PROJECT REPORT

ON

Face Authentication using Support Vector Machines and Eigenfaces

SUBMITTED BY

ARUN KUMAR MURALIDHARAN	B3313036
MODI ASHUTOSH DILIPRAO	B3313035
SHUBHENDU TRIVEDI	B3313064

DEPARTMENT OF ELECTRONICS AND TELECOMUNICATION

P.E.S.'S MODERN COLLEGE OF ENGINEERING

PUNE – 411005

UNIVERSITY OF PUNE

2008-2009

CERTIFICATE

This is to certify that

ARUN KUMAR MURALIDHARAN	B3313036
MODI ASHUTOSH DILIPRAO	B3313035
SHUBHENDU TRIVEDI	B3313064

Of B.E. (E&TC) have successfully completed successfully the project

"Face Authentication using Support Vector Machines and Eigenfaces"

during the first and second semester of the academic year 2008-2009.

The report is submitted as a partial fulfillment of the requirement of the degree in E&TC Engineering as prescribed University of Pune.

HOD
(E & TC)

Project Guide
(Dr. Mrs. K. R. Joshi)

Acknowledgements

The completion of this project work gives much satisfaction and fulfillment. The project could only see the light of the day with the gracious help of some very important people.

We are extremely grateful to our principal and project guide **Prof Dr (Mrs) K.R Joshi** for the enormous encouragement and guidance and also making laboratories and facilities available and giving us new ideas and making us available external help whenever needed.

We would also like to thank our Head of Department **Prof V.N Patil** and all other staff members for their kind support and guidance.

Also, we would like to thank **Prof (Mrs) R.S Kamathe** and **Mr Sumedh Kulkarni** for giving expert guidance and sharing hands on experience in developing such a project. The project would have been impossible had it not been for the duo we mentioned above. We are extremely grateful to them for giving valuable suggestions and also giving their valuable time.

Shubhendu Trivedi
Ashutosh Modi
Arun Muralidharan

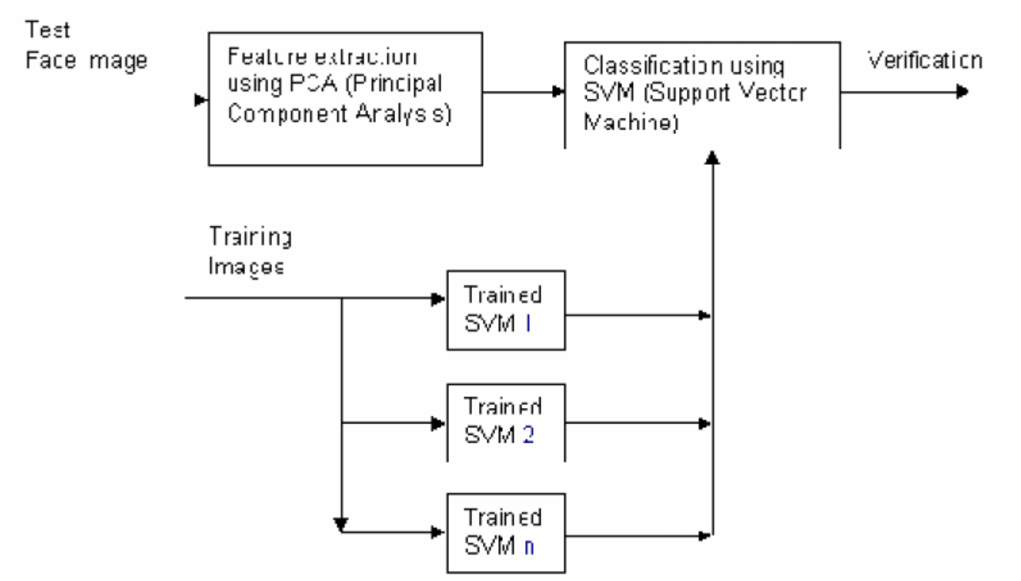
Index

Project Abstract

Face recognition is a high level visual problem that involves discriminating input images into separate classes. Face Authentication is a task that can be considered a subset of authentication.

The task of feature selection is an area of active research; features for face recognition can be both local and global. Eigenfaces that are an information theory based global features were implemented and the weights for each face obtained. The eigenspace thus obtained was pruned to reduce the computational effort also. A test image was then projected onto this eigenspace to obtain a set of weights that were then compared using distance classifiers such as the Mahalanobis distance and the Euclidean distance. The Mahalanobis distance gave much better results as compared to the euclidean distance. The database used was the MIT-CBCL face image database.

For the classification task, Support Vector Machines were also used. Support Vector Machines are a popular category of classifiers that have been derived from Statistical Learning theory by Vladimir Vapnik and his co-workers. The Support Vectors Machines were tuned using a random search algorithm, and cross validated using hold-out cross validation. The SVMs were trained using Sequential Minimal Optimization. Also there was a slight change in the feature representation of features as compared to the distance classifiers. Instead of using a vector of weights and thereby having a k-time two class problem. The task was reduced to a two class problem by mapping the variations of faces into classes only. One being of the variation in the face images of the same person, and the other being that between all the other people other than that person. This works well for face authentication and authentication is basically a two class problem.


Test
Face Image
Feature extraction using PCA (Principal Component Analysis)
Classification using SVM (Support Vector Machine)
Verification
Training Images
Trained SVM 1
Trained SVM 2
Trained SVM n

The accuracy obtained was about 90 percent on the Mit-CBCL database.

An Online version of the project was also implemented wherein images were captured with the camera. To remove reduction in accuracy due to the variation in illumination changes were made and certain eigenfaces responsible were removed.

1. INTRODUCTION

The face is the primary focus of attention in our daily lives during social interactions. It is so because it is a major conveyor of emotion and identity. The human ability to recognize faces is remarkable. The human brain can remember and recognize thousands of faces in its lifetime even after years of separation and in spite of changes in hairstyle, facial hair, aging etc.

Computer based models of face recognition are important for two reasons:
They offer insights into how the brain might be processing visual information to recognize faces and also that given such models are made, they can be put to sound practical use, that is, computers that can recognize faces can be put to a wide array of personal and security uses.

1.1 Biometric Overview

Face authentication is a part of the larger field of biometrics. The word Biometrics comes from the combination of the two Greek words – Bios and Metron, Bios means "life" and "metron" means to measure. We can therefore understand biometrics to be a group of algorithms that measures and certain physiological or behavioral characteristics and then recognizes and authenticates a person based on them.
Common examples of biometrics are fingerprinting, iris scanning, speaker recognition and face authentication. Some rare example are gait recognition. Early quantitative measurements of humans for identification dates back to the late 19th century, including fingerprints and facial measurements. With the development of the integrated circuits in 1960s, automatic fingerprint and speech recognition system were created to protect high security areas.
Face recognition was first implemented the 1980s followed by iris scanning in the 1990s.

Biometrics is classified into two groups: passive and active.

Passive biometrics identifies people without their knowledge or cooperation. Examples of these methods include face recognition, odor recognition, and gait recognition.

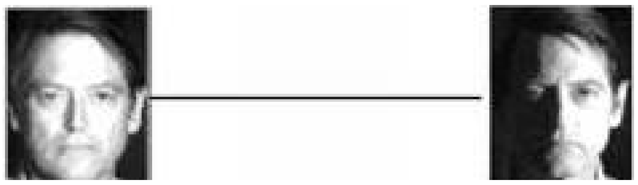
Active biometrics, conversely, require the cooperation of the subject. Examples of these methods include fingerprint authentication, hand graph authentication and iris authentication.

1.2 Face Recognition versus Face Authentication

The two are often confused. Face recognition is most of the times a passive biometric, whereas face authentication an active biometric.

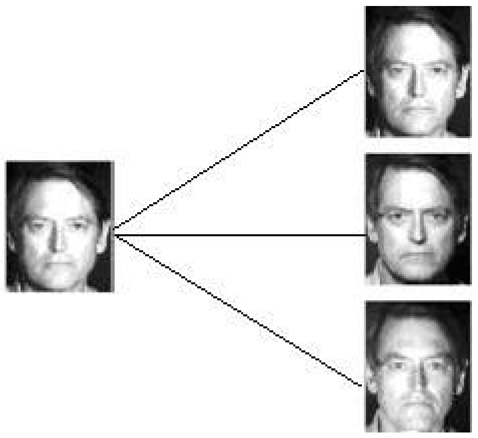
Face Authentication can be considered a subset of face recognition. Though due to the small difference there are a few non-concurrent parts in both the systems.

Face Authentication (also called verification) involves a *one to one* check that compares an input image (also called a query image, probe image or simply probe) with only the image (or class) that the user claims to be. In simple words, if you stand in front of a face authentication system and claim to be a certain user, the system will only check if you are that user or not.



(Fig 1.1)

Face Recognition (or Identification) is another thing, though of course related. It involves a *one to many* comparisons of the input image (or probe or query image) with a template library. In simple words, in a face recognition system the input image will be compared with all the classes and then a decision will be made so as to identify to who the input image belongs to. Or if it does not belong to the database at all.



(Fig 1.2)

1.3 Face Recognition in Multi-Modal Systems and Research Directions

Multi-Modal systems are being pushed increasingly for security. Multi-Modal systems involve the use of more than one biometric in conjunction. This reduces the chances of fake authentication drastically. Some of the latest developed multi-modal systems have speaker and face recognition, gait and face recognition, signature and face recognition and so on. A combination of many modes increases the accuracy of the system.

The face-voice authentication system is one of the most robust authentication systems currently being researched. It combines the single-mode face verification and single-mode voice verification and has strong robustness against impostor and replay attacks. The motivation for Face-Voice Authentication is to enhance discrimination through additional parameters and to decrease vulnerability against impostors. The possible applications for Face-Voice Authentication are physical access to secure areas, telephone call access, payments and banking, internet access, payments and banking, video conferences and link-ups amongst many others.

Face recognition and authentication is now a popular research problem in the computer vision and the applied machine learning domain. The many areas under focus for research in this domain are:

1. Recognition from outdoor facial images.
2. Recognition from non-frontal facial images.
3. Recognition at low false accept/alarm rates.
4. Understanding why males are easier to recognize than females.

5. Greater understanding of the effects of demographic factors on performance.
6. Development of better statistical methods for understanding performance.
7. Develop improved models for predicting identification performance on very large galleries.
8. Effect of algorithm and system training on covariate performance.
9. Integration of morphable models into face recognition performance.
10. Understanding the variation in accuracy due to change in race of probe.

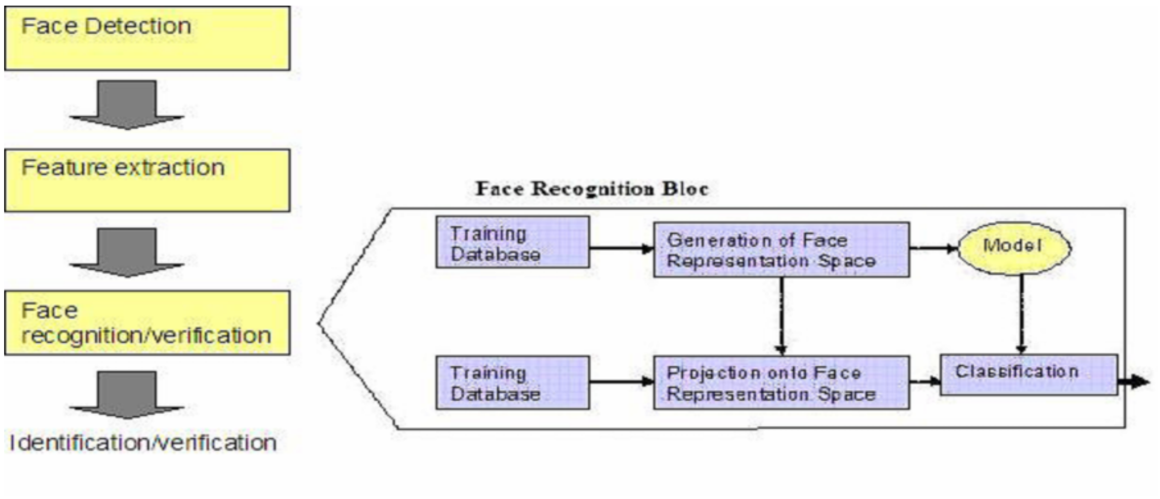
1.4 Problem Overview

A face authentication task involves a number of challenging intermediate steps:

1. Face Detection: Face detection involves detecting faces in an un-controlled environment. It is most suited for detecting faces in video sequences and cases where images have a lot of unwanted objects in the background. There are a number of methods that are used for face detection such Hough transforms.

2. Feature Extraction: This step involves taking the face detected and looking for characteristics that can define the face uniquely. This is currently the most researched area in facial recognition and authentication. Features can local or global, intuitive or un-intuitive etc or combinations of all of these. A number of methods used for face representation have been reviewed in the literature survey.

3. Classification Task: This task can be carried out using distance classifiers, Artificial Neural Networks, Bayesian methods and Support Vector Machines. This step is probably the most elaborate in case some learning methods are used as the learning machine would require extensive training and testing.


Face Detection
Feature extraction
Face recognition/verification
Identification/verification
Face Recognition Bloc
Training Database
Generation of Face Representation Space
Model
Training Database
Projection onto Face Representation Space
Classification

(Fig 1.3)

A face authentication task basically involves detecting faces, representing them conveniently and sufficiently and then classifying them to a specific class.

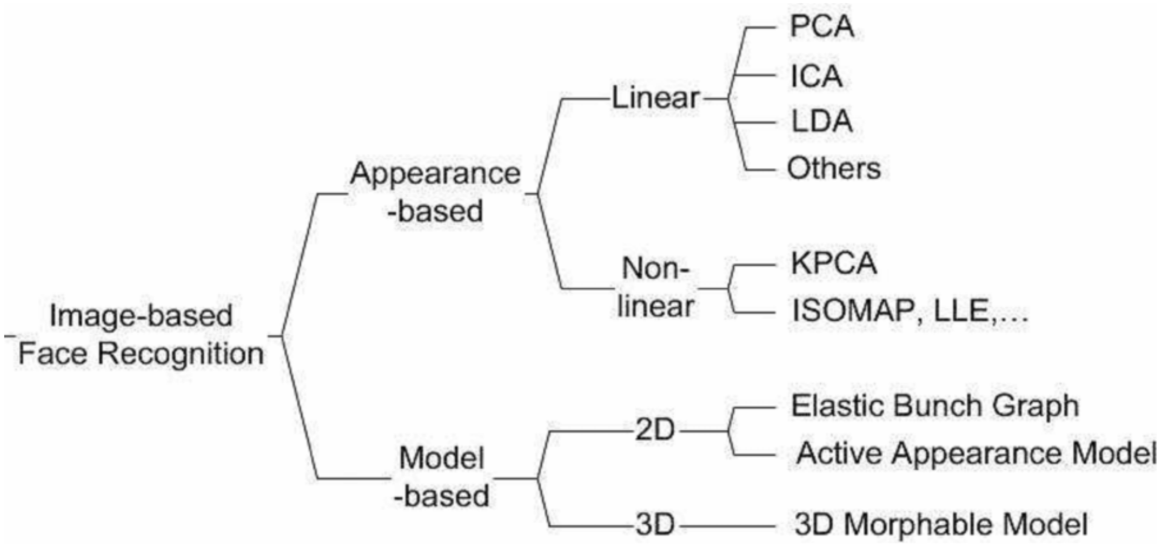
1.5 Work Done

We have implemented an online and offline face authentication system using MATLAB. The system uses Eigenfaces for face representation/feature extraction and Support Vector Machines for the purpose of classification. Support Vector Machines are a popular classification tool derived from statistical learning theory by Vapnik and his co-workers. The Support Vectors Machines were tuned using a random search algorithm, and cross validated using hold-out cross validation. The SVMs were trained using Sequential Minimal Optimization. Also there was a slight change in the feature representation of features as compared to the distance classifiers. Instead of using a vector of weights and thereby having a k-time two class problem. The task was reduced to a two class problem by mapping the variations of faces into classes only. One being of the variation in the face images of the same person, and the other being that between all the other people other than that person. This works well for face authentication and authentication is basically a two class problem.

The accuracy attained was around 85 % which is comparable and even better than systems employing similar techniques.

2. LITERATURE SURVEY

A number of methods for face representation (feature extraction) and classification have been reviewed. We have reviewed methods that take into account global features, local features and sometimes a combination of both. Also, there are two predominant approaches to the face recognition problem: geometric (feature based) and photometric (view based). We have mostly reviewed view based approaches because of their simplicity.



The literature survey was mostly on the methods of face representation and then the subsequent classification task.

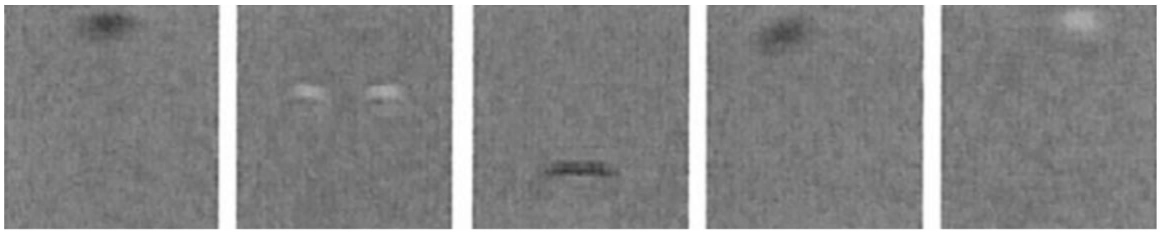
2.1 Face Representation Methods / Models

2.1.1 Eigenfaces [1] (PCA)

This method has been used in the course of this project and has been described in detail in the next section dealing with the algorithms.

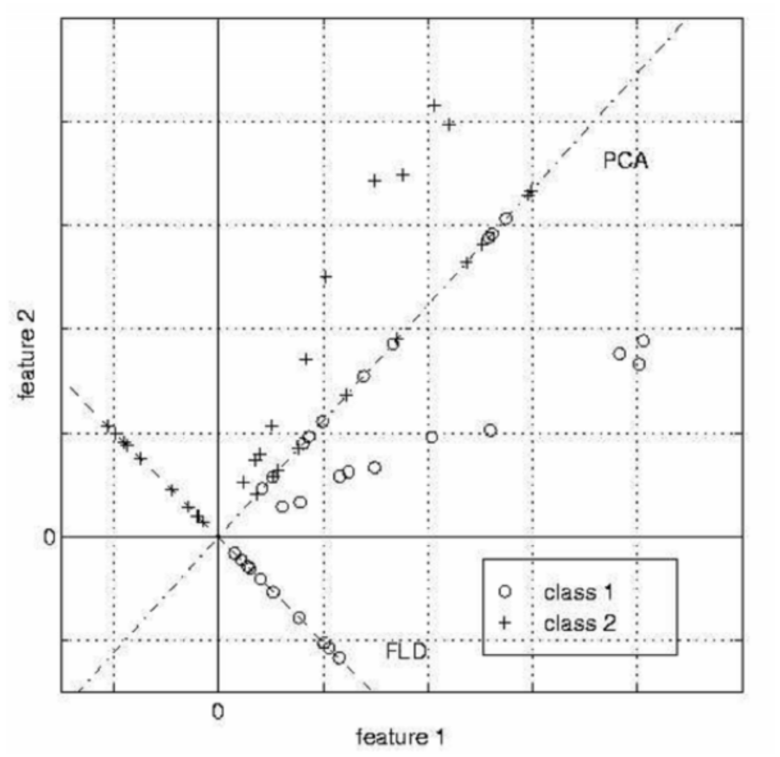
2.1.2 Independent Component Analysis

Independent component analysis is related to PCA and is an unsupervised method for clustering. It is actually a generalization of the PCA. It minimizes the second order and the higher order dependencies in the input data and then attempts to find the basis along which the data is statistically independent. There are two frameworks to implement ICA. The first architecture found spatially local basis images for the faces. The second architecture produced a factorial face code. Both ICA representations were superior to representations based on PCA for recognizing faces across days and changes in expression. A classifier that combined the two ICA representations gave the best performance. Accuracy attained using ICA[15] is around 75-100 percent depending upon the level of difficulty of the database or lighting conditions and the environment. It is also being found that the ICA is biology closer as it embodies a Hebbian.


(Fig 2.1: Independent Components)

2.1.3 Linear Discriminant Analysis

Both PCA and ICA construct the face space without using the face class (category) information. The whole face training data is taken as a whole. In LDA[2] the goal is to find an efficient or interesting way to represent the face vector space. But exploiting the class information can be helpful to the identification tasks. This is shown by the diagram below:



(Fig 2.2: LDA versus PCA)

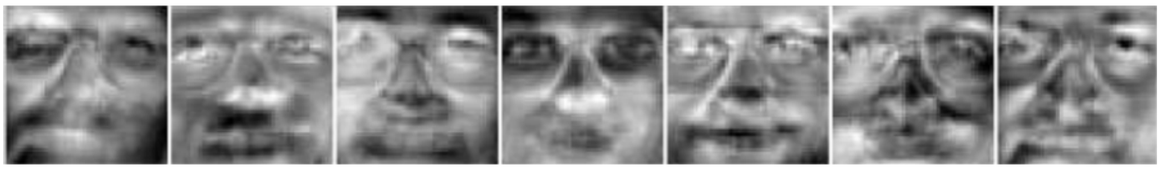
The Fisherface algorithm is derived from the Fisher Linear Discriminant (FLD), which uses class specific information. By defining different classes with different statistics, the images in the learning set are divided into the corresponding classes. Then, techniques similar to those used in Eigenface algorithm are applied. The Fisherface algorithm results in a higher accuracy rate in recognizing faces when compared with Eigenface algorithm.

The Linear Discriminant Analysis finds a transform WLDA, such that:

$$W_{LDA} = \underbrace{arg - max}_{W} \frac{W^T S_B W}{W^T S_W W}$$

Where S_B represents the between class scatter matrix and S_W represents the within class scatter matrix.

The image below gives the basis images (Fisherfaces) for the ORL database:

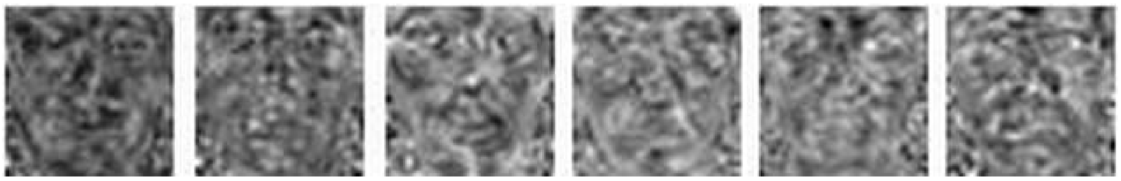


(Fig 2.3: Fischer Faces)

Amongst similar methods and other comparable methods, Fisherfaces seems to be the best in interpolating and extrapolating in the face of lighting variations. It can also handle simultaneous changes in lighting conditions and facial expression better than algorithms such as PCA.

2.1.4 Laplacian Faces

The Laplacianfaces approach [17] is another appearance based face recognition method. By using Locality Preserving Projections (LPP), the face images are mapped into a face subspace for analysis. Different from Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) which effectively see only the Euclidean structure of face space, LPP finds an embedding that preserves local information, and obtains a face subspace that best detects the essential face manifold structure. The Laplacianfaces are the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the face manifold. In this way, the unwanted variations resulting from changes in lighting, facial expression, and pose may be eliminated or reduced. Theoretical analysis shows that PCA, LDA and LPP can be obtained from different graph models. Laplacianfaces has proved better than both the Fisherfaces and Eigenfaces. The image below shows the Laplacianfaces for the ORL database:



(Fig 2.4 Laplacianfaces)

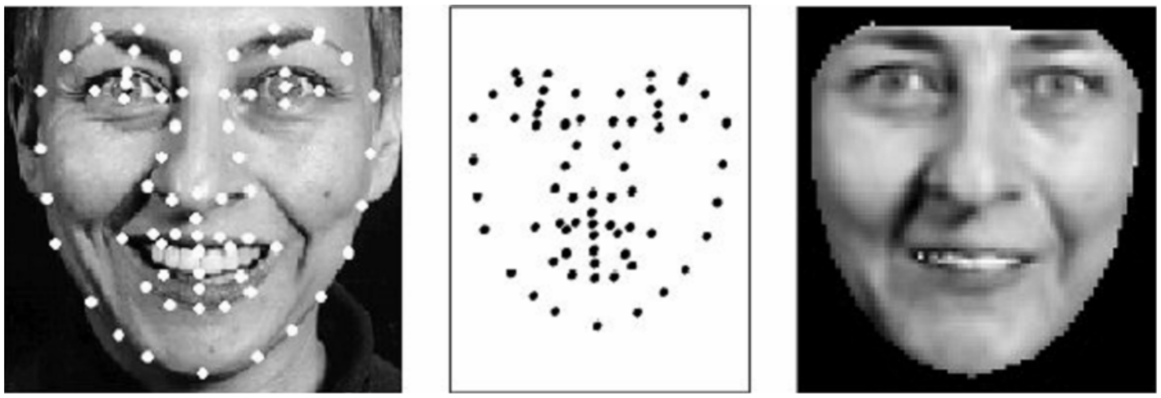
2.1.5 Kernel PCA [11][12]

This method employs the use of Mercer Kernels along with the regular PCA algorithm. The use of Kernels makes mapping better. And thus KPCA gives better results as compared to PCA.

2.1.6 Active Appearance Model

The AAM is a model based approach [18], which is distinct from the above appearance based approaches. They have their advantages as well as disadvantages.

An Active Appearance Model (AAM) is an integrated statistical model which combines a model of shape variation with a model of the appearance variations in a shape-normalized frame. An AAM contains a statistical model of the shape and gray-level appearance of the object of interest which can generalize to almost any valid example. Matching to an image involves finding model parameters which minimize the difference between the image and a synthesized model example, projected onto the image. The potentially large number of parameters makes this a difficult problem.
The AAM is constructed based on a training set of labeled images, where landmark points are marked on each example face at key positions to outline the main features. Consider this in the example below:


(Fig 2.5: The training image is split into shape and shape-normalized texture)

The shape of a face is represented by a vector consisting of the positions of the landmarks, S = (x_1 y_1, ….. , x_n y_n), where ($x_j y_j$) denotes the 2-D image co-ordinate of the jth benchmark.

All shape vectors of faces are normalized into a common coordinate system. The principal component analysis is applied to this set of shape vectors to construct the face shape model, denoted as:

$$s = \overline{s} + P_s b_s$$

Where s represents the shape vector, s bar the mean shape vector, P_s is a set of orthogonal modes of face variation and b_s is a set of shape parameters.

In order to construct the appearance model, the example image is warped to make the control points match the mean shape. Then the warped image region covered by the mean shape is sampled to extract the gray level intensity (texture) information. Similar to the shape model construction, a vector is generated as the representation, g =(I_1, ….. ,I_m)^T. Where Ij denotes the intensity of the sampled pixel in the warped image.

PCA is also applied to construct a linear model:

$$g = \overline{g} + P_g b_g$$

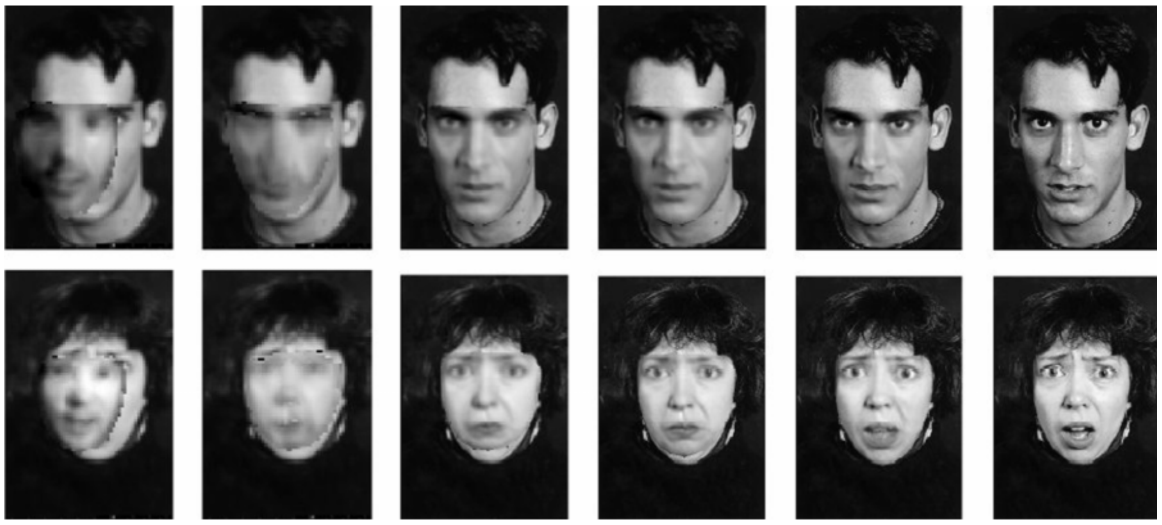
Where g represents the appearance vector, g bar the mean appearance vector, P_s is a set of orthogonal modes of grey level variation and b_s is a set of grey-level model parameters.

Thus, all shape and texture of any example face can be summarized by the vectors bs and bg. The combined model is the concatenated version of bs and bg, denoted as follows:

$$b = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix}$$

Where, Ws is a diagonal matrix of weights for each shape parameter, allowing for the difference in units between the shape and gray scale models. PCA is applied to *b* also, *b* = *Qc* , where *c* is the vector of parameters for the combined model.

Given a new image and constructed model, the metric used to measure the match quality between the model and image is $\Delta = |\,\delta I\,|^2$, where δI is the vector of intensity differences between the given image and the image generated by the model tuned by the model parameters, called residuals. The AAM fitting seeks the optimal set of model parameters that best describes the given image. Cootes observed that displacing each model parameter from the correct value induces a particular pattern in the residuals. In the training phase, the AAM learned a linear model that captured the relationship between parameter displacements and the induced residuals. During the model fitting, it measures the residuals and uses this model to correct the values of current parameters, leading to a better fit. Figure below shows examples of the iterative AAM fitting process.

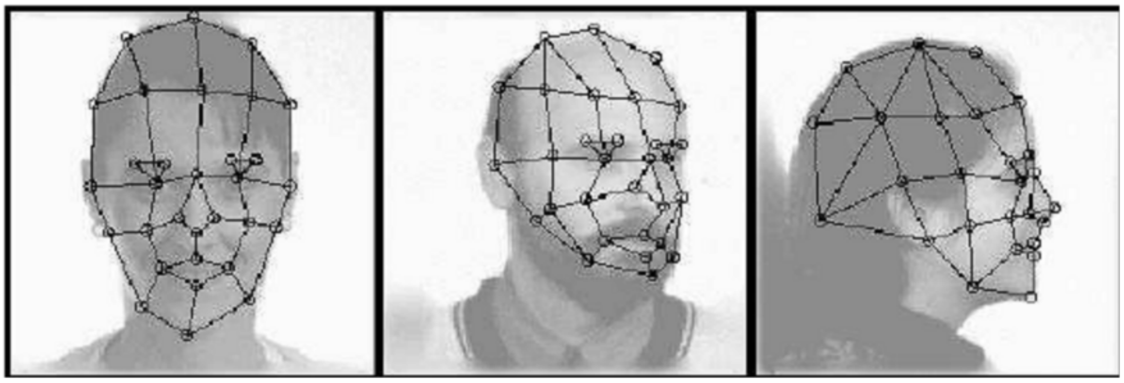


(Fig 2.6: Fitting an AAM)

For all the training images, the corresponding model parameter vectors are used as the feature vectors. The linear discrimination analysis (LDA) is utilized to construct the discriminant subspace for face identity recognition. Given a query image, the AAM fitting is applied to extract the corresponding feature vector. The recognition is achieved by finding the best match between the query feature vector and stored prototype feature vectors, both of which are projected onto the discriminant subspace.

2.1.7 Elastic Bunch Graph

All human faces share a similar topological structure. Wiskott et al [16]. Present a general in-class recognition method for classifying members of a known class of objects. Faces are represented as graphs, with nodes positioned at fiducial points (such as the eyes, the tip of the nose, some contour points, etc.; see Fig.2.7), and edges labeled with 2-D distance vectors.



(Fig 2.7: Elastic Bunch Graph)

Each node contains a set of 40 complex Gabor wavelet coefficients, including both phase and magnitude, known as a jet. Wavelet coefficients are extracted using a family of Gabor kernels with 5 different spatial frequencies and 8 orientations; all kernels are normalized to be of zero mean. Face recognition is based on labeled graphs. A labeled graph is a set of nodes connected by edges; nodes are labeled with jets; edges are labeled with distances. Thus, the geometry of an object is encoded by the edges while the gray value distribution is patch-wise encoded by the nodes (jets). While individual faces can be represented by simple labeled graphs, a face class requires a more comprehensive representation in order to account for all kinds of variations within the class. The Face Bunch Graph has a stack-like structure that combines graphs of individual sample faces, as demonstrated in Fig. 20. It is crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points. All jets referring to the same fiducial point, e.g., all left-eye jets are bundled together in a bunch, from which one can select any jet as an alternative description. The left-eye bunch might contain a male eye, a female eye, both closed and open, etc. Each fiducial point is represented by such a set of alternatives and from each bunch any jet can be selected independently of the jets selected from the other bunches. This provides full combinatorial power of this representation even if it is constituted only from a few graphs.

2.2 Classification Task

Classification can be done using either simple distance measures or learning theory based methods such as support vector machines.

For classification using distance metrics we classify a face as correctly recognized if the term $\mathbf{e_r = \min_l \|\Omega - \Omega^l\|}$, which is the minimum of the norm for the $\mathbf{\Omega}$ of each training image with the $\mathbf{\Omega}$ for an incoming image to be recognized is below a carefully chosen threshold $\mathbf{\Theta}$.

2.2.1 Distance Classifiers

The norm mentioned above could be any of the standard distance metrics [11][13], some of the most popular in classification tasks are:

1. The Manhattan or City-Block Norm:

It is also known as Manhattan distance, boxcar distance, absolute value distance. It represents distance between points in a city road grid. It examines the absolute differences between coordinates of a pair of objects.

$$\| x - y\|_C B = \sum_{i=1}^{D} | x_i - y_i|$$

2. The Euclidean Norm:

The Euclidean Norm is a special case of a general class of norms called the Minkowski Metrics and is given by:

$$\|x - y\|_e = (\sum_{i=1}^{D} |x_i - y_i|^2)^{1/2}$$

3. The Mahalanobis Distance:

The Mahalanobis distance is much superior to the Euclidean norm for the inclusion of the covariance between the datasets being compared.

It is given as:

$$d(x,y) = ((x-y)^T C^{-1}(x-y))^{1/2}$$

Where C is the covariance matrix, and interestingly when the covariance matrix is an identity matrix the Mahalanobis distance is reduced to the Euclidean distance.

Advantages of using the Mahalanobis Distance [11]:

The traditional Euclidean norm that is used extensively has two major problems.

1. Euclidean distance is extremely sensitive to the scales of the variables involved.
2. The Euclidean distance is blind to correlated variables

The Mahalanobis distance takes into account the covariance between the two variables. This eliminates whatever problems related to scale and correlation that are inherent with the Euclidean distance. It thus is a much better distance measure when it comes to pattern recognition applications.

2.2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information processing unit that has certain performance characteristics in common with biological neural networks. ANNs have been developed as generalizations of mathematical models of human cognition or neural biology based on the assumptions that:

1. Information Processing occurs at simple elements called neurons
2. Signals are passed between neurons over connection links
3. Each connection link has associated weight in which a typical neural net multiplies it with the signal to be passed
4. Each neuron applies to an activation function (linear or non linear) to the net input(a sum of the weighted input signals) to determine the output signals

A neural network is characterized by –

1. Its pattern of interconnection of neurons called the architechture
2. Its method of determining the weights on connections which is called the training algorithm
3. Its activation functions

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this supervised learning, to train a network.

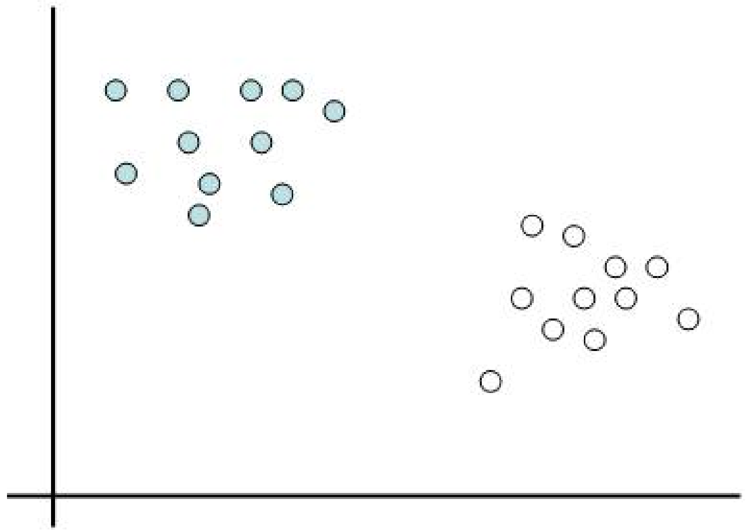
Neural networks, thus, consist of a collection of finite number of simple mathematical elements which perform the task of weighing, summing and giving the output. These outputs are the ones which can be used for making decisions.

Neural Networks can be trained using the hill climbing algorithms such as gradient descent.

2.2.3 Support Vector Machines

Support Vector Machine is a classifier derived from Statistical Learning Theory by Vladimir Vapnik[5] and his co-workers. SVMs embody the principle of structural risk minimization[4][5] which gives them better generalization ability as compared to the empirical risk minimization used by classifiers such as Neural Networks. Support Vector Machines are considered by many to be the best off the shelf supervised learning algorithms available today. The idea in support vector machines is to form a separating hyperplane that is placed most optimally [5].

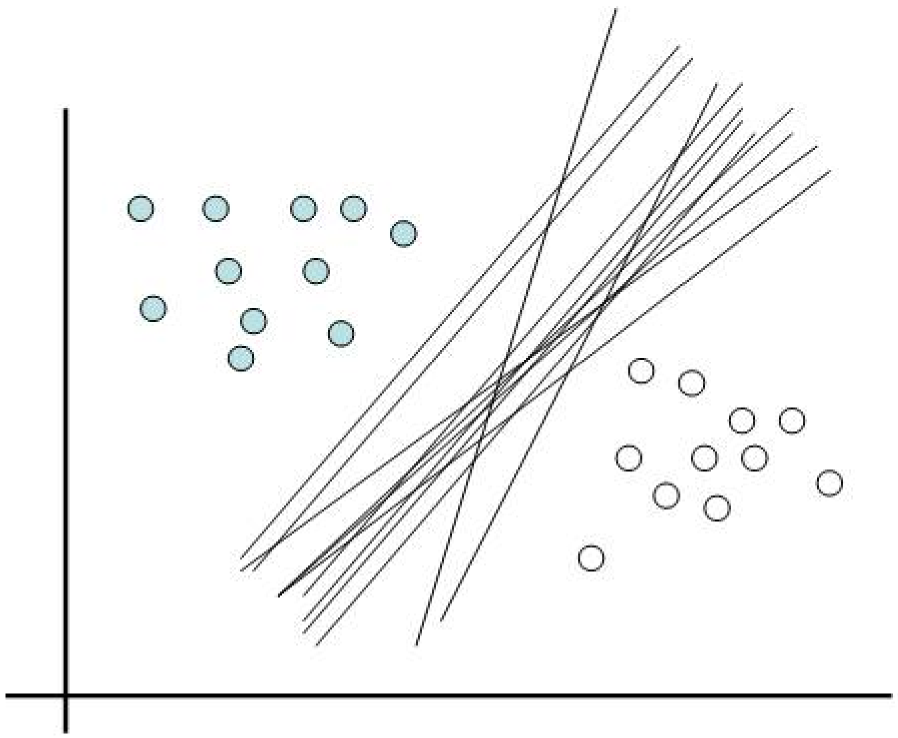
Consider the image below. How would you classify this data?



(Fig 2.8: Data Set)

This set looks linearly separable. That means we could draw a straight line to separate the two classes (indicated by the two different colors). Also note that these data points lie in a two dimensional space, so we talk of a straight line. We could easily graduate to higher dimensions, as an example in a 3-D space we would have spoken of constructing a plane to separate the points and a hyperplane in a $\mathcal{N}$ dimensional space.

Coming back, though we can draw a straight line to separate these two classes, there is a problem. There are an infinite number of candidate lines. Which straight line to choose?



(Fig 2.9: Hyperplanes)

There are two intuitions that lead us to the best hyperplane :

1. Confidence in making the correct prediction [9]: Without getting into much detail (as the point of this post is to see why are support vector machines called so and not what they are), this intuition is formalized by the **functional margin**. The functional margin of a hyperplane given by $wx + b = 0$ w.r.t a specific training example $(x^{(i)}, y^{(i)})$ is defined as:

$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$

If $y^{(i)} = 1$, for a large functional margin (greater confidence in correct prediction) we want $w^T x^{(i)} + b \gg 0$

If $y^{(i)} = -1$, for a large functional margin we want $w^T x^{(i)} + b \ll 0$.

The above captures our first intuition into a single formal statement that we would like the functional margin to be large.

2. Margin [9]: Another intuition about choosing the best hyperplane is to choose one in which the distance from the training points is the maximum. This is formalized by the **geometric margin**. Without getting into the details of the derivation, the geometric margin is given by:

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{||w||}$$

Which is simply the **functional margin normalized**. So these intuitions lead to the **maximum margin classifier** which is a precursor to the SVM.

To sum up: To realize these intuitions and get the best hyperplane, the optimization problem is:

Choose γ, w, b so as to maximize the geometric margin

$max_{\gamma,w,b}\gamma$

Subject to the condition that $y^{(i)}(w^T x^{(i)} + b) > \gamma$ and $||w|| = 1$.

So the Lagrangian for the above optimization problem is constructed:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right].$$

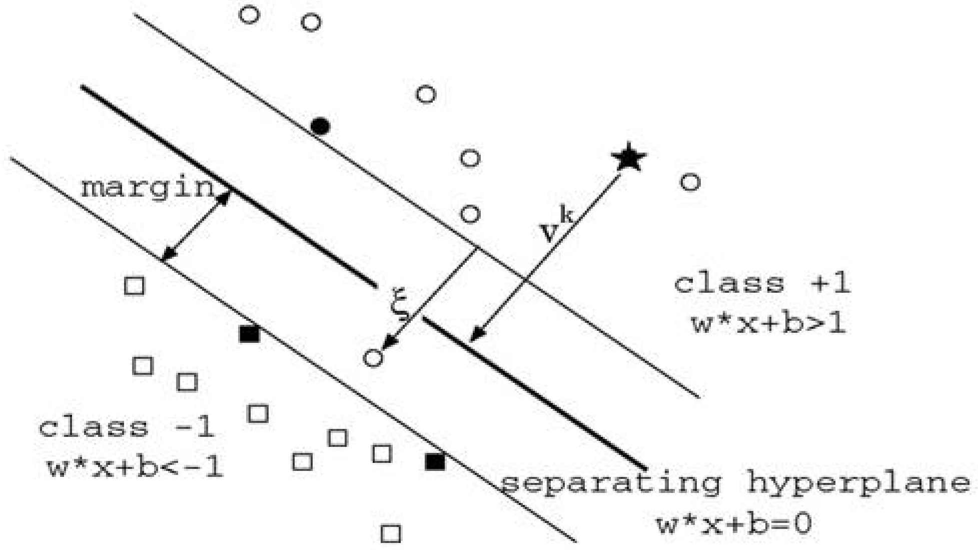
This is the lagrangian, finding the dual of this problem taking into account the KKT conditions:

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

The variable b would be given by:

$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + \min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}.$$

The decision surface is nothing but a weighted combination of the support vectors. In other words, the support vectors decide the nature of the boundary between the two classes. Take a look at the image below:



(Fig 2.9: 2-D hyperplane and Support vectors)

The SVM takes in labeled training examples $\{x_i, y_i\}$, where $x_i$represents the features and $y_i$the class label, that could be either 1 or -1. On training we obtain a set of Support Vectors m, multipliers α_i, $y_i$and the term b. To understand what $b$does, look at the above figure. It is somewhat like the intercept term $c$in the equation of a straight line, $y = mx + c$. The terms $w$and $x$determine the orientation of the hyperplane while $b$determines the actual position of the hyperplane.

As is indicated in the diagram, the linear decision surface is :

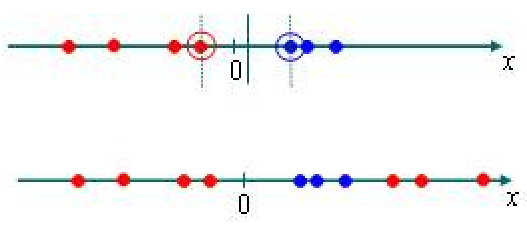
$$w \star x + b = 0 \qquad (1)$$

where $w = \sum_{i=1}^{m} \alpha_i y_i s_i$

where $s_i$are the support vectors.

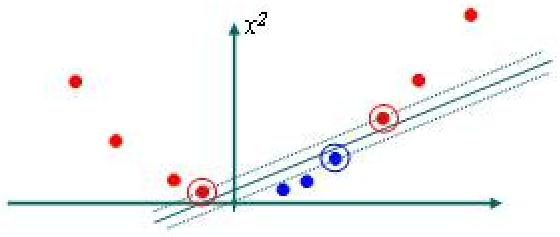
The above holds when the data (classes) is linearly separable. Sometimes however, that's not the case. Take the following example:

The two classes are indicated by the two different colors. The data is clearly not LINEARLY separable.



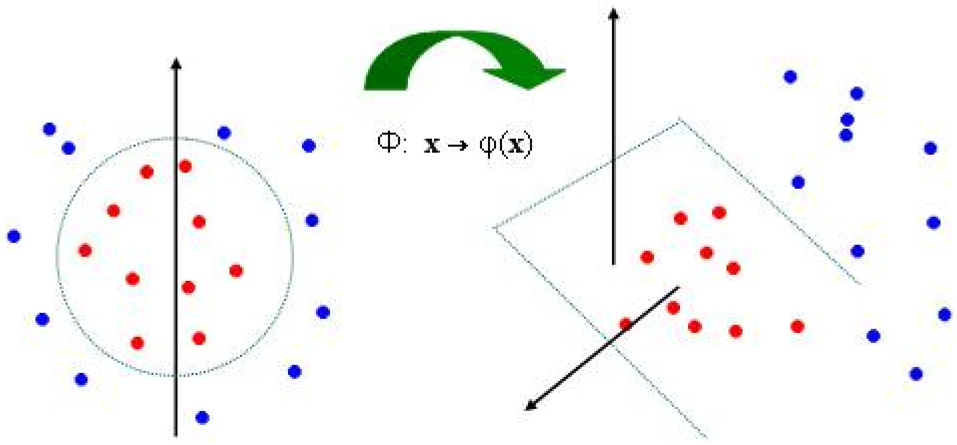
(Fig 2.10 – linearly non-separable data in 1-D)

However when mapped onto two dimensions, a linear decision surface between them can be made with ease.



(Fig 2.11 – linearly non-separable data in 1-D becomes separable in 2-D)

Take another example. In this example the data is not linearly separable in 2-D, so they are mapped onto three dimensions where a linear decision surface between the classes can be made.



(Fig 2.12 – linearly non-separable data in 2D becomes separable in 3D)

By Cover's Theorem [5] it is more likely that a data-set not linearly separable in some dimension would be linearly separable in a higher dimension. The above two examples are simple; sometimes the data might be linearly separable at very high dimensions, maybe at infinite dimensions.

We realize this in the SVM by employing the Kernel trick [12] that is all the inner products are replaced by a mercer Kernel. The various Kernels that may be used are:

1. Radial Basis Kernel
2. Linear Kernel
3. Polynomial Kernel
4. Custom Kernel.

For a general case the optimization problem can be stated as:

$$\begin{aligned}\max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \ \ i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0,\end{aligned}$$

Where, we have to maximize the objective function ***W (α)*** subject to the given constraints. Under these constraints we have to find out the values of the Lagrangian multipliers αi. These help us to find out the values of w and b which give the equation of the separating hyperplane. Also, the inner product between x(i) x(j) can be replaced by any of the following kernels depending on the problem:

1. Linear Kernel function.
2. Polynomial Kernels
3. Radial Basis Function.
4. Use of a Custom Kernel Function.

The choice of the kernel would depend on the problem. In the cases of face recognition, linear kernels and Radial Basis functions are found to have performed well. The above optimization problem is solved using an algorithm called Sequential Minimal Optimization.

The training algorithm will be discussed later.

2.2.4 Reasons for choice of Eigenfaces:

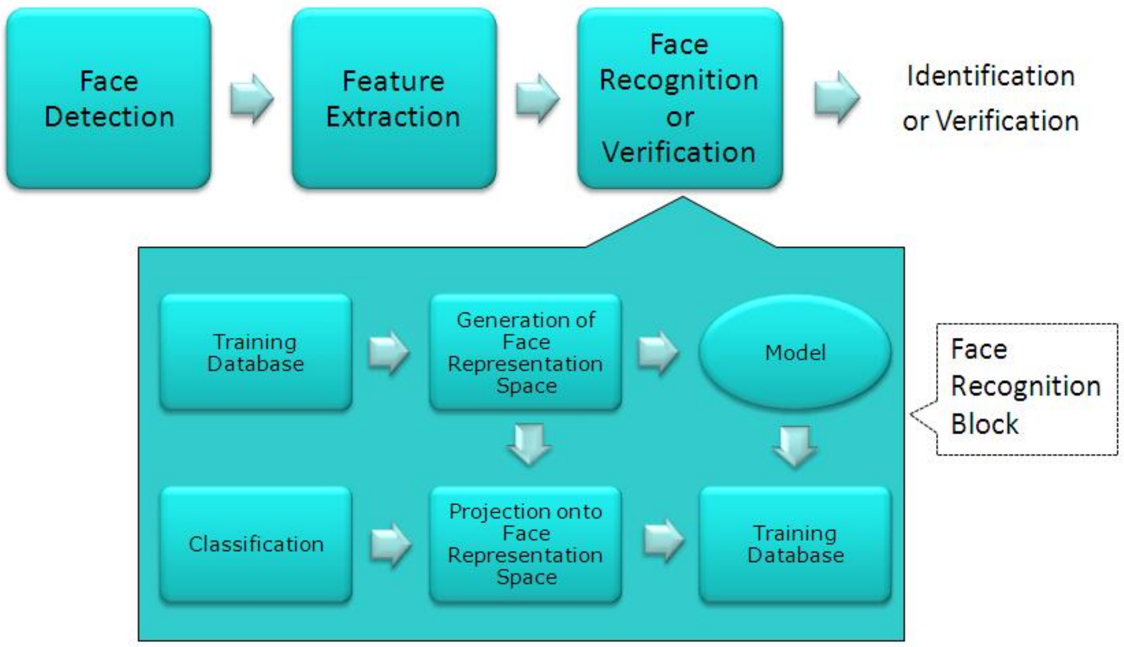
1. Non-iterative global optimization of a natural cost function that implies more mathematical simplicity, which it brings us to the next advantage: better computational efficiency.
2. Computational efficiency [1][3]. When we get the matrix defining the subspace were we will project the samples, the reduction of the face dimension is easy to perform: It just consists in a simple projection.
3. Relatively immune to lighting variations [2]: This can be achieved by simply removing the first three eigenfaces.
4. Occlusion can be avoided by using as many eigenfaces as possible.
5. Feature points are not extracted. That is a difficult task in model based systems.
6. Given the data set is sufficient, predictions for new facial expressions can be made by interpolating and extrapolating between the training set.

2.2.5 Reasons for choice of Support Vector Machines:

1. Use Structural Risk Minimization giving better generalization [5]
2. Optimization is guaranteed
3. Low VC Dimension [4]
4. Computational Cycles are lesser as compared to A.N.N
5. Best off the shelf learning algorithm for classification and regression problems [4][8][9]

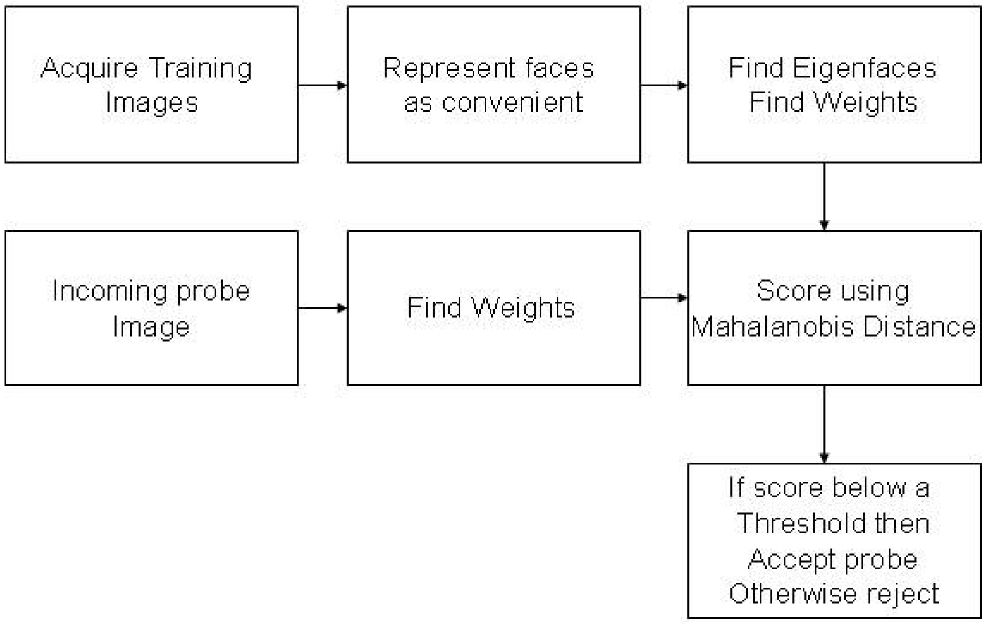
3. Block Diagram

The block diagram for a face authentication system can be as:


Face Detection
Feature Extraction
Face Recognition or Verification
Identification or Verification
Training Database
Generation of Face Representation Space
Model
Face Recognition Block
Classification
Projection onto Face Representation Space
Training Database

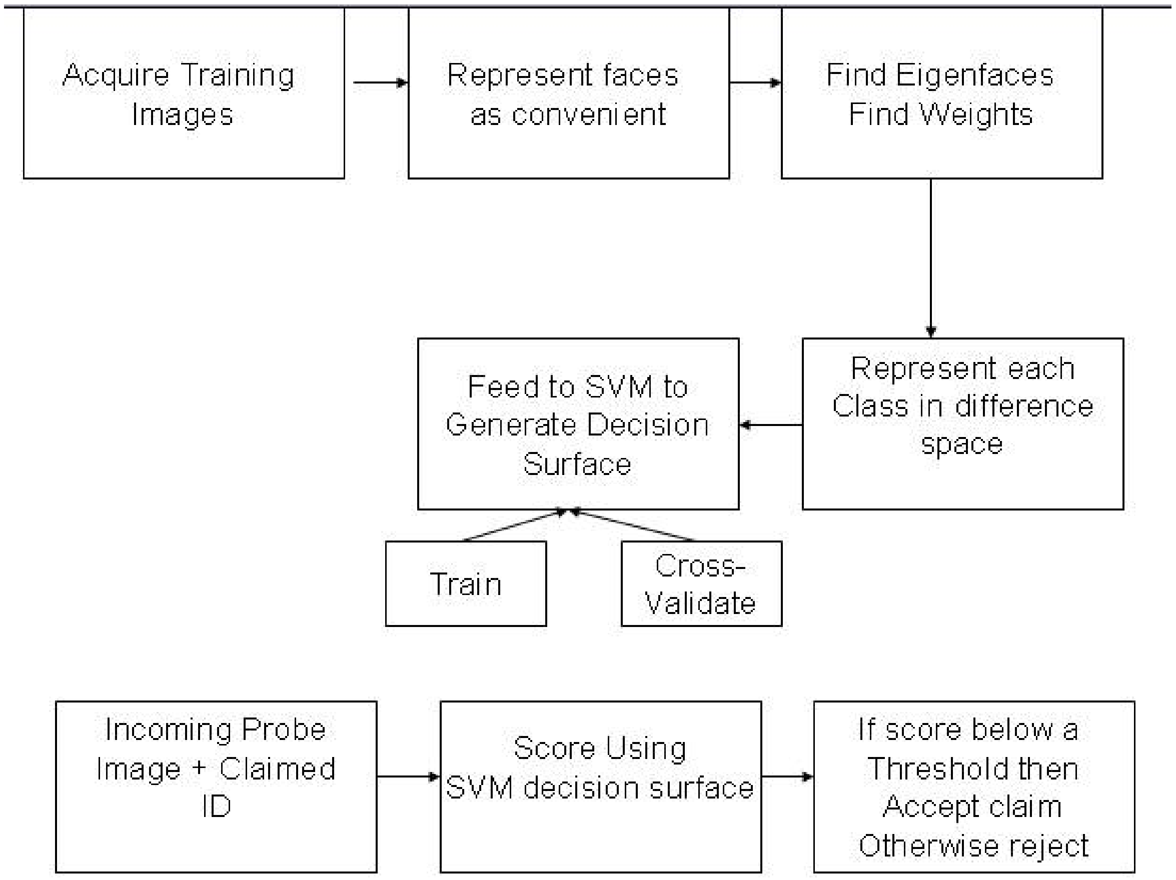
(Fig 3.1)

For the specific case when distance classifiers are used it can be given as:


Acquire Training Images
Represent faces as convenient
Find Eigenfaces Find Weights
Incoming probe Image
Find Weights
Score using Mahalanobis Distance
If score below a Threshold then Accept probe Otherwise reject

(Fig 3.2)

When Support Vector Machines are used and for the purpose, there are a few changes in the system, which are represented in the figure below:


Acquire Training Images
Represent faces as convenient
Find Eigenfaces Find Weights
Represent each Class in difference space
Feed to SVM to Generate Decision Surface
Train
Cross-Validate
Incoming Probe Image + Claimed ID
Score Using SVM decision surface
If score below a Threshold then Accept claim Otherwise reject

(Fig 3.3)

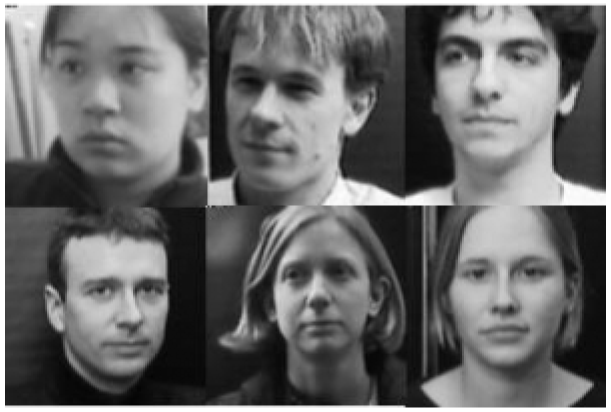
4. Algorithms and System Implementation

Considering the block diagram, we start one block at a time.

4.1 Acquiring Images

Images were acquired in two ways. For the offline system, the images were taken with permission from the MIT-CBCL database and the BITS-Quark database. The system was trained using a set of 10 images for 7 individuals.

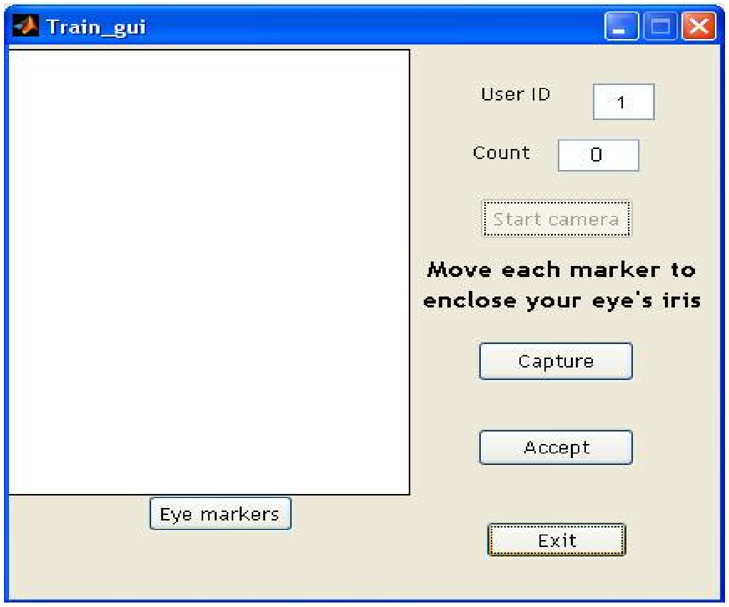
In the online system the images were acquired using the MATLAB image acquisition toolbox. There was an arrangement to keep the images centered as a lack of it would have hit the accuracy greatly. In this case 10 images for 11 individuals were taken.



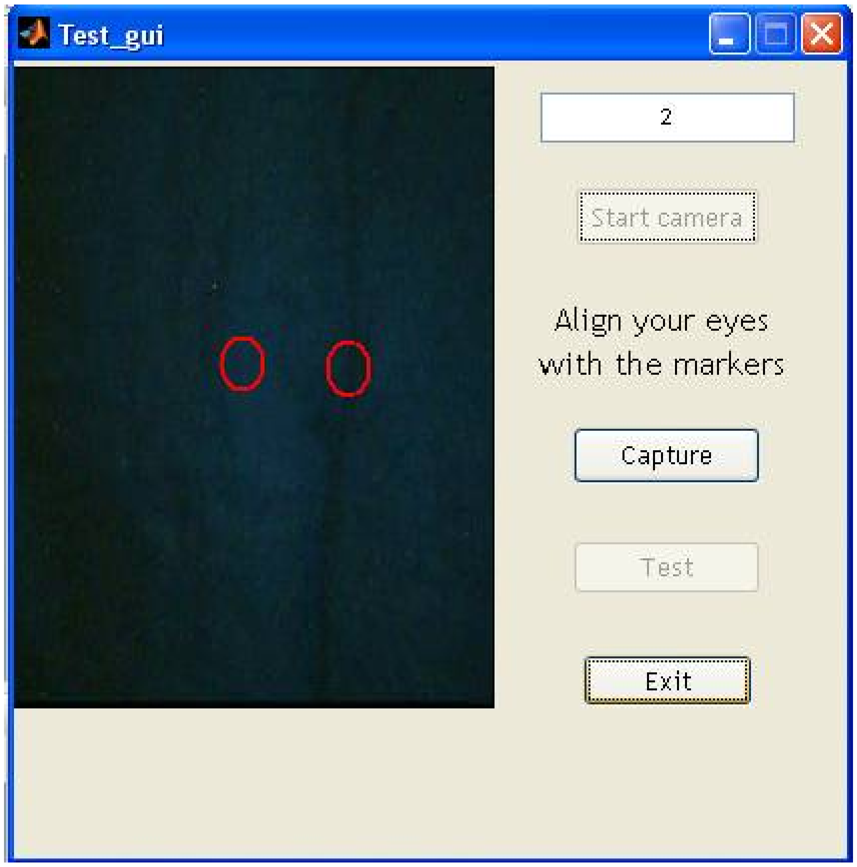
(Fig 4.1: Six sample images from the MIT-CBCL database)

For the online system the images were acquired systematically using MATLAB image acquisition toolbox, the circles were used to adjust the eye (once set during training). The GUI for the same is shown below:

The training images were obtained using the following, the position for eye markers for each individual marked.



The test images were obtained using the following. Note that the position for the eye markers for certain claimed ID is fixed.



(Fig 4.3 Test GUI)

4.2 Finding Eigenfaces

Eigenfaces [1][3] has an interesting parallel in one of the most fundamental ideas in mathematics and signal processing. Fourier series are named so in the honor of Jean Baptiste Joseph Fourier (Generally Fourier is pronounced as “fore-yay”, however the correct French pronunciation is “foor-yay”) who made important contributions to their development. Representation of a signal in the form of a linear combination of complex sinusoids is called the Fourier Series. What this means is that you can’t just split a periodic signal into simple sines and cosines, but you can also approximately reconstruct that signal given you have information how the sines and cosines that make it up are stacked.

<u>More Formally:</u> Put in more formal terms, suppose $f(x)$is a periodic function with period 2πdefined in the interval $c \leq x \leq c+2\pi$and satisfies a set of conditions called the Dirichlet’s conditions:

1. $f(x)$is finite, single valued and its integral exists in the interval.
2. $f(x)$has a finite number of discontinuities in the interval.
3. $f(x)$has a finite number of extrema in that interval.

then $f(x)$can be represented by the trigonometric series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n cos(nx) + b_n sin(nx)] \qquad (1)$$

The above representation of $f(x)$is called the Fourier series and the coefficients a_0, $a_n$and $b_n$are called the fourier coefficients and are determined from $f(x)$by Euler’s formulae. The coefficients are given as :

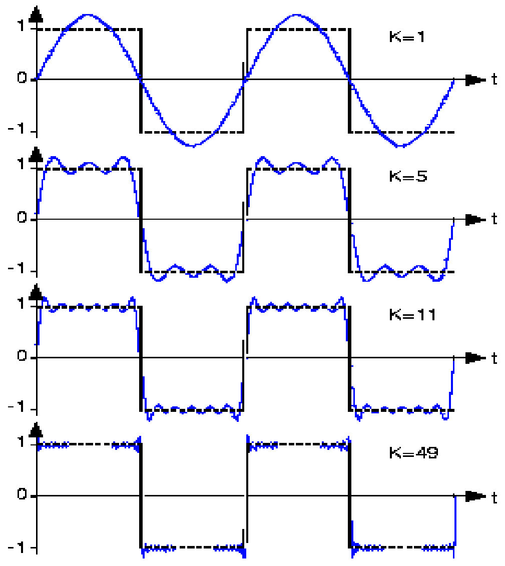
$$a_0 = \frac{1}{\pi}\int_c^{c+2\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_c^{c+2\pi} f(x)cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_c^{c+2\pi} f(x)sin(nx)dx$$

Note: It is common to define the above using $c = -\pi$

An example that illustrates (1)or the Fourier series is:



(Fig 4.4)

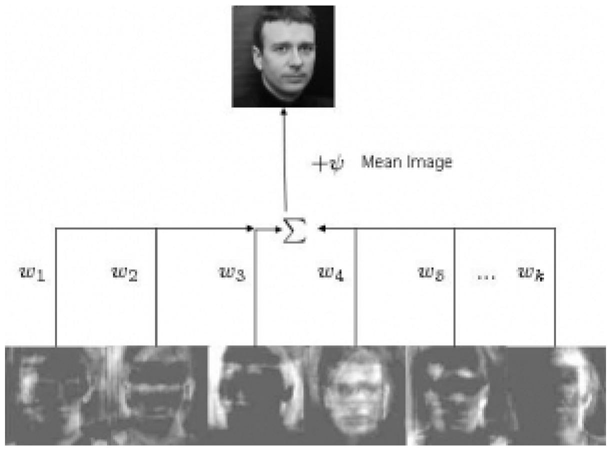
A square wave (given in black) can be approximated to by using a series of sines and cosines (result of this summation shown in blue). Clearly in the limiting case, we could reconstruct the square wave exactly with simply sines and cosines.

Though not exactly the same, the idea behind Eigenfaces is similar. The aim is to represent a face as a linear combination of a set of basis images. That is :

$$\Phi_i = \sum_{j=1}^{K} w_j u_j$$

Where $\Phi_i$represents the i^{th}face with the mean subtracted from it, $w_j$represent weights and $u_j$the eigenvectors. If this makes somewhat sketchy sense then don't worry. This was just like mentioning at the start what we have to do. It would be clear exactly what it is once you finish reading.

This can be represented aptly in a figure as:



(Fig 4.5: Image Reconstruction)

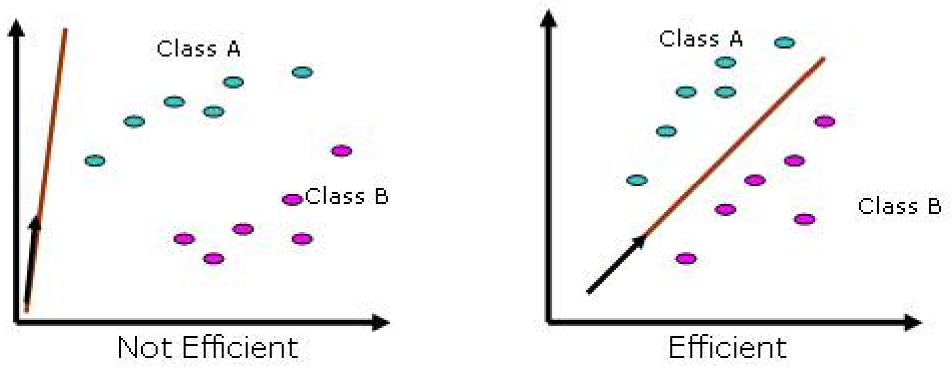
In the above figure, a face that was in the training database was reconstructed by taking a weighted summation of all the basis faces and then adding to them the mean face. Please note that in the figure the ghost like basis images (also called as Eigenfaces, we'll see why they are called so) **are not in order of their importance**. They have just been picked randomly from a pool of 70 by us. These Eigenfaces were prepared using images from the MIT-CBCL database (also the brightness of the Eigenfaces to make them clearer after obtaining them, therefore the brightness of the reconstructed image looks different than those of the basis images).

An Information Theory Approach:

First of all the idea of Eigenfaces considers face recognition as **a 2-D recognition problem**; this is based on the assumption that at the time of recognition, **faces will be mostly upright and frontal**. Because of this, detailed 3-D information about the face is not needed. This reduces complexity by a significant bit.

Before the method for face recognition using Eigenfaces was introduced, most of the face recognition literature dealt with local and intuitive features, such as distance between eyes, ears and similar other features. This wasn't very effective. Eigenfaces inspired by a method used in an earlier paper was a significant departure from the idea of using only intuitive features. It uses an Information Theory approach wherein the most relevant face information is encoded in a group of faces that will best distinguish the faces. It transforms the face images in to a set of basis faces, which essentially are the principal components of the face images.

The Principal Components [10][13] basically seek directions in which it is more efficient to represent the data. This is particularly useful for reducing the computational effort. To understand this, suppose we get 60 such directions, out of these about 40 might be insignificant and only 20 might represent the variation in data significantly, so for calculations it would work quite well to only use the 20 and leave out the rest. This is illustrated by this figure:



(Fig 4.6: Principal Components)

Such an information theory approach will encode not only the local features but also the global features. **Such features may or may not be intuitively understandable**. When we find the principal components or the Eigenvectors of the image set, each Eigenvector has some contribution from EACH face used in the training set. So the Eigenvectors also have a face like appearance. These look ghost like and are ghost images or Eigenfaces. Every image in the training set can be represented as a weighted linear combination of these basis faces.

The number of Eigenfaces that we would obtain therefore would be equal to the number of images in the training set. Let us take this number to be M. Like I mentioned one paragraph before, some of these Eigenfaces are more important in encoding the variation in face images, thus we could also approximate faces using only the K most significant Eigenfaces.

Assumptions:

1. There are M images in the training set.

2. There are K most significant Eigenfaces using which we can satisfactorily approximate a face. Needless to say K < M.

3. All images are $N \times N$ matrices, which can be represented as $N^2 \times 1$ dimensional vectors. The same logic would apply to images that are not of equal length and breadths. To take an example: An image of size 112 x 112 can be represented as a vector of dimension 12544 or simply as a point in a 12544 dimensional space.

Algorithm for Finding Eigenfaces:

1. Obtain M training images I_1, I_2… I_M, it is very important that the images are centered.

(Fig 4.7: Training Images from MIT-CBCL database)

2. Represent each image I_i as a vector Γ_i as discussed above.

$$I_i = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}_{N \times N} \xrightarrow{\text{concatenation}} \begin{bmatrix} a_{11} \\ \vdots \\ a_{1N} \\ \vdots \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix}_{N^2 \times 1} = \Gamma_i$$

3. Find the average face vector Ψ.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

4. Subtract the mean face from each face vector Γ_i to get a set of vectors Φ_i. The purpose of subtracting the mean image from each image vector is to be left with only the distinguishing features from each face and “removing” in a way information that is common.

$$\Phi_i = \Gamma_i - \Psi$$

5. Find the Covariance matrix C:

$$C = AA^T\text{, where } A = [\Phi_1, \Phi_2 \ldots \Phi_M]$$

Note that the Covariance matrix has simply been made by putting one modified image vector obtained in one column each.

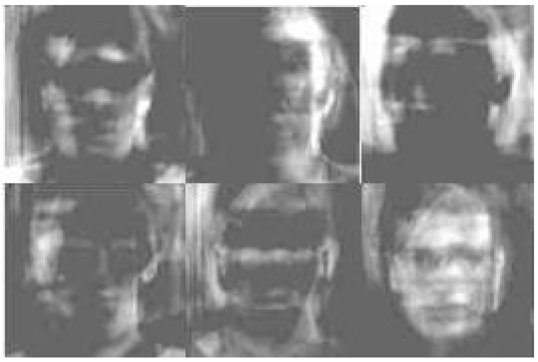
Also note that $C$is a $N^2 \times N^2$matrix and $A$is a $N^2 \times M$matrix.

6. We now need to calculate the Eigenvectors u_i of C, However note that C is a $N^2 \times N^2$matrix and it would return N^2Eigenvectors each being N^2dimensional. For an image this number is huge. The computations required would easily make your system run out of memory. How do we get around this problem?

7. Instead of the Matrix AA^Tconsider the matrix A^TA. Remember $A$is a $N^2 \times M$matrix, thus $A^TA$is a $M \times M$matrix. If we find the Eigenvectors of this matrix, it would return MEigenvectors, each of Dimension $M \times 1$, let’s call these Eigenvectors v_i.

Now from some properties of matrices, it follows that: $u_i = Av_i$. We have found out $v_i$earlier. This implies that using $v_i$we can calculate the M largest Eigenvectors of AA^T. Remember that $M \ll N^2$as M is simply the number of training images.

8. Find the best MEigenvectors of $C = AA^T$by using the relation discussed above. That is: $u_i = Av_i$. Also keep in mind that $\|u_i\| = 1$.



(Fig 4.8:6 Eigenfaces for the training set chosen from the MIT-CBCL database, these are not in any order]

9. Select the best K Eigenvectors, the **selection of these Eigenvectors is done heuristically**.

4.3 Finding Weights

The Eigenvectors found at the end of the previous section, u_i when converted to a matrix in a process that is reverse to that in STEP 2, have a face like appearance. **Since** these are Eigenvectors and have a face like appearance, they are called Eigenfaces. Sometimes, they are also called as **Ghost Images** because of their weird appearance.

Now each face in the training set (minus the mean), Φ_i can be represented as a linear combination of these Eigenvectors u_i:

$\Phi_i = \sum_{j=1}^{K} w_j u_j$m, where u_j's are Eigenfaces.

These weights can be calculated as:

$$w_j = u_j^T \Phi_i.$$

Each normalized training image is represented in this basis as a vector.

$$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$

where i = 1,2… M. This means we have to calculate such a vector corresponding to every image in the training set and store them as templates.

4.4 Recognition Task using Distance Classifiers

Now consider we have found out the Eigenfaces for the training images , their associated weights after selecting a set of most relevant Eigenfaces and have stored these vectors corresponding to each training image.

If an unknown probe face $\Gamma$is to be recognized then:

1. We normalize the incoming probe Γas $\Phi = \Gamma - \Psi$.

2. We then project this normalized probe onto the Eigenspace (the collection of Eigenvectors/faces) and find out the weights.

$$w_i = u_i^T \Phi.$$

3. The normalized probe Φcan then simply be represented as:

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}$$

After the feature vector (weight vector) for the probe has been found out, we simply need to classify it. For the classification task we could simply use some distance measures or use some classifier like Support Vector Machines. In case we use distance measures, classification is done as:

Find $e_r = min \left\|\Omega - \Omega_i\right\|$. This means we take the weight vector of the probe we have just found out and find its distance with the weight vectors associated with each of the training image.

And if $e_r < \Theta$, where $\Theta$is a threshold chosen heuristically, then we can say that the probe image is recognized as the image with which it gives the lowest score.

If however $e_r > \Theta$then the probe does not belong to the database. I will come to the point on how the threshold should be chosen.

For distance measures the most commonly used measure is the Euclidean Distance. The other being the Mahalanobis Distance. The Mahalanobis distance generally gives superior performance. Let's take a brief digression and look at these two simple distance measures and then return to the task of choosing a threshold.

4.5 Authentication Task using Support Vector Machines [4][6][7]

4.5.1 Representation in Difference Space [6]

SVMs are binary classifiers, that is - they give the class which might be 1 or -1, so we would have to modify the representation of faces a little bit than what we were doing in that previous post to make it somewhat more desirable. In the previous approach that is "a view based or face space approach", each image was encoded separately. Here, **we would change the representation and encode faces into a difference space**. The difference space takes into account the dissimilarities between faces.

In the difference space there can be **two different classes**.

1. The class that encodes the dissimilarities between different images of the same person,

2. The other class encodes the dissimilarities between images of other people. These two classes are then given to a SVM which then generates a decision surface.

As I wrote earlier, Face recognition traditionally can be thought of as a $\mathcal{K}$ class problem and face authentication can be thought of as a $\mathcal{K}$ instances two class problem. To reduce it to a two class problem we formulate the problem into a difference space as I have already mentioned.

Now consider a training set $\mathcal{T} = \{ t_1, \ldots, t_M \}$ having M training images belonging to $\mathcal{K}$ individuals. Each individual can have more than one image, that means $M > \mathcal{K}$ of course. It is from $\mathcal{T}$ that we generate the two classes I mentioned above.

1. **The within class differences set.** This set takes into account the differences in the images of the same class or individual. In more formal terms:

$$\mathcal{C}_1 = \{ t_i - t_j | t_i \backsim t_j \}$$

Where t_i and t_j are images and $t_i \backsim t_j$ indicates that they belong to the same person.

This set contains the differences not just for one individual but for all $\mathcal{K}$ individuals.

2. **The between class differences set.** This set gives the dissimilarities of different images of different individually. In more formal terms:

$$\mathcal{C}_2 = \{ t_i - t_j | t_i \nsim t_j \}$$

Where t_i and t_j are images and $t_i \nsim t_j$ indicates that they do not belong to the same person.

The two classes C_1 and C_2 are input to the SVM that it then to be cross validated and trained to carry out the recognition/authentication task; the SVM will generate a decision surface. A decision on the incoming probe will be taken after this.

4.5.2 Face Authentication

For Authentication the incoming probe $p$and a claimed identity $i$is presented.

Using this, we first find out the similarity score:

$$\delta = \sum_{i=1}^{m} \alpha_i y_i K(s_i, ClaimedID - p) + b$$

We then accept this claim if it lies below a certain threshold Δor else reject it. Δ is to be found heuristically.

4.5.3 Face Recognition

Consider a set of images $\mathcal{T} = \{\ t_1, \ldots, t_M\}$, and a probe $p$which is to be indentified.

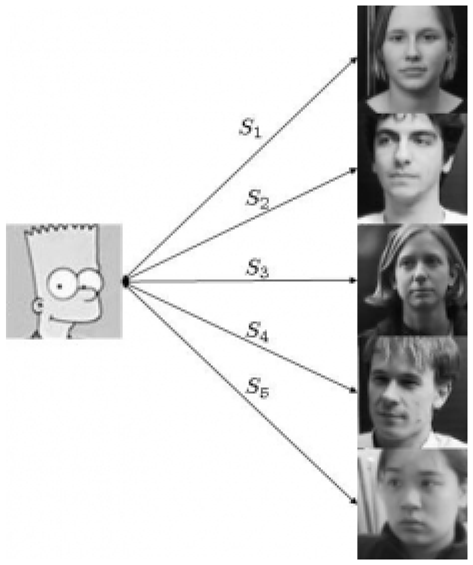
We take $p$and score it with every image in the set t_i:

$$\delta = \sum_{i=1}^{m} \alpha_i y_i K(s_i, t_i - p) + b$$

The image with the lowest score but below a threshold is recognized.

4.5.4 Need for Threshold

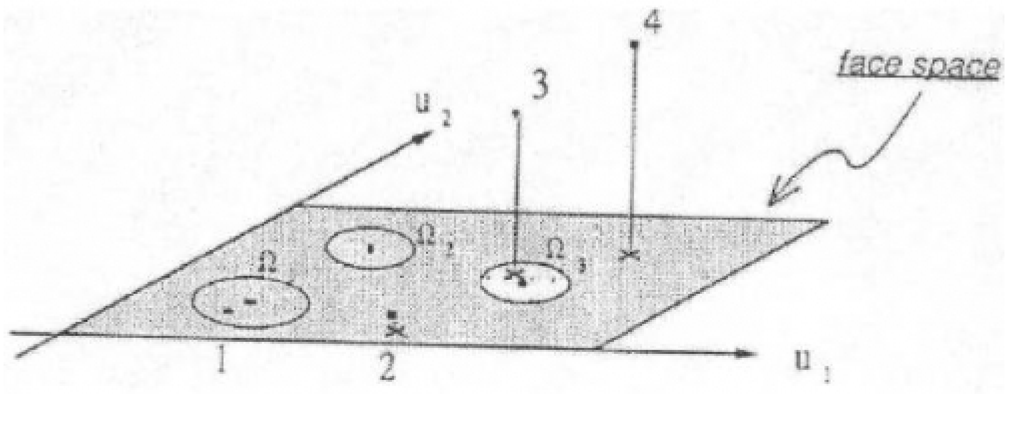
Consider for simplicity we have ONLY 5 images in the training set. And a probe that is not in the training set comes up for the recognition task. The score for each of the 5 images will be found out with the incoming probe. And even if an image of the probe is not in the database, it will still say the probe is recognized as the training image with which its score is the lowest. Clearly this is an anomaly that we need to look at. It is for this purpose that we decide the threshold. The threshold Θ is decided heuristically.



(Fig 4.9: Need for Threshold)

Now to illustrate the above, consider a simpson image as a non-face image, this image will be scored with each of the training images. Let's say $S_4$is the lowest score out of all. But the probe image is clearly not beloning to the database. To choose the threshold we choose a large set of random images (both face and non-face), we then calculate the scores for images of people in the database and also for this random set and set the threshold Θ accordingly.

4.5.5 Note on Face-Space



(Fig 4.10: Face Space)

Consider a simplified representation of the face space as shown in the figure above. The images of a face and in particular the faces in the training set should lie near the face space. Which in general describes images that are face like.

The projection distance ϵ_f should be under a threshold Θ as already seen. The images of known individual fall near some face class in the face space.

There are four possible combinations on where an input image can lie:

1. Near a face class and near the face space: This is the case when the probe is nothing but the facial image of a known individual (known = image of this person is already in the database).

2. Near face space but away from face class: This is the case when the probe image is of a person (i.e a facial image), but does not belong to anybody in the database i.e away from any face class.

3. Distant from face space and near face class : This happens when the probe image is not of a face however it still resembles a particular face class stored in the database.

4. Distant from both the face space and face class: When the probe is not a face image i.e is away from the face space and is nothing like any face class stored.

Out of the four, type 3 is responsible for most false positives. To avoid them, face detection is recommended to be a part of such a system**.**

4.5.6 Cross Validation of SVM [9]

Cross validation is a concept developed from statistical learning theory [5]. Its objective is to choose a hypothesis with the best generalization ability not just the least training error.
Suppose we are given a training set S. Given what we know about empirical risk minimization, here's what might initially seem like a algorithm, resulting from using empirical risk minimization for model selection:
1. Train each model Mi on S, to get some hypothesis hi.
2. Pick the hypotheses with the smallest training error.

This algorithm does not work. Consider choosing the order of a polynomial. The higher the order of the polynomial, the better it will _t the training set S, and thus the lower the training error. Hence, this method will always select a high-variance, high-degree polynomial model, which we saw previously is often poor choice.
Here's an algorithm that works better. In hold-out cross validation (also called simple cross validation), we do the following:
1. Randomly split S into Strain (say, 70% of the data) and Scv (the remaining 30%). Here, Scv is called the hold-out cross validation set.
2. Train each model Mi on Strain only, to get some hypothesis hi.
3. Select and output the hypothesis hi that had the smallest error on the hold out cross validation set.
By testing on a set of examples Scv that the models were not trained on, we obtain a better estimate of each hypothesis hi's true generalization error, and can then pick the one with the smallest estimated generalization error. Usually, somewhere between 1=4 to 1=3 of the data is used in the hold out cross validation set, and 30% is a typical choice.

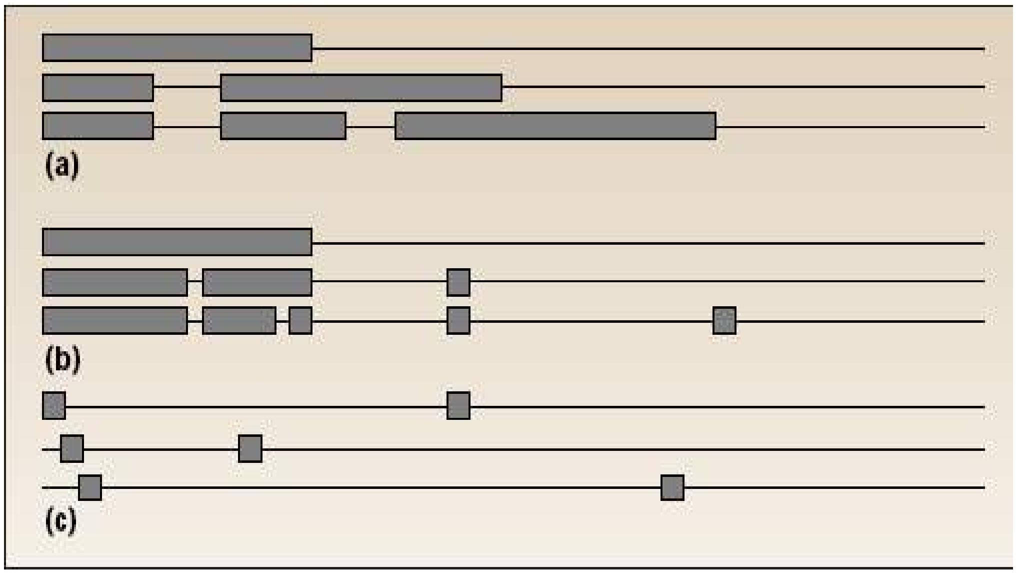
Some other cross validation methods are:
1. K-Fold Cross Validation
2. Leave-one-out Cross validation.

These work well when the number of samples per class is very scarce. In our system, we employed hold-out cross validation.

4.4.7 Training the SVM

For training the SVM there are a number of Algorithms:
1. Projected Gradient Conjugated Chunking Algorithm [9]
2. Oslun’s Algorithm [9]
3. Sequential Minimal Optimization [14].

The SMO algorithm by John Platte is the fastest training algorithm in use for SVMs. As in indicated by the figure below:



(Fig 4. 11: Comparison of the three training Algorithms)

The reason for the selection of SMO [8] over the other two algorithms can be understood with the aid of this diagram.

In this figure, (a) represents PCG Chunking (b) represents Osuna's algorithm and (c) represents the SMO algorithm.

Each algorithm has three stages in training.

The three faint lines for each represent the total training set. The blocks represent the duration to converge to the optimal value of the Lagrangian multiplier Alpha.

Clearly in the third case the convergence is attained very rapidly. In many cases the the SMO algorithm is as fast as 1000 times the PCG chunking algorithm.

The sequential minimal optimization algorithm is actually a deviation of the usual co-ordinate ascent method in which an unconstraint function is optimized w.r.t only one dependent variable taking all others as constant. The same is repeated for all the dependant variables. Co-ordinate ascent as such will not work for solving the dual optimization problem that was posed earlier for the presence of the following constraint

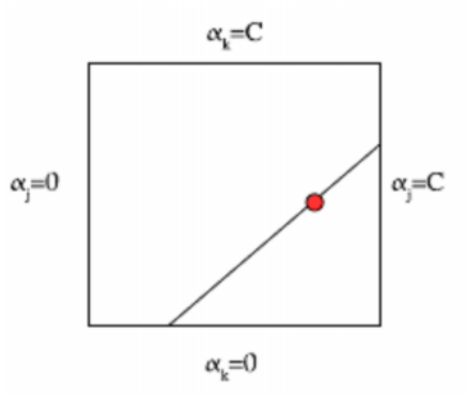
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

The presence of this constraint means that if suppose we were to optimize the objective function with respect to only α_1 keeping others constant, we may represent the constraint as

$$\alpha_1 y^{(1)} = -\sum_{i=2}^{m} \alpha_i y^{(i)}$$

This means that we can not optimize the objective function w.r.t any α_i without violating the constraint. Hence the SMO algorithm relies on optimizing the objective function w.r.t two dependent variables without violating the constraint imposed on it.

The allowed values for α are found on a line and inside a square. The square is basically decided by the KKT conditions which impose a bound on the value of α given by C. These are referred to as the box constraints.



The above is the case when $y_k \neq y_j$. Where y Є {1,-1} are target variables.

The SMO algorithm simply does the following:

```
Repeat till convergence {
1. Select some pair $\alpha_i$ and $\alpha_j$ to update next (using a heuristic that allows to pick those values of α that will make the maximum progress towards the global maximum)
2. Re-optimize W(α) w.r.t $\alpha_i$ and $\alpha_j$ while holding all the other $\alpha_k$ 's fixed.
}
```

Two important things about the SMO algorithm is the choice of the heuristic with which to choose $\alpha_{'s}$ and the other is the method with which to update the value of b.

The SVM parameters were tuned in the search space using **Random Search with Multiple Restarts**.

To Sum –Up:

The steps to apply SVMs are:

1. Transform the data to the format of the SVM software.
2. Conduct scaling on data.
3. Consider the RBF kernel first.
4. Use cross-validation to find the best parameters C and γ.
5. Use these values of best parameters to train the whole training set.
6. Test for images in the designated test set.

5. Softwares Used

1. MATLAB R2007b

    1. Image Processing toolbox.
    2. Image Acquisition toolbox
    3. Statistics toolbox
    4. GUI Builder

2. SVMLight

A software implementation of SVM in ANSI C with a MATLAB front-end

3. SVM MATLAB toolbox by Gunn

6. Experimental Work and Performance Evaluation

6.1 For offline system:

- Number of Eigenfaces generated : 70
- Number of Eigenfaces taken: 70
- Image Resolution: 115 x 115
- Recognition Accuracy : 100%
- Decision Threshold : Not Considered

6.1.1 Results for Offline System**:**

Image (Test Set) (In bracket – Actual Image)	**Identified as (City-Block)**	**Identified as (Euclidean)**	**Identified as (Mahalanobis)**	**Identified as (SVM)**
1. (9)	**9**	**9**	**9**	**9**
2. (2)	**2**	**2**	**2**	**2**
3. (-)	**6**	**8**	**10**	**10**
4. (-)	**9**	**9**	**10**	**9**
5. (3)	**3**	**3**	**3**	**3**
6. (1)	**1**	**1**	**1**	**1**
7. (10)	**10**	**10**	**10**	**10**
8. (10)	**10**	**10**	**10**	**10**
9. (5)	**5**	**5**	**5**	**5**
10 (6)	**6**	**6**	**6**	**6**

6.2 For Online System:

Camera Specifications:

Lens: 6.0 mm

Resolution: 320 x 240

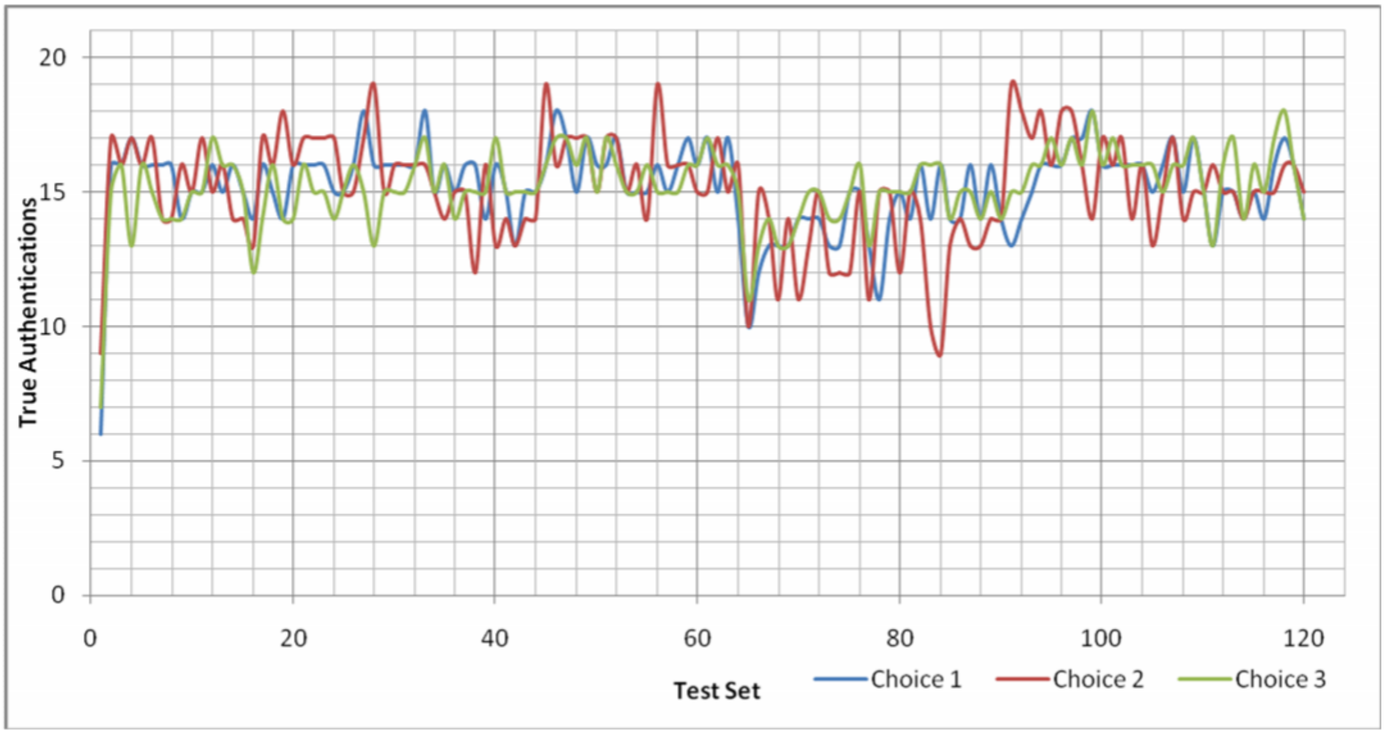
- Number of Eigenfaces generated : 110
- Number of Eigenfaces taken: 98
- Image Resolution: 115 x 115
- Recognition Accuracy in controlled conditions: 85 %

- Recognition Accuracy in un-controlled conditions :
- Recognition Accuracy considering lighting variations:
- Recognition Accuracy considering Occlusion:
- Recognition Accuracy considering Lighting Variations, Un-controlled background environment and occlusions:
- Decision Threshold taken: 0.6535
- EER :
- Average time for cross validation of each combination : 800 seconds
- SVM Parameters :
    - $C = 0.0462$
    - $\gamma = 0.1038$

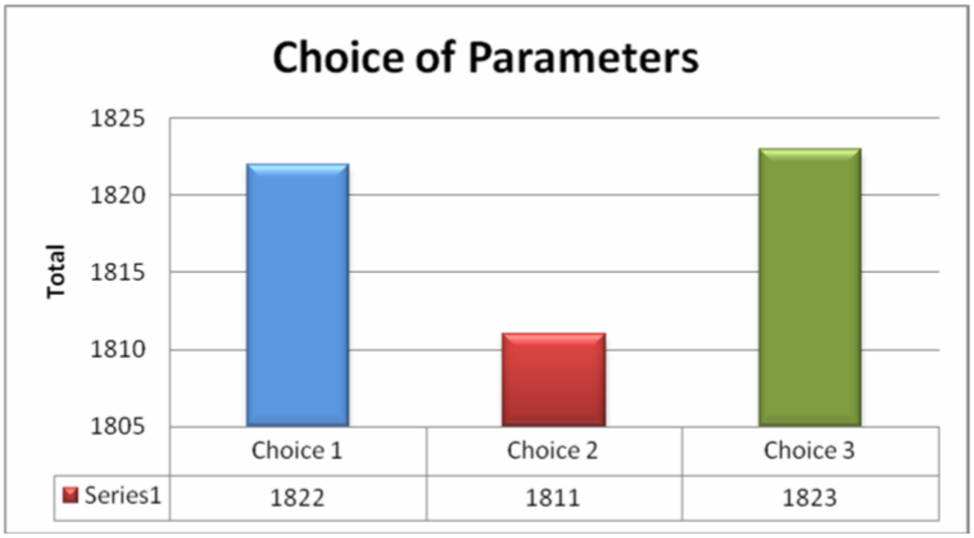
The readings for the cross validation process are as:

Choice 1: C = 0.0462, γ = 0.0692;
Choice 2: C = 0.0462, γ = 0.0173;
Choice 3: C = 0.0462, γ = 0.1038;

Cross Validation Result



(Fig 6. 1)



	Choice 1	Choice 2	Choice 3
🟥 Series1	1822	1811	1823

(Fig 6.2)

Considering the above Results, the values of C and γ were chosen.

7. Applications

1. In e-Commerce Systems
2. Biometric Security Systems
3. Surveillance (Having a database of wanted crooks) (for recognition)
4. Immigration/Customs.
5. Misc. Access Controls.
6. The most likely use of face authentication would be in multi-modal systems. Where it would be combined with some other biometric such as speaker recognition or conversational biometrics for authentication.

8. Possible Improvements and Future Scope

In the future there are a number of improvements that can be affected; some of it would require new research.

1. Study of effect of change in Image resolution on the recognition accuracy. As change in resolution of image can have a marked impact on the recognition accuracy.
2. Using weighted PCA. i.e. Splitting the image into grids and taking PCA of each element separately.
3. Using Kernel PCA, a method involving choosing a Kernel for the first step itself would allow better mapping.
4. Designing a Custom or multiple - weighted Kernel, instead of a Radial Basis Function. This Kernel would be especially suited for face recognition and related tasks.
5. Using and comparing the baseline Eigenfaces system with a parallel and cascaded combination of gabor features and eigenfaces. Such a system has not yet been attempted.
6. Employing a combination of Active Appearance models with Eigenfaces and Support Vector Machines.
7. Implementing the system as per the BANCA recommendations.
8. Supervised selection of the decision threshold instead of a heuristic choice.

Complete Bibliography

1. Matthew A. Turk and Alex P. Pentland, *Face Recognition Using Eigenfaces*, MIT Vision and Modeling Lab, IEEE CVPR ‘91.

2. Belhumeur, Hespanha, *Eigenfaces Versus Fischerfaces: Recognition using Class Specific Linear Projection,* IEEE PAMI ‘97.

3. Matthew A. Turk and Alex P. Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience ‘91.

4. Christopher J. C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition,* Data Mining and Knowledge Discovery, ’99.

5. Vladimir Vapnik, *The Nature of Statistical Learning Theory (Book),* Springer ‘99.

6. P. J. Phillips, *Support Vector Machines Applied to Face Recognition*, neural Information Processing Systems ‘99.

7. Vladimir Vapnik, *Feature Selection for SVM*, Neural and information processing systems proceedings ‘95.

8. Marti Hearst, *Support Vector Machines*, IEEE Intelligent Systems, ‘05.

9. Andrew Ng, *Lecture Notes for Machine Learning, CS 229*, Stanford University, ‘07

10. Lindsay Smith, A Tutorial on PCA.

11. R. Duda, P. Hart, and D. Stork, *Pattern Classification (2nd Edition),* Wiley-Interscience, ‘00.

12. Bernhard Scholkopf, Alexander J. Smola, *Learning With Kernels,* The MIT Press, ’98.

13. Yambor, Draper, Beveridge, *Analyzing PCA based face recognition Algorithms: Eigen vector selection and distance measures.* Colorado State University. ‘00.

14. John C. Platte, *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. Neural Information Processing Systems. 1999.

15. Kailash Karande, Sanjay Talbar, *Face Recognition under Variation of Pose and Illumination using Independent Component Analysis,* IEEE ICGST-GVIP, ’08.

16. L.Wiskott, J. andN, Kr¨uger, and C. D. Malsburg, *Face recognition by elastic bunch graph matching,* IEEE PAMI, ’97.

17. He, Yan, Hu, Niyogi, and Zhang, *Face recognition using laplacianfaces,* IEEE PAMI, ’05.

18. Edwards, Cootes, Taylor, *Face Recognition using Active Appearance Models*, ACM, ’98.

19. Credit is hereby given to the Massachusetts Institute of Technology and to the Center for Biological and Computational Learning for providing the database of facial images.

20. Credit is hereby given to Prof Joachims of the Cornell University for SVMlight

21. Credit is hereby given to Dr Steve Gunn of Univ. Southampton for the SVM MATLAB toolbox.