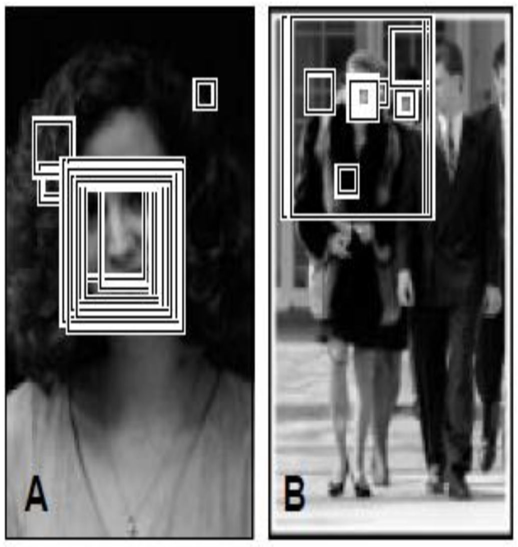


Fig 3.3: Images with the above threshold values are indicated by boxes

To train the neural network used in stage one to serve as an accurate filter, a large number of face and non face images are needed. Nearly 1050 face examples were gathered from face databases at CMU, Harvard2, and from the World Wide Web. The images contained faces of various sizes, orientations, positions, and intensities. The eyes, tip of nose, and corners and center of the mouth of each face were labeled manually.

These points were used to normalize each face to the same scale, orientation, and position, as follows:

1. Initialize _ F, a vector which will be the average positions of each labeled feature over all the faces, with the feature locations in the first face F1.
2. The feature coordinates in _ F are rotated, translated, and scaled, so that the average locations of the eyes will appear at predetermined locations in a 20x20 pixel window.

3. For each face i, compute the best rotation, translation, and scaling to align the face's features with the average feature locations _ F. Such transformations can be written as a linear function of their parameters. Thus, we can write a system of linear equations mapping the features from Fi to _ F. The least squares solution to this over-constrained system yields the parameters for the best alignment transformation. Call the aligned feature locations F0

4. Update _ F by averaging the aligned feature locations F0 I for each face i.

5. Go to step 2.

The alignment algorithm converges within five iterations, yielding for each face a function which maps that face to a 20x20 pixel window. Fifteen face examples are generated for the training set from each original image, by randomly rotating the images (about their center points) up to 10_, scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20x20 window in the set is then preprocessed (by applying lighting correction and histogram equalization).

A few example images are shown in Fig 3.4. The randomization gives the filter invariance to translations of less than a pixel and scaling of 20%. Larger changes in translation and scale are dealt with by applying the filter at every pixel position in an image pyramid, in which the images are scaled by factors of 1.2.

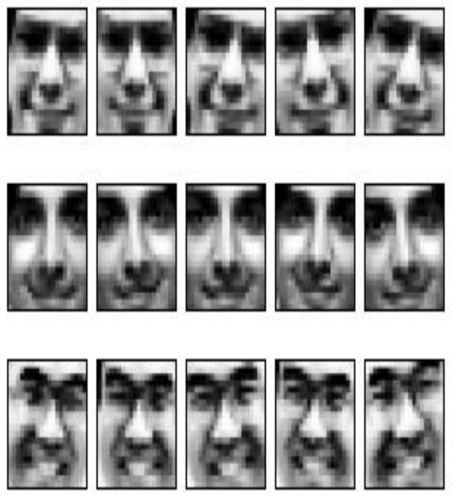


Fig 3.4: Examples of the face images randomly mirrored, rotated, translated and scaled by small amounts

Practically any image can serve as a non-face example because the space of non-face images is much larger than the space of face images. However, collecting a “representative” set of non faces is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner:

1. Create an initial set of non face images by generating 1000 random images. Apply the preprocessing steps to each of these images.
2. Train a neural network to produce an output of 1 for the face examples, and -1 for the non face examples. The training algorithm is standard error back propagation with momentum . On the first iteration of this loop, the network’s weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.
3. Run the system on an image of scenery *which contains no faces*. Collect sub images in which the network incorrectly identifies a face (an output activation > 0).
4. Select up to 250 of these sub images at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

Some examples of non faces that are collected during training are shown in Fig 3.5. Note that some of the examples resemble faces, although they are not very close to the positive examples shown in Fig 3.4. The presence of these examples forces the neural network to learn the precise boundary between face and non face images. We used 120 images of scenery for collecting negative examples in the bootstrap manner described above. A typical training run selects approximately 8000 non face images from the 146,212,178 sub images that are available at all locations and scales in the training scenery images.

A similar training algorithm was described in , where at each iteration an entirely new network was trained with the examples on which the previous networks had made mistakes.

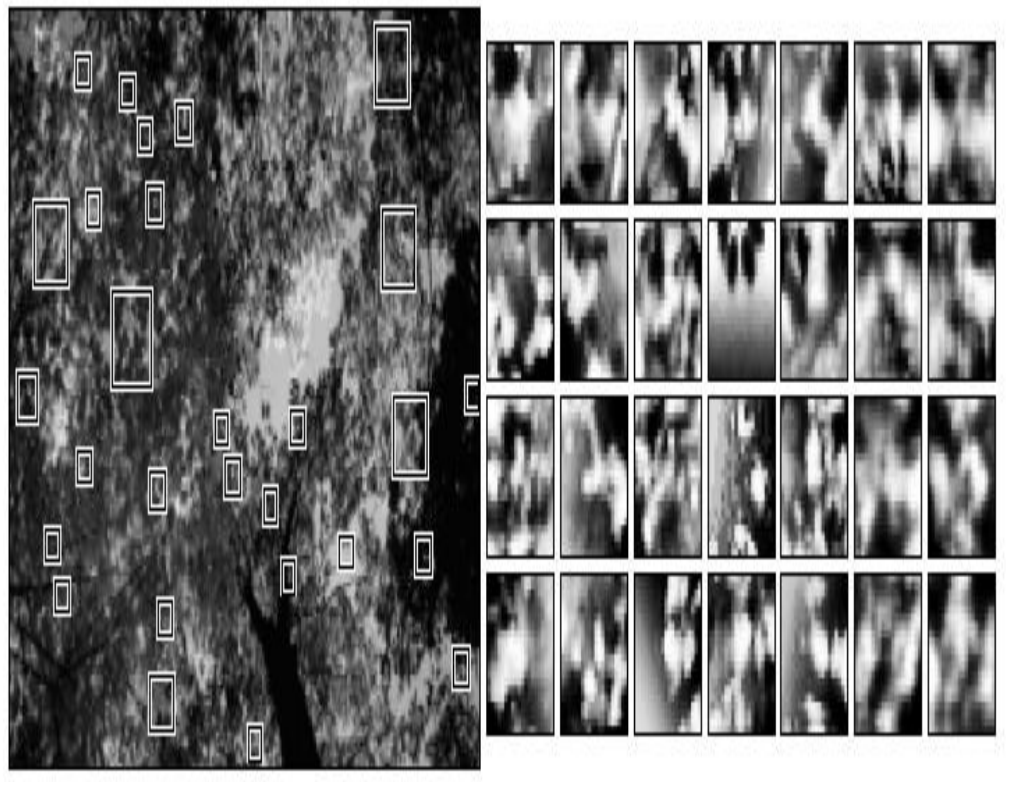


Fig 3.5: During training, the partially-trained system is applied to images of scenery which do not contain faces.

Any regions in image recognized as face are errors and added into set of negative training.

Stage Two: Merging Overlapping Detections and Arbitration

The examples in Fig 3.3 showed that the raw output from a single network will contain a number of false detections. In this section, we present two strategies to improve the reliability of the detector: merging overlapping detections from a single network and arbitrating among multiple networks.

3.3 MERGING OVERLAPPING DETECTIONS

Note that in Fig 3.3, most faces are detected at multiple nearby positions or scales, while false detections often occur with less consistency. This observation leads to a heuristic which can eliminate many false detections.

For each location and scale, the number of detections within a specified neighborhood of that location can be counted. If the number is above a threshold, then that location is classified as a face. The centroid of the nearby detections defines the location of the detection result, thereby collapsing multiple detections. In the experiments section, this heuristic will be referred to as "thresholding".

If a particular location is correctly identified as a face, then all other detection locations which overlap it are likely to be errors, and can therefore be eliminated. Based on the above heuristic regarding nearby detections, we preserve the location with the higher number of detections within a small neighborhood, and eliminate locations with fewer detections. In the discussion of the experiments, this heuristic is called "overlap elimination". There are relatively few cases in which this heuristic fails; however, one such case is illustrated by the left two faces in Fig 3.3, where one face partially occludes another.

The implementation of these two heuristics is illustrated in Fig 3.6. Each detection at a particular location and scale is marked in an image pyramid, labeled the "output" pyramid. Then, each location in the pyramid is replaced by the number of detections in a specified neighborhood of that location.

This has the effect of "spreading out" the detections. Normally, the neighborhood extends an equal number of pixels in the dimensions of scale and position, but for clarity in Fig. 3.6 detections are only spread out in position. A threshold is applied to these values, and the centroids (in both position and scale) of all above threshold regions are computed.

All detections contributing to a centroid are collapsed down to a single point. Each centroid is then examined in order, starting from the ones which had the highest number of detections within the specified neighborhood. If any other centroid locations represent a face overlapping with the current centroid, they are removed from the output pyramid. All remaining centroid locations constitute the final detection result. In the face detection work described in, similar observations about the nature of the outputs were made, resulting in the development of heuristics similar to those described above.

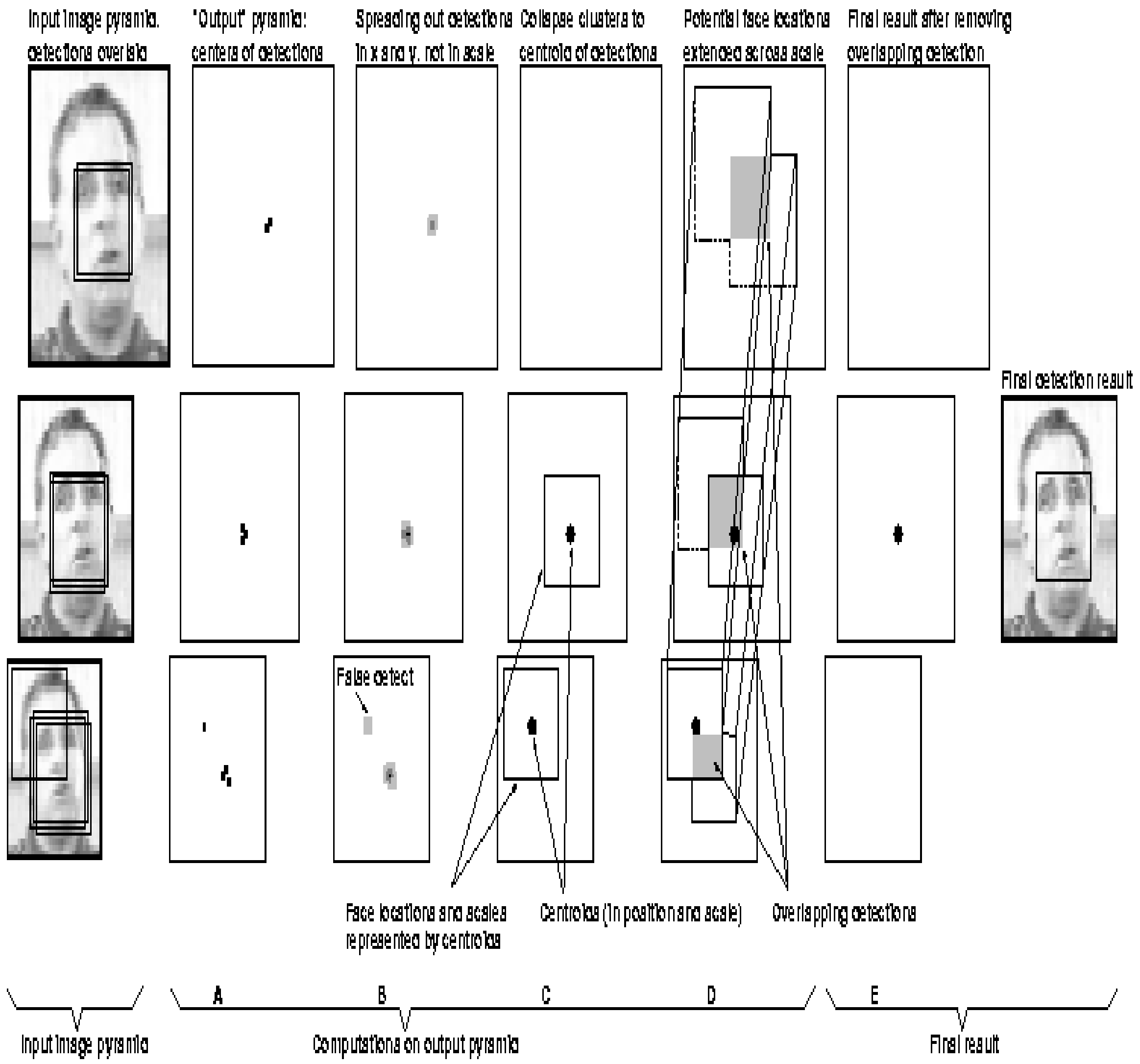

Input image pyramid, detections overlaid
"Output" pyramid: centers of detections
Spreading out detections in x and y, not in scale
Collapse clusters to centroids of detections
Potential face locations extended across scale
Final result after removing overlapping detection
Final detection result
False detect
Face locations and scales represented by centroids
Centroids (in position and scale)
Overlapping detections
A
B
C
D
E
Input image pyramid
Computations on output pyramid
Final result

Fig 3.6: Multiple detections from a single network

The framework used for merging multiple detections from a single network:

A) The detections are recorded in an image pyramid.
B) The detections are ``spread out" and a threshold is applied.
C) The centroids in scale and position are computed, and the regions contributing to each centroid are collapsed to single points. In the example shown, this leaves only two detections in the output pyramid.
D) The final step is to check the proposed face locations for overlaps, and
E) to remove overlapping detections if they exist. In this example, removing the overlapping detection eliminates what would otherwise be a false positive.

3.4 ARBITRATION AMONG MULTIPLE NETWORKS

To further reduce the number of false positives, we can apply multiple networks, and arbitrate between their outputs to produce the final decision. Each network is trained in a similar manner, but with random initial weights, random initial non-face images, and permutations of the order of presentation of the scenery images. As will be seen in the next section, the detection and false positive rates of the individual networks will be quite close. However, because of different training conditions and because of self-selection of negative training examples, the networks will have different biases and will make different errors.

The implementation of arbitration is illustrated. Each detection at a particular position and scale is recorded in an image pyramid, as was done with the previous heuristics. One way to combine two such pyramids is by ANDing them. This strategy signals a detection only if both networks detect a face at precisely the same scale and position. Due to the different biases of the individual networks, they will rarely agree on a false detection of a face. This allows ANDing to eliminate most false detections. Unfortunately, this heuristic can decrease the detection rate because a face detected by only one network will be thrown out. However, we will see later that individual networks can all detect roughly the same set of faces, so that the number of faces lost due to ANDing is small.

Similar heuristics, such as ORing the outputs of two networks, or voting among three networks, were also tried. Each of these arbitration methods can be applied before or after the "thresholding" and "overlap elimination" heuristics. If applied afterwards, we combine the centroid locations rather than actual detection locations, and require them to be within some neighborhood of one another rather than precisely aligned.

Arbitration strategies such as ANDing, ORing, or voting seem intuitively reasonable, but perhaps there are some less obvious heuristics that could perform better. To test this hypothesis, we applied a separate neural network to arbitrate among multiple detection networks. For a location of interest, the arbitration network examines a small neighborhood surrounding that location in the output pyramid of each individual network.

For each pyramid, we count the number of detections in a 3x3 pixel region at each of three scales around the location of interest, resulting in three numbers for each detector, which are fed to the arbitration network, as shown. The arbitration network is trained to produce a positive output for a given set of inputs only if that location contains a face, and to produce a negative output for locations without a face. As will be seen in the next section, using an arbitration network in this fashion produced results comparable to (and in some cases, slightly better than) those produced by the heuristics presented earlier.

CHAPTER-IV

NEURAL NETWORK

In the overall design of the neural network you must make sure that the number of free parameters using in the neural network are less than the number of training examples [1]. If this does occur then over fitting will occur and it will become impossible for the neural network to learn. The other issue to be concerned about is making sure that the number of features that the neural network is going to have as input into the input layer is reduced; otherwise the size of the neural network will exceed the memory space of any server that the application would be running on.

4.1 NEURAL NETWORK ENSEMBLE

The Neural Network Ensemble is a collection of multilayer feed-forward neural network, which is a network with hidden neurons. These hidden neurons are contained within a hidden layer and within most common multilayer feed-forward neural networks (MNN) they typically only contain a single hidden layer as shown in Fig 4.1. The advantage of adding a hidden layer to the neural network is that it enlarges the space of hypotheses that the neural network can represent.

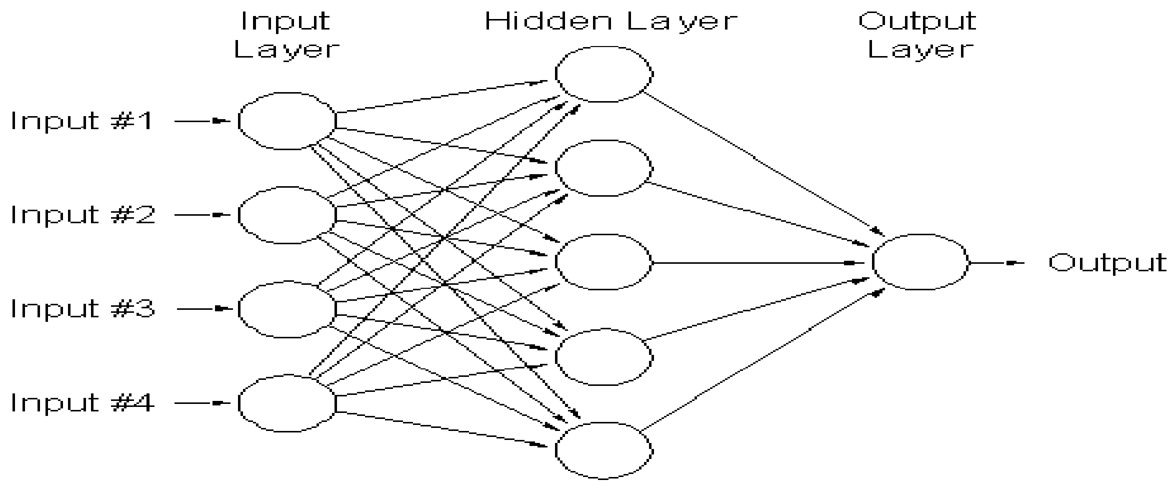


Fig 4.1: Multilayer Neural Network

As you can see in Fig 4.1 the output from the previous layer is the input for the next layer. Not all MNN have a single neuron within the output layer and may have more than one depending on the classification that the neural network is trying to classify on. In dealing with the domain of face recognition we will have a neuron in the output layer for each face or person that we are trying to classify.

In order to calculate the number of neurons needed in the Input Layer to classifying a face from an image is determined by the product of the row and column size of the image. For example if we have a 200x100 pixel image then we will need 20,000 neurons in the Input Layer of our MNN. With this many neurons in the Input Layer alone, this is the reason for have the Eigen faces or another way of reducing the search space. Finally we have the edges or transformations between the outputs of the neurons from the previous layer to the input of the neurons in the next layer.

Each neuron has an activation function that determines what the output of the neuron will be. In regards to pattern/face recognition it has show that the tan-sigmoid activation function with an output range of [-1, 1] is better than the log-sigmoid [6]. The value of the activation function is the value that is passed as the input to the next layer of the neural network. The closer the value is to 1 the higher the probability that the class associated with the neuron in the output layer is the class of the image that is being evaluated. Now that the architecture of the multilayer neural network has been described, we need to discuss the recommended method for training the MNN.

While training the MNN we will be determining the number of nodes contained within the hidden layer through experimentation. For example we can utilize cross-validation techniques, typically choosing the 10-fold cross-validation, to test the MNN with different number of nodes within the hidden layer to determine the number of nodes that gives us the most accurate predictions. When we are performing the training we are utilizing back propagation in order to determine the weights on the edges between the nodes of the different layers.

This is accomplished by back-propagating the error from the output layer back through the hidden layer. It has been found that the most efficient algorithm for back propagation for pattern face recognition according to the criteria of training process stability and recognition accuracy is the gradient descent method with momentum and adaptive learning rate [6]. Going through the process of back propagating the error to alter the weights in the neural network is called an epoch.

The system will continue to go through hundreds of epochs until the specified stopping criteria for changes in the weights has been met. This is one of the big advantages and disadvantages of the neural network and back propagation. The advantage is that the developer is letting the system tune itself by back propagating the error and altering the weights. The disadvantage is the amount of time it takes to reach the stopping criteria and the number of epochs that the system has to go through in order to meet that criterion. In section 3.4.2 we will look at a way to accelerate the back propagation problem.

Now that we have covered the background on multilayer neural networks, we need to look at the advantages of the ensemble of multilayer feed-forward neural network (EMNN). As stated before the EMNN is a collection of MNNs, where each has its weights initialized randomly and are trained in the same manner as discussed above. After they are trained and all of the MNNs have met the stopping criteria set for altering the weights during back propagation, they are ready to classify the images.

With the EMNN we are looking at having each individual MNN provide a vote on the classification of the image is and the EMNN will return the majority vote provided by the collection of MNNs. This is the main advantage of utilizing the EMNN architecture, since it increases the recognition rate compared to using a single MNN [6].

In Fig 4.2 we can see the benefits of using the multiple MNN in the EMNN over a single MNN. The results in Fig 4.2 show that having five or seven MNNs in the EMNN has a lower average recognition error. In comparing the EMNN (5xMNNs) vs. the EMNN (7xMNNs) we can see that there is not a drastic advantage between the two. So choosing the EMNN (5xMNNs) is selected due to the fact that the time for training the neural networks is less than the one with seven [6].

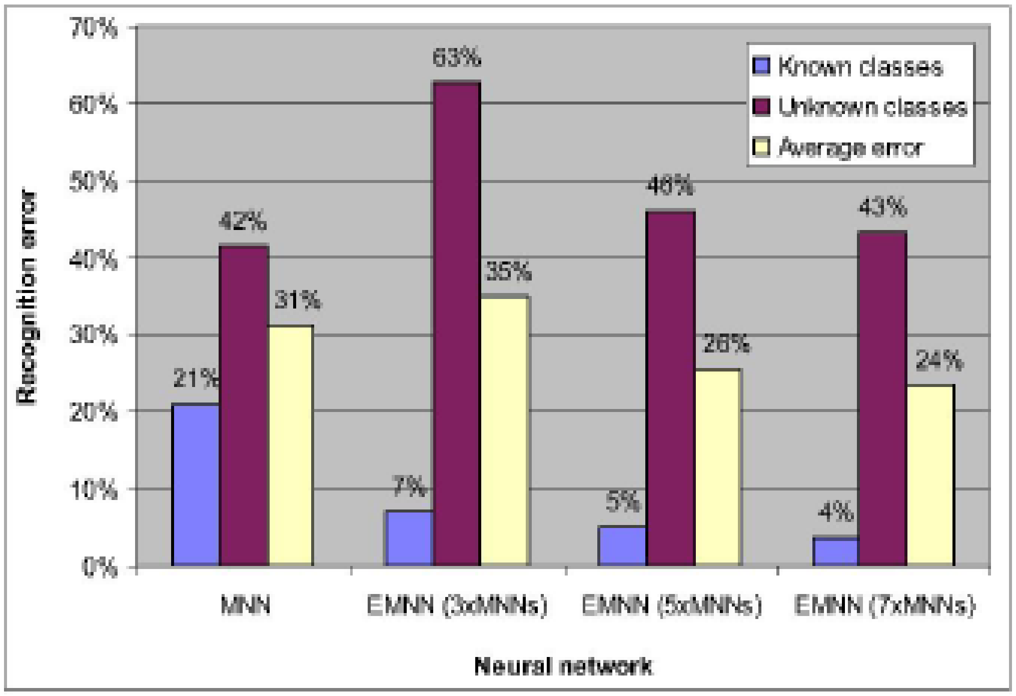


Fig 4.2: Face Recognition using MNN and EMNN

In order to improve on the high recognition error rate on unknown classes, a modification to the EMNN's decision rule is needed. Instead of just using the voting rule, it turns out that using a combination of the voting rule and a recognition threshold value work much better as discovered in [6]. The new decision rule for the EMNN is when one of the MNN's output layers output is within the majority, it must be greater than or equal to the threshold value [6].

This new decision rule is called 3t and the results are shown in Fig 4.3. As you can see the error rate for unknown classes has been reduced from 46% to 25% when the threshold value was 0.4. The average error went from 26% down to 17% when the threshold value was 0.4.

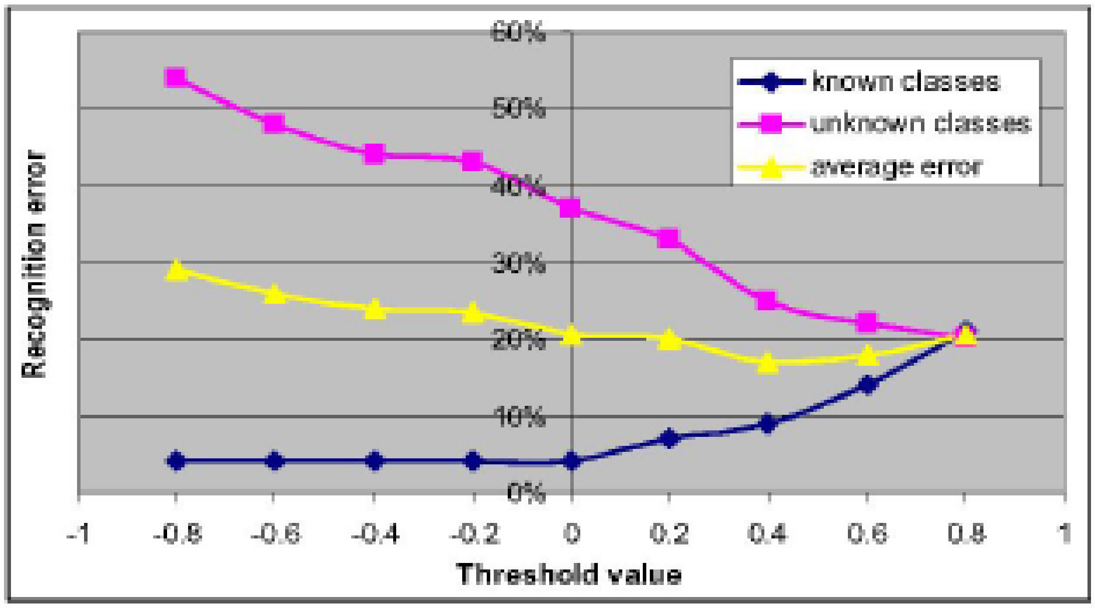


Fig 4.3: EMNN with 3t decision rule

After altering the decision rule again to be when the majority of the EMNNs contained the most of MNNs plus one and the threshold for one or two of the MNNs were used, there was another improvement in the results [6].

As you can see in Fig 4.4 the average recognition error went from a 17% in the previous altered decision rule to 13%.

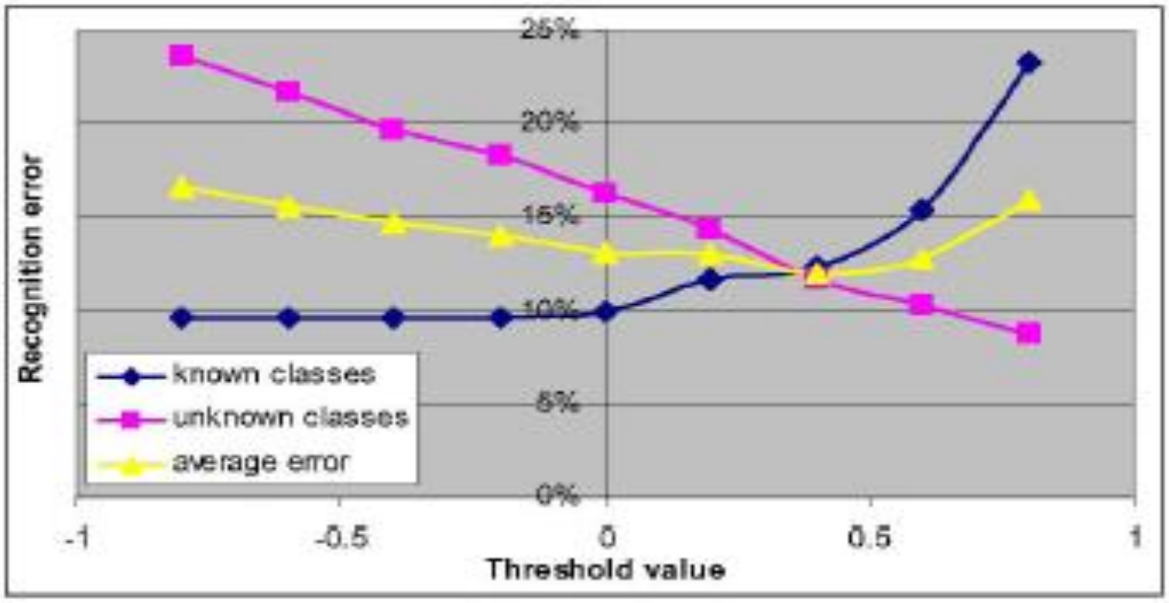


Fig 4.4: EMNN with 4t decision rule

Therefore, utilizing the EMNN architecture for determining the classification of the image has a very low error rate due to the increase in number of MNNs used for each classification. We will look at ways to improve the speed in training the classifier next.

4.2 ACCELERATING BACK PROPAGATION

The time required to train a neural network utilizing the back propagation method is the most deterring factor for anyone wanting to utilize it with the training of a neural network. Back propagation learning is a slow and time consuming process for most applications. For instance, as the application becomes more complex, there is an increase in the number of neurons, the process of back propagation requires too much time and thus is not scalable in large systems.

As we found earlier the gradient (or steepest) decent is the best method for pattern/face recognition to reduce the error. The squared error for instance n over all the neurons j in the output layer where dj(n) is the desired outcome and aj(n) is the actual outcome is written as:

$$E(n) = (1/2) \sum (dj(n) - aj(n))2$$

To calculate the average squared error over the total number N is written as:

$$Eavg = (1/N) \sum E(n)$$

During the learning process the back propagation wants to minimize Eavg by adjusting the synaptic weight Wji(n) on the edge connecting the output of neuron(i) to input of neuron(j) and thresholds. The weights are adjusted using the formula below during gradient descent

$$Wji(n+1) = Wji(n) - \eta(\partial E(n) / \partial Wji(n))$$

There are two basic issues associated with the learning rate of the back propagation algorithm. The first issue is dealing with the convergence rate that is dependent on λmax \ λmin. If a large value is selected for the learning rate for the weights associated with λmin, then the learning rate will be too large for the weights associated with λmax. The same is true if a small value is selected for the learning rate for the weights associated with λmax and then the learning rate will be too small for the weights associated with λmin [1]. In summary, the smaller the learning rate the slower the rate of convergence which is much slower and the higher the learning rate the higher the rate of convergence but cause the issue with oscillating across the minimum error.

The other basic issue is related to the length of gradient over a wide range which makes solving difficult problems very time consuming [1]. Therefore, the learning rate needs to be assigned a small value in order to find a solution at the cost of a slow convergence rate and a slow system. So from these two issues we can see that the value of the learning rate is critical, which leads us to selecting the Method 2 learning rate update rule defined in [1]. The result presented in Table 4.1 were taken from [1] show that the Method 2 learning rate update rule has a large increase in improvement over the back propagation and then number of epochs are greatly reduced.

Even though the Method 1 has better numbers the Method 2 does not require any comparison of the gradient $\partial E(n) / \partial Wji(n)$, therefore Method 2 was selected as the optimal solution in accelerating the learning process of the back propagation [1].

Table 4.1: Results of Method to Accelerate Back Propagation

Algorithm	**Iterations (average)**	**% Improvement Over (BP)**	**Non-convergence**
On-line BP (Hyperbolic tangent)	32482	-	
Learning rate update rule(Hyperbolic tangentMethod1)	1894	94.2%	0
On-line BP (sigmoid)	34670	-	-
Learning rate update rule (sigmoid,Method2)	4965	85.7%	0

4.3 BAYESIAN CLASSIFIER

Here we are looking at the probabilistic similarity measure based on the Bayesian belief that two images intensity differences are the characteristics of a typical difference in appearance of a specific person. For example some of these differences could be lighting or expressions or similar small changes. The difference in intensity is represented as $\Delta = I1 - I2$, where I1 is the intensity from image 1 and I2 is the intensity of image 2.

One of the types of classes of facial image variations is the interpersonal variations $\Omega(I)$, which is corresponding to facial expression and different lighting of the same person. The other type of class of facial image variation is extra personal variations $\Omega(E)$, which is corresponding to variations between different individuals.

With the above information defined we will be looking at calculating the similarity measure in terms of the probability

$$S(I(1), I(2)) = P(\Delta \in \Omega(I)) = P(\Omega(I) \mid \Delta)$$

Where $P(\Omega(I) \mid \Delta)$ is the a posteriori probability given by Bayes rule, using the estimates of the likelihoods $P(\Delta \mid \Omega(I))$ and $P(\Delta \mid \Omega(E))$ [5,6]. Given these likelihoods we can calculate the similarity score between image 1 and image 2 in terms of the intrapersonal a posteriori probability as given Bys rule using equation (10) below:

$$S(I(1),I(2)) = \frac{P(\Delta|\Omega(I))P(\Omega(I))}{P(\Delta|\Omega(I))P(\Omega(I)) + P(\Delta|\Omega(E))P(\Omega(E))}$$

This simpler problem is then solved using the maximum a posteriori (MAP) rule, where two images are of the same individual if $S(I(1),I(2)) > 1/2$ or $P(\Omega(I)|\Delta) > P(\Omega(E)|\Delta)$ [5,6]. An alternative probabilistic similarity measure can be defined using the intrapersonal likelihood be itself to create a simpler formula

$$S = P\left(\Delta\middle|\Omega(I)\right) = \frac{e^{\frac{-1}{2}}||i_j - i_k||^2}{(2\pi)^{D/2}|\sum_I|^{1/2}}$$

Which leads to maximum likelihood (ML) recognition. The experimental results presented in [4] indicates that this simplified ML measure is almost as effective as the more complicated MAP measure in most cases.

CHAPTER - V

DATA SOURCES

5.1 AT&T THE DATABASE OF FACES

The Database of faces was formally known as the ORL Database of Faces. The database can be located at south amaston state university research database. The database contains 10 different images of each distinct individual and there are 40 individuals. Some of the individuals the pictures were taken at different times, with varying lighting and facial expressions.

Some of the types of facial expression are open or closed eyes, smiling or not smiling and facial details of glasses or no glasses. From the research that I have done I have noticed that there are more issues related to having facial augmentation by having colored glasses or change in facial hair. This is due to the fact that the eyes are not able to be seen and in the other case the features of the face are hidden by facial hair.

All of the images included in the database were taken against a dark homogeneous background, which is not as realistic in a real life setting. Also the subjects are upright and frontal position with some slight side movement, which again is much more of the best case scenarios if you were to ever deal with realistic situations. An example is provided in Fig 5.1.

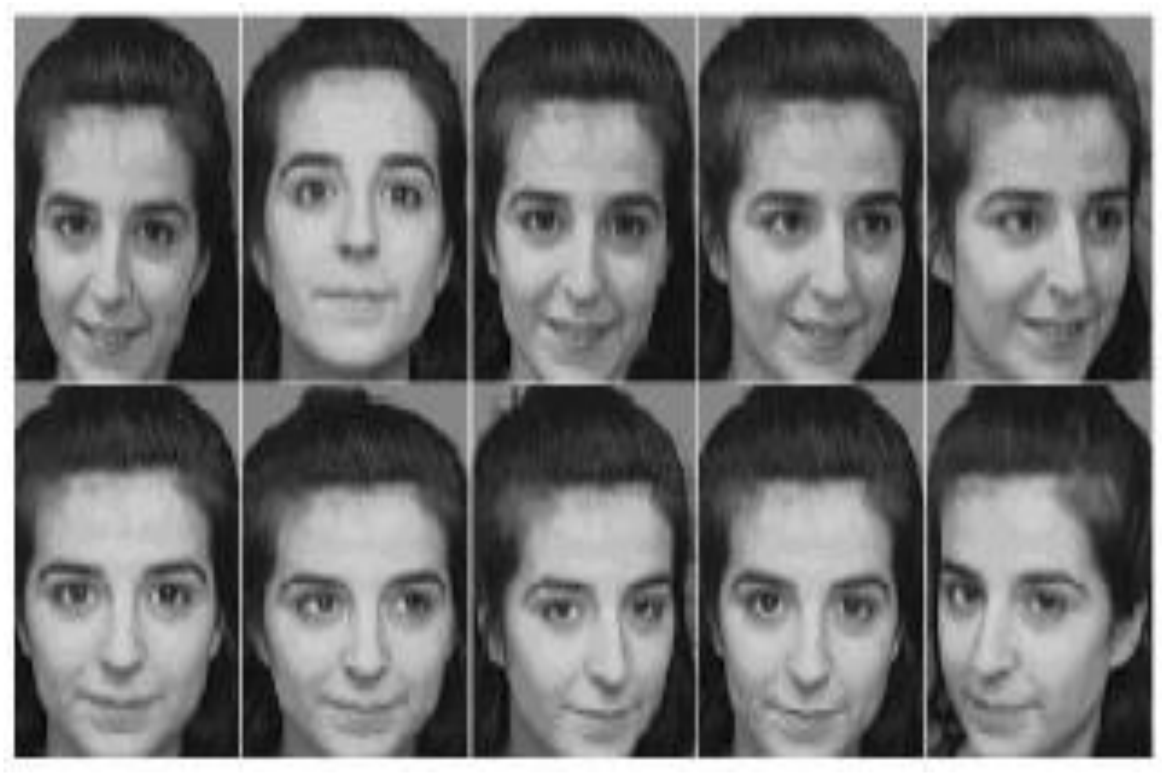


Fig 5.1: Example of the Database of Faces

5.2 FERET Data source

The FERET database contains images of 1196 individuals, with up to 5 different images captured for each individual. An example is provided in Fig 5.1. The images are separated into two sets: gallery images and probes images. Gallery images are images with known labels, while probe images are matched to gallery images for identification. The database is broken into four categories:

FB: Two images were taken of an individual, one after the other. One image is of the individual with a neutral facial expression, while the other is of the individual with a different expression. One of the images is placed into the gallery file while the other is used as a probe. In this category, the gallery contains 1196 images, and the probe set has 1195 images.

Duplicate I: The only restriction of this category is that the gallery and probe images are different. The images could have been taken on the same day or 1 ½ years apart. In this category, the gallery consists of the same 1196 images as the FB gallery while the probe set contains 722 images.

FC: Images in the probe set are taken with a different camera and under different lighting than the images in the gallery set. The gallery contains the same 1196 images as the FB & Duplicate I galleries, while the probe set contains 194 images.

Duplicate II: Images in the probe set were taken at least 1 year after the images in the gallery. The gallery contains 864 images, while the probe set has 234 images.

Computational models of face recognition, in particular, are interesting because they can contribute not only to theoretical insights but also to practical applications. Computers that recognize faces could be applied to a wide variety of problems, including criminal identification, security systems, image and film processing, and human computer interaction. Unfortunately, developing a computational model of face recognition is quite difficult, because faces are complex, multidimensional, and meaningful visual stimuli.

The user should focus his attention toward developing a sort of early, pre attentive Pattern recognition capability that does not depend on having three-dimensional information or detailed geometry. He should develop a computational model of face recognition that is fast, reasonably simple, and accurate.

Automatically learning and later recognizing new faces is practical within this framework. Recognition under widely varying conditions is achieved by training on a limited number of characteristic views. The approach has advantages over other face recognition schemes in its speed and simplicity learning capacity.

Images of faces, represented as high-dimensional pixel arrays, often belong to a manifold of intrinsically low dimension.

Face recognition, and computer vision research in general, has witnessed a growing interest in techniques that capitalize on this observation, and apply algebraic and statistical tools for extraction and analysis of the underlying manifold. Eigenface is a face recognition approach that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals.

The computational approach taken in this system is motivated by both physiology and information theory, as well as by the practical requirements of near-real-time performance and accuracy. This approach treats the face recognition problem as an intrinsically two-dimensional (2-D) recognition problem rather then requiring recovery of three-dimensional geometry, taking advantage of the fact that faces are normally upright and thus may be described by a small set of 2-D characteristic views.

FACE SPACE AND ITS DIMENSIONALITY

Computer analysis of face images deals with a visual signal (light reflected of the surface of a face) that is registered by a digital sensor as an array of pixel values. The pixels may encode color or only intensity. After proper normalization and resizing to a fixed m-by-n size, the pixel array can be represented as a point (i.e. vector) in a m-by-n-dimensional image space by simply writing its pixel values in a fixed (typically raster) order. A critical issue in the analysis of such multi-dimensional data is the dimensionality, the number of coordinate necessary to specify a data point.

IMAGE SPACE Vs FACE SPACE

In order to specify an arbitrary image in the image space, one needs to specify every pixel value. Thus the “nominal" dimensionality of the space, dictated by the pixel representation, is mn – a very high number even for images of modest size However, much of the surface of a face is smooth and has regular texture.

Therefore, per-pixel sampling is in fact unnecessarily dense. The value of a pixel is typically highly correlated with the values of the surrounding pixels.

Moreover, the appearance of faces is highly constrained; for example, any frontal view of a face is roughly symmetrical, has eyes on the sides, nose in the middle, etc. A vast proportion of the points in the image space do no represent physically possible faces. Thus, the natural constraints dictate that the face images will in fact be confined to a subspace, which is referred to as the face space.

CHAPTER- VI

IMPLEMENTATION OF FACE RECOGNITON IN MATLAB

The entire sequence of training and testing is sequential and can be broadly classified as consisting of following two steps:

1. Database Preparation
2. Training
3. Testing

The steps are shown below.

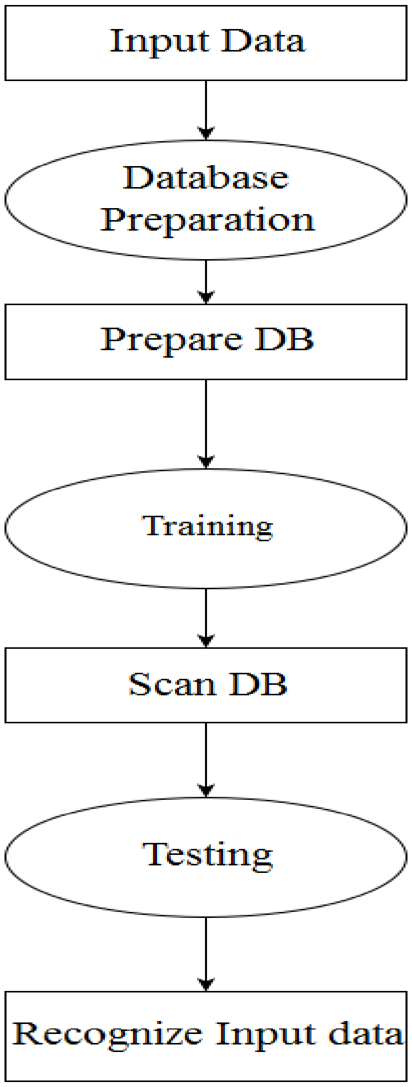


Fig 6.1: Flowchart indicating the sequence of implementation

6.1 DATABASE PREPARATION

The database was obtained with 15 photographs of each person at different viewing angels and different expressions. There are 28 persons in database1.The Database is kept in the train folder which contains subfolders for each person having all his/her photographs[8]. Database was also prepared for testing phase by taking 4-5 photographs of 10 persons in different expressions and viewing angles but in similar conditions (such as lighting, background, distance from camera etc.) using a low resolution camera. And these images were stored in test folder.

6.2 TRAINING

1. Select any one (.bmp) file from train database using open file dialog box.
2. By using that read all the faces of each person in train folder.
3. Normalize all the faces.
4. Find significant Eigenvectors of Reduced Covariance Matrix.
5.Hence calculate the Eigenvectors of Covariance Matrix.
6.Calculate Recognizing Pattern Vectors for each image and average RPV for each person
7. For each person calculate the maximum out of the distances of his entire image RPVs from average RPV of that person.

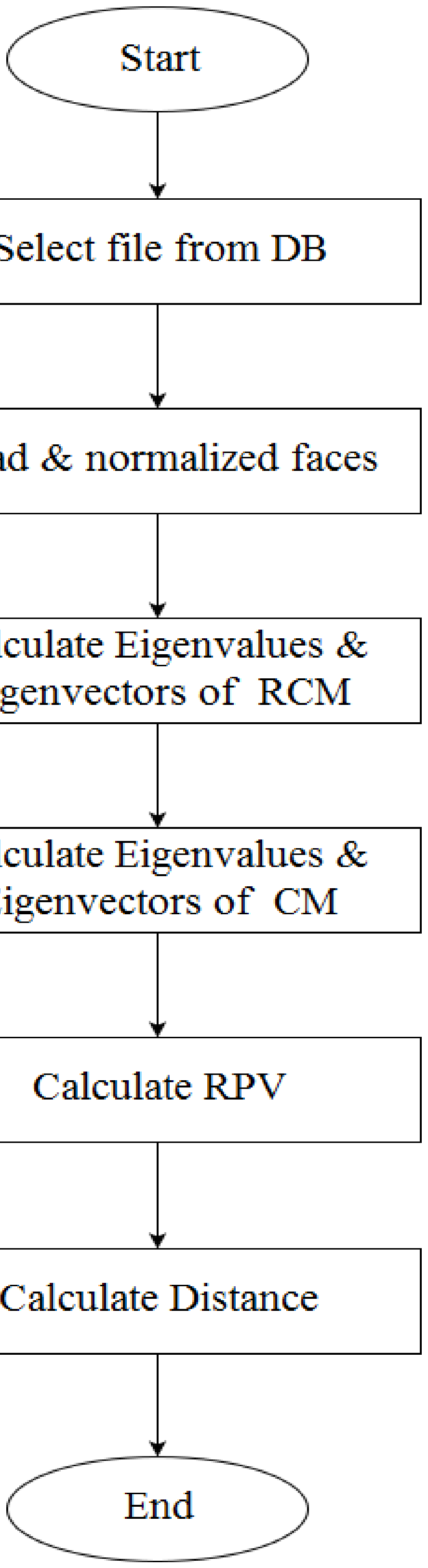

Start
Select file from DB
Read & normalized faces
Calculate Eigenvalues & Eigenvectors of RCM
Calculate Eigenvalues & Eigenvectors of CM
Calculate RPV
Calculate Distance
End

Fig 6.2: Flowchart for training

6.3 TESTING

Testing is carried out by following steps:

1. Select an image which is to be tested using open file dialog box.
2. Image is Read and normalize.
3. Calculate the RPV of image using Eigenvector of Covariance Matrix.
4. Find the distance of this input image RPV from average RPVs of all the persons.
5. Find the person from which the distance is minimum.
6. If this minimum distance is less than the maximum distance of that person calculated during training than the person is identified as this person.

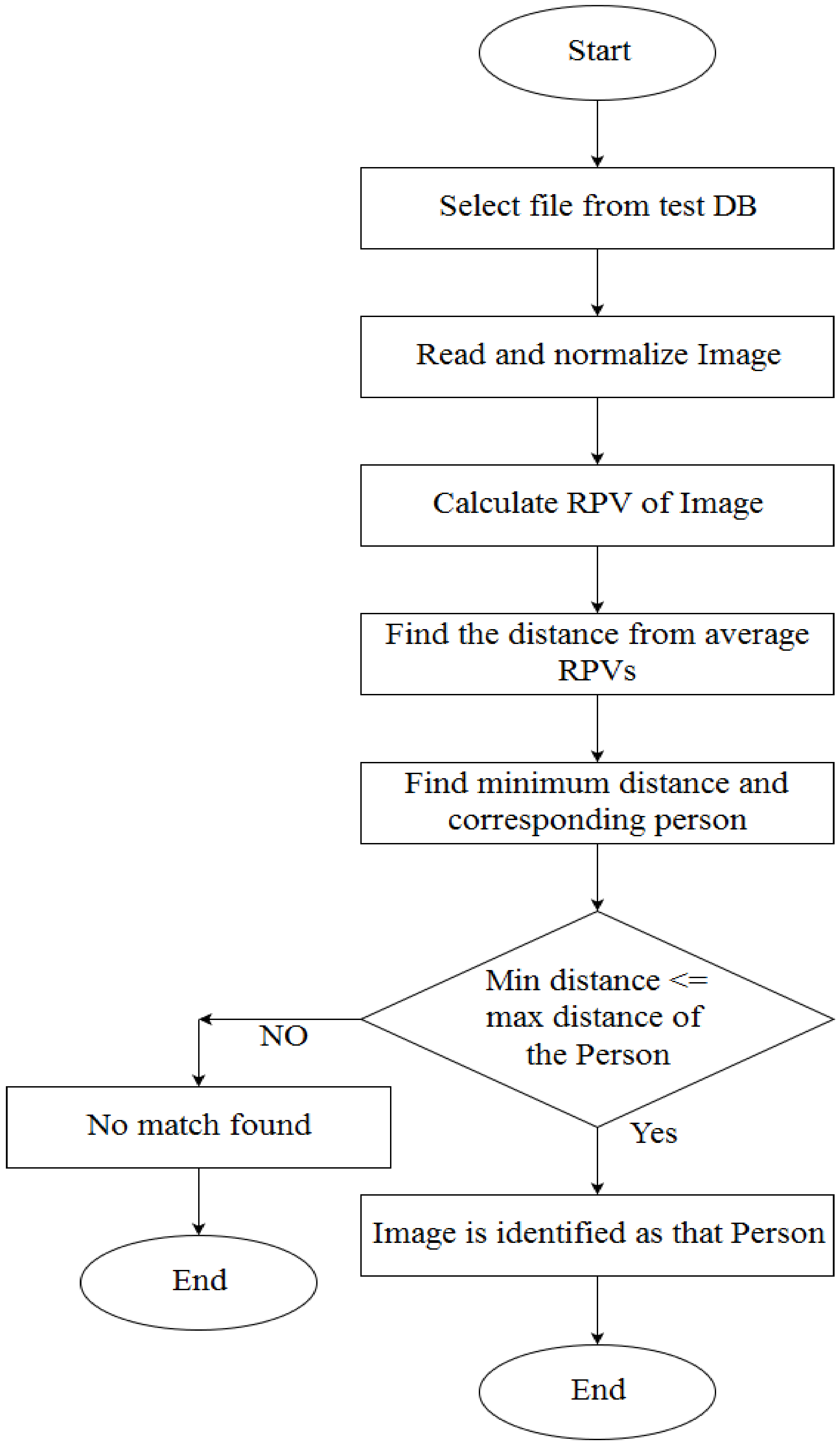

Start
Select file from test DB
Read and normalize Image
Calculate RPV of Image
Find the distance from average RPVs
Find minimum distance and corresponding person
Min distance <= max distance of the Person
NO
No match found
End
Yes
Image is identified as that Person
End

Fig 6.3: Flow chart for testing

6.4 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a dimensionality reduction technique based on extracting the desired number of principal components of the multi-dimensional data. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables. The first principal component is the linear combination of the original dimensions that has the maximum variance; the n-th principal component is the linear combination with the highest variance, subject to being orthogonal to the n -1 first principal components.

An important and largely unsolved problem in dimensionality reduction is the choice of k-the intrinsic dimensionality of the principal manifold. No analytical derivation of this number for a complex natural visual signal is available to date. To simplify this problem, it is common to assume that in the noisy embedding of the signal of interest (in our case, a point sampled from the faces pace) in a high-dimensional space, the signal-to-noise ratio is high. Statistically, that means that the variance of the data along the principal modes of the manifold is high compared to the variance within the complementary space.

This assumption relates to the Eigen spectrum - the set of the Eigen values of the data covariance matrix. The i-th eigen value is equal to the variance along the i-th principal component; thus, a reasonable algorithm for detecting k is to search for the location along the decreasing eigen spectrum where the value of λi, drops significantly Since the basis vectors constructed by PCA had the same dimension as the input face images, they were named “Eigen faces".

PCA is an information theory approach of coding and decoding face images may give insight into the information content of face images, emphasizing the significant local and global "features". Such features may or may not be directly related to face features such as eyes, nose, lips, and hair.

In the language of information theory, we want to extract the relevant information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of models encoded similarly. A simple approach to extracting the information contained in an image of face is to somehow capture the variation in a collection of images, independent of any judgment of features, and use this information to encode and compare individual face images.

These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less of each eigenvector, so that we can display the eigenvector as a sort of ghostly face which we call an eigenface. Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. Each face can also be approximated using only the "best" eigenfaces-those that have the largest eigenvalues and which therefore account for the most variance within the set of face images. The best M eigenfaces span an M-Dimensional subspace- "face space" – of all possible images.

This approach of face recognition involves the following operations:

Initialization operations

1. Acquire an initial set of face images (the training set).
2. Calculate the eigenfaces from the training set, keeping only the M images that correspond to the highest eigenvalues. These M images define the face space. As new faces are experienced; the eigenfaces can be up-dated or recalculated.
3. Calculate the corresponding distribution in M-dimensional weight space for each known individual, by projecting his or her face images onto the "face space".

Recognize new face images

1. Calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.
2. Determine if the image is a face by checking to see if the image is sufficiently close to "face space".
3. If it is a face, classify the weight pattern as either a known person or as unknown.

4. (Optional) Update the eigenfaces and/or weight patterns.
5. (Optional) If the same unknown face is seen several times, calculate its characteristic weight pattern and incorporate into the known faces.

6.5 EIGEN RECOGNITION

To summarize the Eigen faces approach to face recognition involves the following steps:

- Collect a set of characteristic face images of the known individuals. This set should include a number of images of each person, with some variation in expression and in the lighting. (Say four images of ten people, so $M = 40$)

- Calculate the (40 x 40) matrix L, find its Eigen values and eigenvectors, and choose the M' eigenvectors with the highest associated Eigen values. (Let $M' = 10$ in example).

- Combine the normalized set of images to produce the ($M' = 10$) Eigen faces k.

- For each known individual, calculate the class vector Ωk by averaging pattern vector calculated from the original (four) images of the individual. Choose a threshold $\theta\varepsilon$ that defines the maximum allowable distance from any face class, and a threshold $\theta\varepsilon$ that defines the maximum allowable distance from face space.

- For each new image to be identified, calculate its pattern vector Ω, the distance εI to each known class, and the distance ε to face space. If the minimum distance $\varepsilon k < \theta\varepsilon$ and the distance $\varepsilon < \theta\varepsilon$, classify the input face as the individual associated with class vector Ωk. If the minimum distance $\varepsilon k = \theta\varepsilon$ but distance $\varepsilon < \theta\varepsilon$, then the image may be classify as "unknown", and optionally used to begin a new face class.

- If new image is classified as a known individual; this image may be added to the original set of familiar face images, and the Eigen faces may be recalculated. This gives the opportunity to modify the face space as the system encounters more instances of the known faces.

CHAPTER -VII

APPLICATIONS

1. Access Control

- ATM
- Airport

2. Entertainment

- Video Game
- Virtual Reality
- Training Programs
- Human-Computer-Interaction
- Human-Robotics
- Family Photo Album

3. Smart Cards

- Drivers' Licenses
- Passports
- Voter Registrations
- Welfare Fraud
- Voter Registration

4. Information Security

- TV Parental control
- Desktop Logon
- Personal Device Logon
- Database Security
- Medical Access
- Internet Access

5. Law Enforcement & Surveillance

- Advanced Video surveillance
- Drug Trafficking
- Portal Control

6. Multimedia management

- Multimedia management is used in the face based database search.

7. Commercial Applications

- Motion Capture for movie special effects.
- Face Recognition biometric systems.
- Home Robotics.

CHAPTER VIII

EXPERIMENTAL RESULTS

A number of experiments were performed to evaluate the system. We first show an analysis of which features the neural network is using to detect faces, then present the error rates of the system over two large test sets.

8.1 SENSITIVITY ANALYSIS

In order to determine which part of its input image the network uses to decide whether the input is a face, we performed a sensitivity analysis using the method of [2]. We collected a positive test set based on the training database of face images, but with different randomized scales, translations, and rotations than were used for training. The negative test set was built from a set of negative examples collected during the training of other networks. Each of the 20x20 pixel input images was divided into 100 2x2 pixel subimages. For each subimage in turn, we went through the test set, replacing that subimage with random noise, and tested the neural network.

The resulting root mean square error of the network on the test set is an indication of how important that portion of the image is for the detection task. Plots of the error rates for two networks we trained a. Network 1 uses two sets of the hidden units illustrated in Fig 8.1, while Network 2 uses three sets. The networks rely most heavily on the eyes, then on the nose, and then on the mouth .

Anecdotally, we have seen this behavior on several real test images. In cases in which only one eye is visible, detection of a face is possible, though less reliable, than when the entire face is visible. The system is less sensitive to the occlusion of the nose or mouth.

8.2 TESTING

The system was tested on two large sets of images, which are distinct from the training sets. Test Set 1 consists of a total of 130 images collected at CMU, including images from the World Wide Web, scanned from photographs and newspaper pictures, and digitized from broadcast television3. It also includes 23 images used in to measure the accuracy of their system. The images contain a total of 507 frontal faces, and require the networks to examine 83,099,211 20x20 pixel windows.

The images have a wide variety of complex backgrounds, and are useful in measuring the false alarm rate of the system. Test Set 2 is a subset of the FERET database [9, 10]. Each image contains one face, and has (in most cases) a uniform background and good lighting. There are a wide variety of faces in the database, which are taken at a variety of angles. Thus these images are more useful for checking the angular sensitivity of the detector, and less useful for measuring the false alarm rate.

The outputs from our face detection networks are not binary. The neural network produce real values between 1 and -1, indicating whether or not the input contains a face. A threshold value of zero is used during *training* to select the negative examples (if the network outputs a value of greater than zero for any input from a scenery image, it is considered a mistake). Although this value is intuitively reasonable, by changing this value during *testing*, we can vary how conservative the system is.

To examine the effect of this threshold value during testing, we measured the detection and false positive rates as the threshold was varied from 1 to -1. At a threshold of 1, the false detection rate is zero, but no faces are detected. As the threshold is decreased, the number of correct detections will increase, but so will the number of false detections. This tradeoff is presented , which shows the detection rate plotted against the number of false positives as the threshold is varied, for the two networks presented in the previous section.

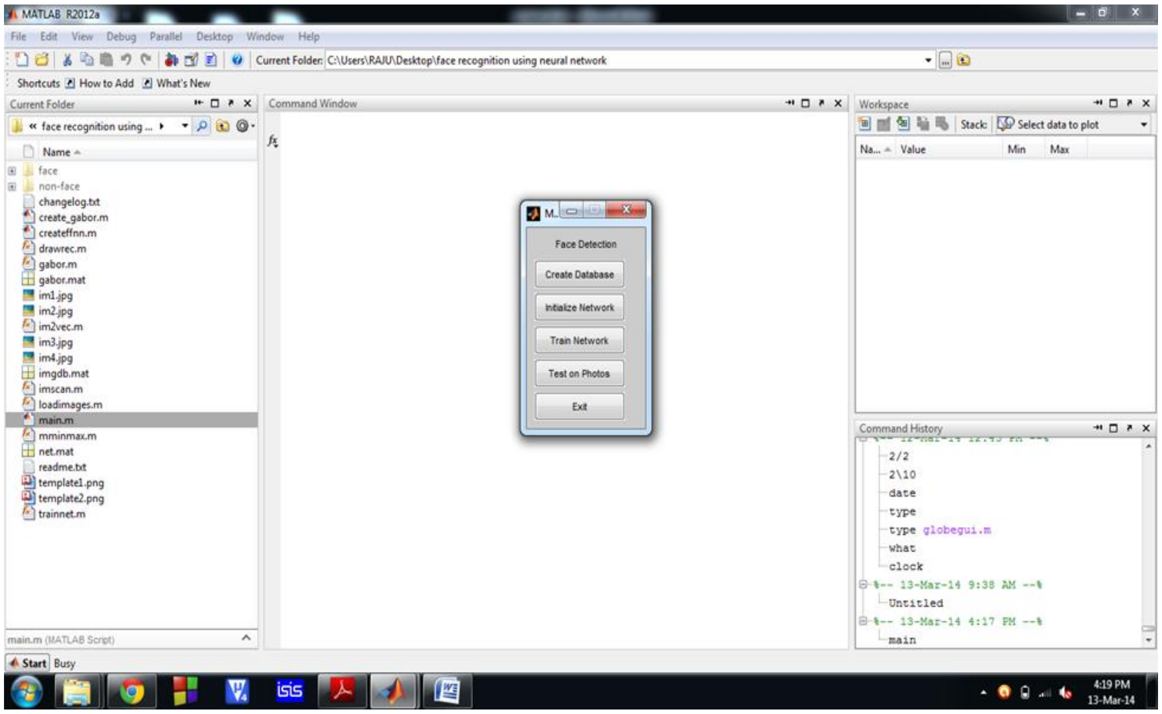
Since the zero threshold locations are close to the “knees” of the curves, as can be seen from the figure, we used a zero threshold value throughout testing.presented in a different order during training. The results for ANDing and ORing networks were based on Networks 1 and 2, while voting and network arbitration were based on Networks 1, 2, and 3. The neural network arbitrators were trained using the images from which the face examples

As discussed earlier, the “thresholding” heuristic for merging detections requires two parameters, which specify the size of the neighborhood used in searching for nearby detections, and the threshold on the number of detections that must be found in that neighborhood. Centroids must be in order to be counted as identical. Systems 1 ORing, and voting arbitration methods have a parameter specifying how close two detections .

1. A false detection made by only one network is suppressed, leading to a lower false positive rate.
2. On the other hand, when ORing is used, faces detected correctly by only one network will be.

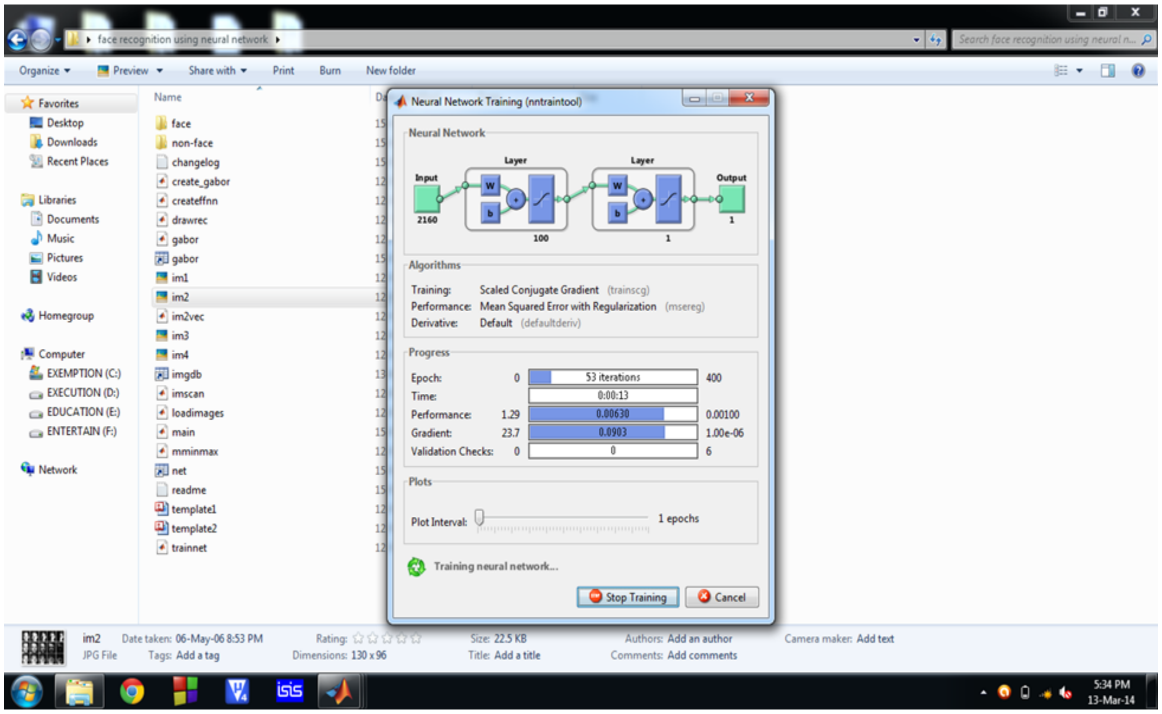
8.3 SCREEN SHOTS

Step:1 Create the data base



```
2/2
2\10
date
type
type globegui.m
what
clock
%-- 13-Mar-14 9:38 AM --%
Untitled
%-- 13-Mar-14 4:17 PM --%
main
```

Step:2 Initialize Networks



Algorithms

Training: Scaled Conjugate Gradient (trainscg)
Performance: Mean Squared Error with Regularization (msereg)
Derivative: Default (defaultderiv)

Progress

Epoch:	0	53 iterations	400
Time:		0:00:13	
Performance:	1.29	0.00630	0.00100
Gradient:	23.7	0.0903	1.00e-06
Validation Checks:	0	0	6

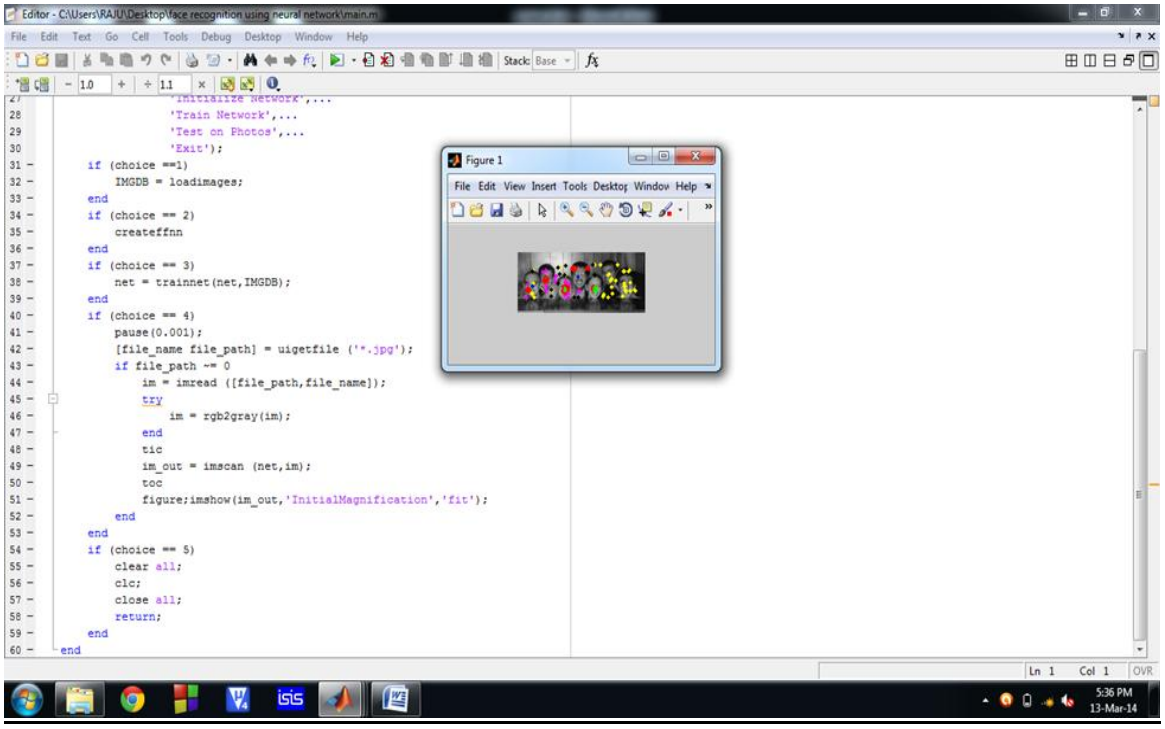
Plots

Plot Interval: 1 epochs

Training neural network...

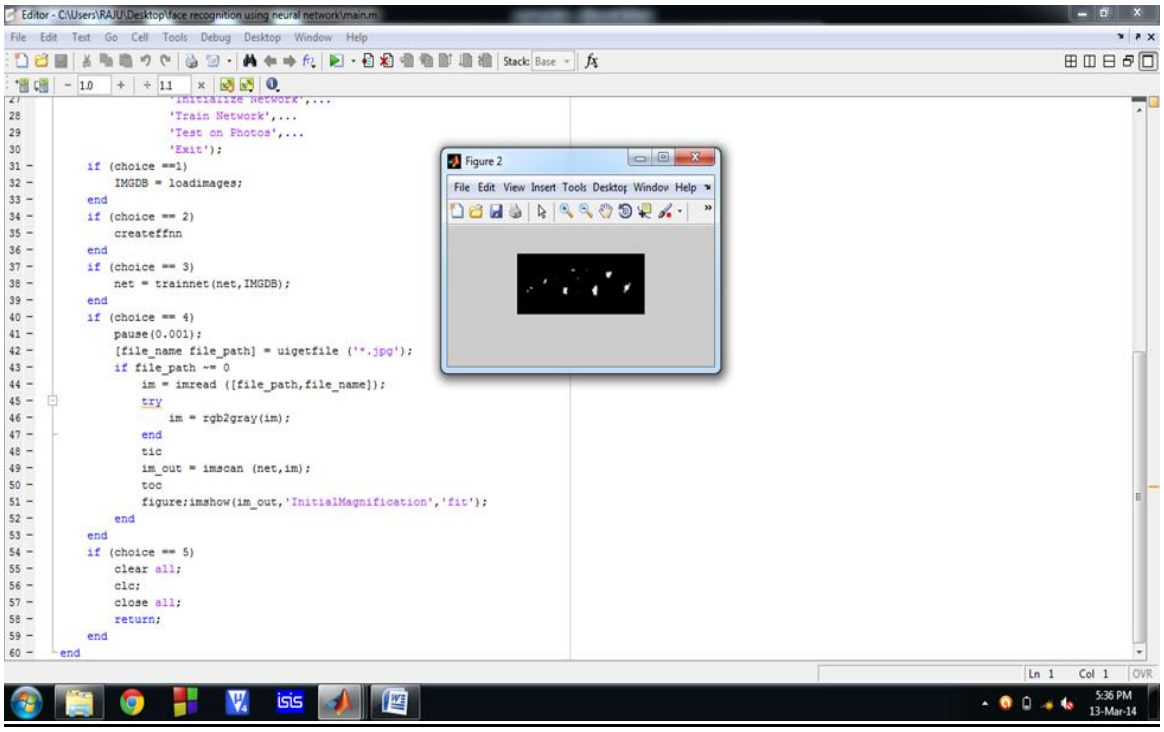
im2 Date taken: 06-May-06 8:53 PM Rating: Size: 22.5 KB Authors: Add an author Camera maker: Add text
JPG File Tags: Add a tag Dimensions: 130 x 96 Title: Add a title Comments: Add comments

Step:3 Test on photo


Editor - C:\Users\RAJU\Desktop\face recognition using neural network\main.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
Figure 1
File Edit View Insert Tools Desktop Window Help
Ln 1 Col 1 OVR
5:36 PM
13-Mar-14

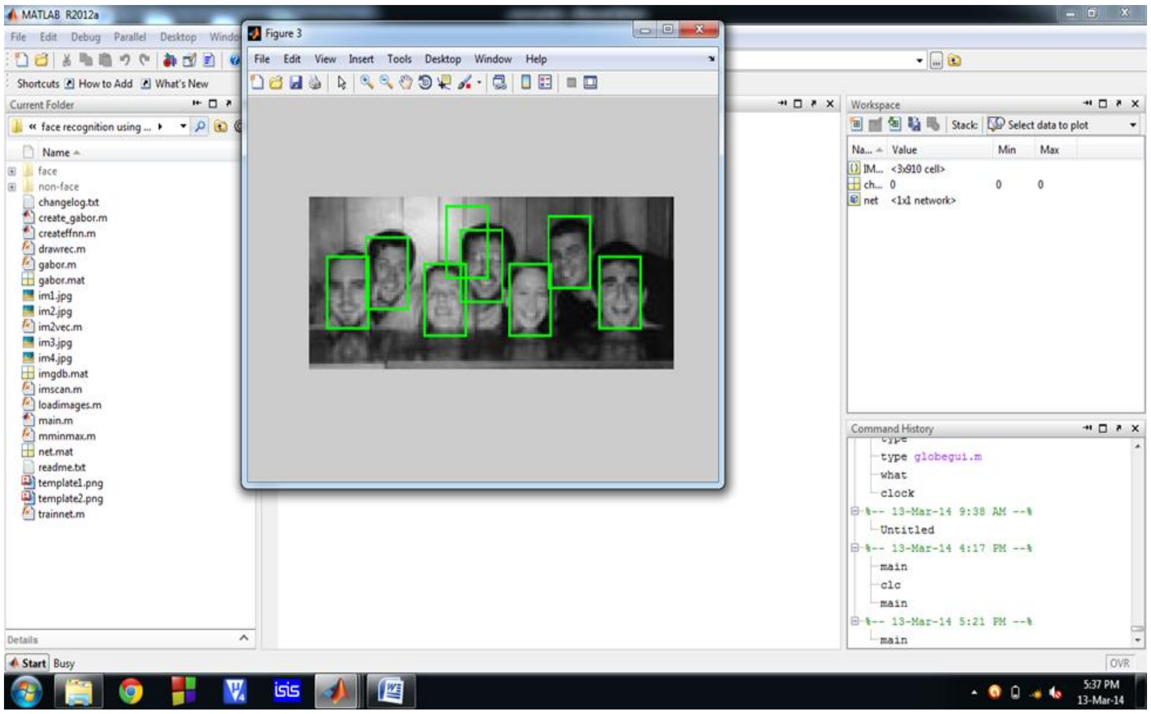
```
                'Initialize Network',...
                'Train Network',...
                'Test on Photos',...
                'Exit');
    if (choice ==1)
        IMGDB = loadimages;
    end
    if (choice == 2)
        createffnn
    end
    if (choice == 3)
        net = trainnet(net,IMGDB);
    end
    if (choice == 4)
        pause(0.001);
        [file_name file_path] = uigetfile ('*.jpg');
        if file_path ~= 0
            im = imread ([file_path,file_name]);
            try
                im = rgb2gray(im);
            end
            tic
            im_out = imscan (net,im);
            toc
            figure;imshow(im_out,'InitialMagnification','fit');
        end
    end
    if (choice == 5)
        clear all;
        clc;
        close all;
        return;
    end
end
```

Step:4 Detected points in the image


Editor - C:\Users\RAJU\Desktop\face recognition using neural network\main.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
Figure 2
File Edit View Insert Tools Desktop Window Help
Ln 1 Col 1 OVR
5:36 PM
13-Mar-14

```
                'Initialize Network',...
                'Train Network',...
                'Test on Photos',...
                'Exit');
    if (choice ==1)
        IMGDB = loadimages;
    end
    if (choice == 2)
        createffnn
    end
    if (choice == 3)
        net = trainnet(net,IMGDB);
    end
    if (choice == 4)
        pause(0.001);
        [file_name file_path] = uigetfile ('*.jpg');
        if file_path ~= 0
            im = imread ([file_path,file_name]);
            try
                im = rgb2gray(im);
            end
            tic
            im_out = imscan (net,im);
            toc
            figure;imshow(im_out,'InitialMagnification','fit');
        end
    end
    if (choice == 5)
        clear all;
        clc;
        close all;
        return;
    end
end
```

THE OUTPUT IS OBSERVED AS FOLLOWS



Na...	Value	Min	Max
IM...	<3x910 cell>		
ch...	0	0	0
net	<1x1 network>		

```
type globegui.m
what
clock
%-- 13-Mar-14 9:38 AM --%
Untitled
%-- 13-Mar-14 4:17 PM --%
main
clc
main
%-- 13-Mar-14 5:21 PM --%
main
```

CHAPTER - IX

CONCLUSION AND FUTURE SCOPE

Our algorithm can recognize between 77.9% and 90.3% of faces in a set of 130 test images, with an acceptable number of false detections. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or thresholds used. The system has been tested on a wide variety of images, with many faces and unconstrained backgrounds. A fast version of the system can process a 320x240 pixel image in 2 to 4 seconds on a 200 MHz R4400 SGI Indigo 2.

There are a number of directions for future work. The main limitation of the current system is that it only detects upright faces looking at the camera. Separate versions of the system could be trained for each head orientation, and the results could be combined using arbitration methods similar to those presented here. Preliminary work in this area indicates that detecting profiles views of faces is more difficult than detecting frontal views, because they have fewer stable features and because the input window will contain more background pixels. We have also applied the same algorithm for the detection of car tires and human eyes, although more work is needed.

Even within the domain of detecting frontal views of faces, more work remains. When an image sequence is available, temporal coherence can focus attention on particular portions of the images. As a face moves about, its location in one frame is a strong predictor of its location in next frame. Standard tracking methods, as well as expectation-based methods , can be applied to focus the detector’s attention. Other methods of improving system performance include obtaining more positive examples for training, or applying more sophisticated image preprocessing and normalization techniques.

One application of this work is in the area of media technology. Every year, improved technology provides cheaper and more efficient ways of storing and retrieving visual information.

However, automatic high-level classification of the information content is very limited, this is a bottleneck that prevents media technology from reaching its full potential. Systems utilizing the detector described above allow a user to make requests of the form “Show me the people who appear in this video” or “Which images on the World Wide Web contain faces?” and to have their queries answered automatically.

REFERENCES

[1] Shumeet Baluja. *Expectation-Based Selective Attention*. Available as CS Technical Report CMU-CS-96-182, October 1996.

[2] Gilles Burel and Dominique Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 15:963–967, October 1994.

[3] Antonio J. Colmenarez and Thomas S. Huang. Face detection with information-based maximum discrimination. In *Computer Vision and Pattern Recognition*, pages 782–787, 1997.

[4] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705– 719, 1993.

[5] Charles Frankel,Michael J. Swain, and Vassilis Athitsos. WebSeer: An image search enginefor the world wide web. Technical Report TR-96-14, University of Chicago, August 1996.

[6] Venu Govindaraju. Locating human faces in photographs. *International Journal of Computer Vision*, 19(2):129–146, 1996.

[7] 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.
Rowley, Baluja, and Kanade: Neural Network-Based Face Detection (PAMI, January 1998).

[8] Alex Pentland, BabackMoghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[9] P. Jonathon Phillips, Hyeonjoon Moon, Patrick Rauss, and Syed A. Rizvi. The FERET evaluation methodology for face-recognition algorithms. In *Computer Vision and Pattern Recognition*, pages 137–143, 1997.

[10] P. Jonathon Phillips, Patrick J. Rauss, and Sandor Z. Der. FERET (face recognition technology) recognition algorithm development and test results. Technical Report ARL-TR-995, Army Research Laboratory, October 1996.

[11] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *Readings in Speech Recognition*, pages 393–404, 1989.

APPENDIX

MATLAB SOURCE CODE

%% creat gabor.m

```
 close all;

clear all;

clc;


G = cell(5,8);

for s = 1:5

    for j = 1:8

        G{s,j}=zeros(32,32);

    end

end

for s = 1:5

    for j = 1:8

        G{s,9-j} = gabor([32 32],(s-1),j-1,pi,sqrt(2),pi);

    end

end


figure;
```

```
for s = 1:5

   for j = 1:8

       subplot(5,8,(s-1)*8+j);

       imshow(real(G{s,j}),[]);

   end

end


for s = 1:5

   for j = 1:8

       G{s,j}=fft2(G{s,j});

   end

end

save gabor G
```

%%trainnet.m

```
function NET = trainnet(net,IMGDB)


%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

net.trainFcn = 'trainscg';

net.trainParam.lr = 0.4;
```

```
net.trainParam.epochs = 400;

net.trainParam.show = 10;

net.trainParam.goal = 1e-3;

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

T{1,1} = cell2mat(IMGDB(2,:));

P{1,1} = cell2mat(IMGDB(3,:));

net = train(net,P,T);

save net net

NET = net;
```

%%main.m

```
clear all;

close all;

clc;

if ~exist('gabor.mat','file')

   fprintf ('Creating Gabor Filters ...');

   create_gabor;

end

if exist('net.mat','file')

   load net;

else

   createffnn
```

```
end

if exist('imgdb.mat','file')

   load imgdb;

else

   IMGDB = loadimages;

end

while (1==1)

   choice=menu('Face Detection',...

            'Create Database',...

            'Initialize Network',...

            'Train Network',...

            'Test on Photos',...

            'Exit');

   if (choice ==1)

      IMGDB = loadimages;

   end

   if (choice == 2)

      createffnn

   end

   if (choice == 3)

      net = trainnet(net,IMGDB);

   end


 if (choice == 4)
```

```
    pause(0.001);

    [file_name file_path] = uigetfile ('*.jpg');

    if file_path ~= 0

        im = imread ([file_path,file_name]);

        try

            im = rgb2gray(im);

        end

        tic

        im_out = imscan (net,im);

        toc

        figure;imshow(im_out,'InitialMagnification','fit');

    end

end

if (choice == 5)

    clear all;

    clc;

    close all;

    return;

end
```