Frontal View
Human Face Detection and Recognition

This thesis is submitted in partial fulfilment of the requirement for the B.Sc.(Honours) degree in Computer Science.

Lalendra Sumitha Balasuriya

Department of Statistics and Computer Science
University of Colombo
Sri Lanka
May 2000

To my mother and father,
for opening my eyes to the world.

Declaration

I hereby certify that this thesis entitled "Frontal View Human Face Detection and Recognition" is entirely my own work. It has not been submitted nor is it being currently submitted for any other degree.

……

Candidate L. S. Balasuriya

Date ……

……

Supervisor Dr. N. D. Kodikara

Date: ……

Acknowledgements

I sincerely wish to express my appreciation for the valuable help, which I received during the completion of the thesis by the following:

Dr. N. D. Kodikara, for supervising my project, guiding me throughout my research and for his very informative lectures on Computer Graphics and Vision.

Dr. Ruwan Weerasinghe, for his interesting lectures on Intelligent Systems and Mr Gihan Seneviratne, for his lectures on Neural Networks and for the many discussions we had on network models.

Dr Saman Halgamuge, Damminda Alahakoon and Ajantha Atukorale, without whose help and support at crucial times in my research could not have gone so far.

Doug Hundley, for guiding me though the mysterious world of Principal Component Analysis.

Sir Arthur C. Clarke, for the inspiring discussion we had about my project.

Raditha Dissanayake, for lending me his digital camera without which this project would not have been possible.

To all the 'test subjects' who (reluctantly) posed for my human face database.

To my father for drafting my figures and proof reading this dissertation, Rasini for (carelessly) normalising the face database and my mother for her love, support and constant supply of food at all hours.

Methmal, Asankha, Thushan, Sagara, Lilantha, Chanaka, Sulochana, Supun and all my other colleagues and friends at the university for their help, comments and input during my research.

Abstract

This thesis is an attempt to unravel the classical problem of human face recognition. The researcher addressed the problem of automated face recognition by functionally dividing it into face detection and face recognition. Different approaches to the problems of face detection and face recognition were evaluated, and five systems were proposed and implemented using the Matlab technical computing language.

In the implemented frontal-view face detection systems, automated face detection was achieved using a deformable template algorithm based on image invariants. The deformable template was implemented with a perceptron. Unsupervised learning using Kohonen Feature Maps was used to create the Perceptron's A-units. The natural symmetry of faces was utilised to improve the efficiency of the face detection model. The deformable template was run down the line of symmetry of the face in search of the exact face location.

Automated frontal view face recognition was realised using Principal Component Analysis, also known as the Karhunen-Loeve transform. Manual face detection was used to test the implemented automated face recognition system. The frontal view face recognition system is also expanded into a pose invariant face recognition system which is implemented and tested on facial images for subjects with different poses.

The researcher gathered a face database of 30 individuals consisting of over 450 facial images to test fully automated face detection without verification, fully automated face detection with verification, manual face detection and automated face recognition, fully automated face detection and recognition and pose invariant face recognition. Successful results were obtained for automated face detection and for automated face recognition under robust conditions. Fully automated face detection and recognition was not realised because an eye detection system could not be implemented. Pose invariant face recognition was also successfully implemented under controlled conditions.

Table of Contents

Table of Figures

1.1 Face recognition systems

Automated face recognition is an interesting computer vision problem with many commercial and law enforcement applications. Mugshot matching, user verification and user access control, crowd surveillance, enhanced human computer interaction all become possible if an effective face recognition system can be implemented. While research into this area dates back to the 1960's, it is only very recently that acceptable results have been obtained. However, face recognition is still an area of active research since a completely successful approach or model has not been proposed to solve the face recognition problem.

The inadequacy of automated face recognition systems is especially apparent when compared to our own innate face recognition ability. We perform face recognition, an extremely complex visual task, almost instantaneously and our own recognition ability is far more robust than any computer's can hope to be. We can recognise a familiar individual under very adverse lighting conditions, from varying angles or view points. Scaling differences (a face being near or far away), different backgrounds do not affect our ability to recognise faces and we can even recognise individuals with just a fraction of their face visible or even after several years have past. Furthermore, we are able to recognize the faces of several thousand individuals whom we have met during our lifetime.

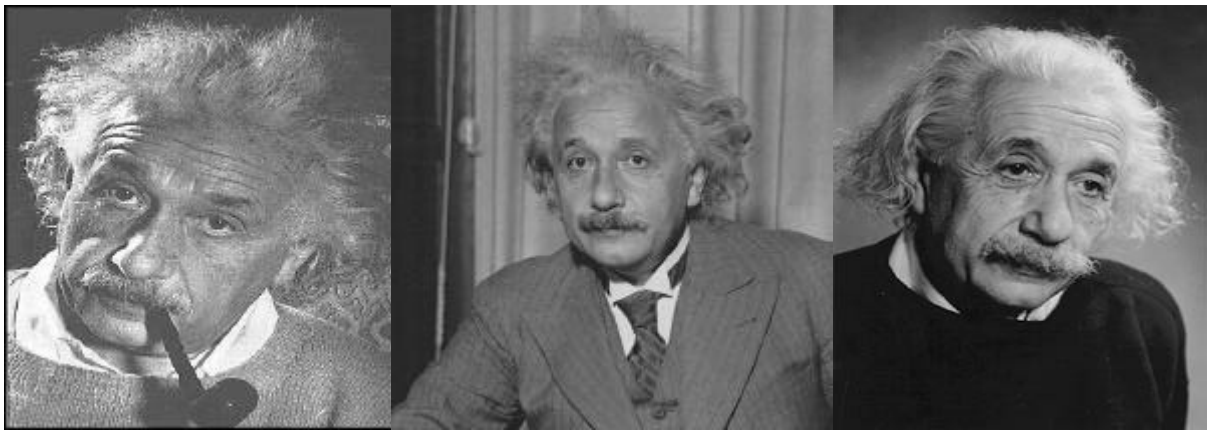


Figure 1.1 Three different photographs of Albert Einstein, easily recognised as the eminent physicist even though the photographs may vary greatly from our own recollection of him.

1.2 The difficulty of computer vision

Unfortunately it is not possible now, nor will it be possible in the foreseeable future to make a computing machine that actually 'understands' what is sees. The level of vision and understanding which is instinctive to us (humans) is still far out of the reach of our silicon creations.

Figure 1.2 *Polypedates eques* in Eastern Sinharaja

The ability to understand that the above image is not just a collection of pixels but is of a camouflaged frog on a log and to be able to identify exactly where the frog ends and log begins on the image is truly incredible. The fact that half of a primates cerebral cortex is dedicated to visual processing underlies the difficulty of this task (Zeki, 1993). This faculty is a result of millions of years of evolution and it would be naïve to think that we can enable computers to perform similar tasks.

But technically, why are computer vision problems so hard to solve? After all, while laudable results have been obtained in other artificial intelligence areas such as natural language processing, game theory, forecasting, control and even speech processing, computer vision seems to have lagged behind.

The main difficulty in vision problems is that almost all of them are ill-defined. For example, while segmenting (dividing) our image of *Polypedates eques* into areas of "frog",

"log" and "background" is intuitive and an innate ability to us (humans), it does not seem possible to find a definite problem specification of this task that a computer would understand.

Another factor is that even well defined computer vision problems may be ill-posed. Hadamard (1923) defined a problem as well posed if

(1) a solution exists,
(2) the solution is unique,
(3) the solution depends continuously on the initial data (stability property).

Many computer vision problems are ill-posed because information is lost in the transformation from the 3D world to a 2D image. Therefore, we cannot uniquely reconstruct the 3D representation from the 2D image and multiple solutions are often 'correct'.

The complexity of computer vision problems is exacerbated by the fact that we are dealing with huge chunks of data. A typical gray-scale image has 640x480 pixels, each with 8-bit (256) intensity values (gray-levels). Therefore, the size of the whole image is 640x480x8 bits = 2,457,600 bits. Any algorithm with high complexity would be extremely slow in computer vision and we must therefore make an effort to solve these problems using very simple processing techniques.

However, even with all these constraints it is possible to get useful results in computer vision by reducing a problem's generality. The computer vision application's problem domain can be restricted to a well-defined structured environment and assumptions could be made about lighting, types of object, etc.

Therefore, instead of trying to create a system that is suitable for all vision problems the computer vision and artificial intelligence communities have concentrated on obtaining useful results to real-world, limited applications in vision. Automated face recognition has thus become the holy grail of computer vision artificial intelligence. It is probably the most challenging and ambitious of the computer vision projects that are being studied and is not just a fascinating theoretical problem, but there is a real-world need for such a system.

1.3 Thesis Outline

This report details the research and development of frontal-view human face detection and recognition systems. Emphasis is even to the different computational and mathematical models that were modified by the researcher to satisfy these specific problems.

Chapter Two is an analysis of the face recognition problem as a whole. Many possible approaches to face detection and face recognition are discussed. The problem scope of the project is decided which leads to the design of the outline of a frontal-view human face detection and recognition model.

Chapter Three describes the development of the face detection model. Research was done into unsupervised learning and Kohonen Feature Maps. By combining this research with an efficient deformable template algorithm devised by the author and Rosenblatt's (1962) perceptron model, a fully automated face detection system was implemented. A manual face detection system was also implemented since a human operator's face detection ability surpasses that of a computing machine.

Chapter Four is based around face recognition itself. A human's own innate face recognition ability is discussed and a computational model for frontal-view face recognition based on Principal Component Analysis is developed. The Principal Component Analysis concept of 'reconstruction' is investigated and the face detection algorithm is modified to use verify face detection. The frontal view face recognition system is extended to a pose-invariant face recognition system.

Chapter Five discusses the image processing techniques which where necessary to provide lighting invariency and robustness. The development of the lighting invariency model is illustrated.

Chapter Six contains system tests and evaluation of all the proposed systems which were implemented by the researcher. These systems were implemented on Matlab 5.3. The following were tested

1) Fully automated face detection without verification
2) Fully automated face detection with verification
3) Manual face detection and automated face recognition
4) Fully automated face detection and recognition
5) Pose invariant face recognition

Chapter Seven discusses in detail further work that could be done to extend the capabilities and robustness of the implemented system. Also, two essential research areas in the field of face recognition are identified. The first is a system for transforming frontal view face images for pose invariant face recognition using virtual views. The second is a system using Support Vector Machines for pose invariant Face Detection.

Chapter Eight contains the author's conclusions on the work carried out on face detection and recognition.

2.1 Face detection.

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step.

There are two types of face detection problems:

1) Face detection in images and
2) Real-time face detection

2.1.1 Face detection in images

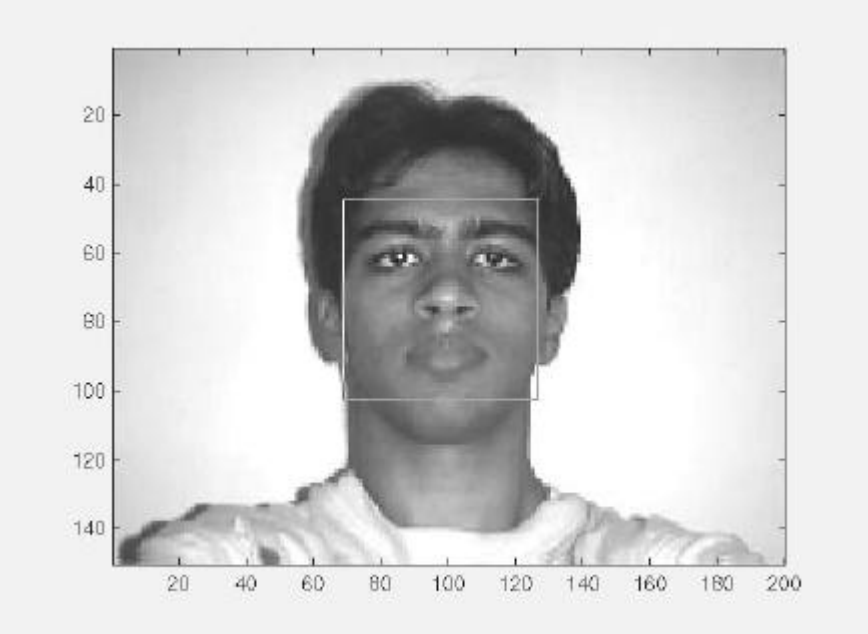


Figure 2.1 A successful face detection in an image with a frontal view of a human face

Most face detection systems attempt to extract a fraction of the whole face, thereby eliminating most of the background and other areas of an individual's head such as hair that are not necessary for the face recognition task. With static images, this is often done by running a '*window*' across the image. The face detection system then judges if a face is present inside the window (Brunelli and Poggio, 1993). Unfortunately, with static images there is a very large search space of possible locations of a face in an image. Faces may be

large or small and be positioned anywhere from the upper left to the lower right of the image.

Most face detection systems use an example based learning approach to decide whether or not a face is present in the *window* at that given instant (Sung and Poggio,1994 and Sung,1995). A neural network or some other classifier is trained using supervised learning with 'face' and 'non-face' examples, thereby enabling it to classify an image (*window* in face detection system) as a 'face' or 'non-face'.. Unfortunately, while it is relatively easy to find face examples, how would one find a representative sample of images which represent non-faces (Rowley et al., 1996)? Therefore, face detection systems using example based learning need thousands of 'face' and 'non-face' images for effective training. Rowley, Baluja, and Kanade (Rowley et al.,1996) used 1025 face images and 8000 non-face images (generated from 146,212,178 sub-images) for their training set!

There is another technique for determining whether there is a face inside the face detection system's *window* - using Template Matching. The difference between a fixed target pattern (face) and the *window* is computed and thresholded. If the window contains a pattern which is close to the target pattern(face) then the window is judged as containing a face. An implementation of template matching called Correlation Templates uses a whole bank of fixed sized templates to detect facial features in an image (Bichsel, 1991 & Brunelli and Poggio, 1993). By using several templates of different (fixed) sizes, faces of different scales (sizes) are detected. The other implementation of template matching is using a deformable template (Yuille, 1992). Instead of using several fixed size templates, we use a deformable template (which is non-rigid) and there by change the size of the template hoping to detect a face in an image.

A face detection scheme that is related to template matching is image invariants. Here the fact that the local ordinal structure of brightness distribution of a face remains largely unchanged under different illumination conditions (Sinha, 1994) is used to construct a spatial template of the face which closely corresponds to facial features. In other words, the average grey-scale intensities in human faces are used as a basis for face detection. For example, almost always an individuals eye region is darker than his forehead or nose. Therefore an image will match the template if it satisfies the 'darker than' and 'brighter than' relationships (Sung and Poggio, 1994).

2.1.2 Real-time face detection

Real-time face detection involves detection of a face from a series of frames from a video-capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving. We walk around, blink, fidget, wave our hands about, etc.

Figure 2.2: Frame 1 from camera

Figure 2.3: Frame 2 from camera

Figure 2.4: Spatio-Temporally filtered image

Since in real-time face detection, the system is presented with a series of frames in which to detect a face, by using spatio-temperal filtering (finding the difference between subsequent frames), the area of the frame that has changed can be identified and the individual detected (Wang and Adelson, 1994 and Adelson and Bergen 1986).

Further more as seen in Figure 2.4, exact face locations can be easily identified by using a few simple rules, such as,

1) the head is the small blob above a larger blob -the body
2) head motion must be reasonably slow and contiguous -heads won't jump around erratically (Turk and Pentland 1991a, 1991b).

Real-time face detection has therefore become a relatively simple problem and is possible even in unstructured and uncontrolled environments using these very simple image processing techniques and reasoning rules.

2.2 Face recognition

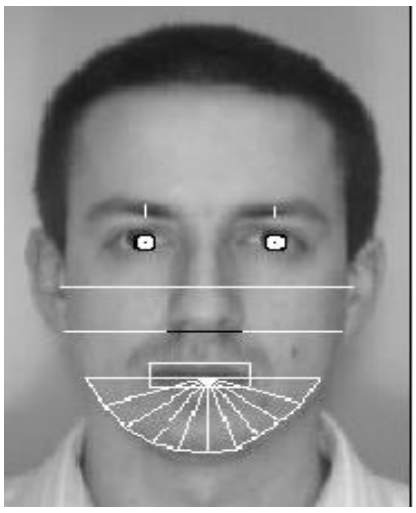
Over the last few decades many techniques have been proposed for face recognition. Many of the techniques proposed during the early stages of computer vision cannot be considered successful, but almost all of the recent approaches to the face recognition problem have been creditable. According to the research by Brunelli and Poggio (1993) all approaches to human face recognition can be divided into two strategies:

(1) geometrical features and

(2) template matching.

2.2.1 Face recognition using geometrical features

This technique involves computation of a set of geometrical features such as nose width and length, mouth position and chin shape, etc. from the picture of the face we want to recognize. This set of features is then matched with the features of known individuals. A suitable metric such as Euclidean distance (finding the closest vector) can be used to find the closest match. Most pioneering work in face recognition was done using geometric features (Kanade, 1973), although Craw et al. (1987) did relatively recent work in this area.



Geometrical features (white) which could be used for face recognition

Figure 2.5

The advantage of using geometrical features as a basis for face recognition is that recognition is possible even at very low resolutions and with noisy images (images with many disorderly pixel intensities). Although the face cannot be viewed in detail its overall geometrical configuration can be extracted for face recognition. The technique's main

disadvantage is that automated extraction of the facial geometrical features is very hard. Automated geometrical feature extraction based recognition is also very sensitive to the scaling and rotation of a face in the image plane (Brunelli and Poggio, 1993). This is apparent when we examine Kanade's(1973) results where he reported a recognition rate of between 45-75 % with a database of only 20 people. However if these features are extracted manually as in Goldstein et al. (1971), and Kaya and Kobayashi (1972) satisfactory results may be obtained.

2.2.2 Face recognition using template matching

This is similar the template matching technique used in face detection, except here we are not trying to classify an image as a 'face' or 'non-face' but are trying to recognize a face.

Whole face, eyes, nose and mouth regions which could be used in a template matching strategy

Figure 2.6

The basis of the template matching strategy is to extract whole facial regions (matrix of pixels) and compare these with the stored images of known individuals. Once again Euclidean distance can be used to find the closest match. The simple technique of comparing grey-scale intensity values for face recognition was used by Baron (1981). However there are far more sophisticated methods of template matching for face recognition. These involve extensive pre-processing and transformation of the extracted grey-level intensity values. For example, Turk and Pentland (1991a) used Principal Component Analysis, sometimes known as the eigenfaces approach, to pre-process the gray-levels and Wiskott et al. (1997) used Elastic Graphs encoded using Gabor filters to pre-process the extracted regions.

An investigation of geometrical features versus template matching for face recognition by Brunelli and Poggio (1993) came to the conclusion that although a feature based strategy may offer higher recognition speed and smaller memory requirements, template based techniques offer superior recognition accuracy.

2.3 Problem scope and system specifications

The following problem scope for this project was arrived at after reviewing the literature on face detection and face recognition, and determining possible real-world situations where such systems would be of use. The following system(s) requirements were identified

- A system to detect frontal view faces in static images
- A system to recognize a given frontal view face
- Only expressionless, frontal view faces will be presented to the face detection and face recognition systems
- All implemented systems must display a high degree of lighting invariency.
- All systems must posses near real-time performance.
- Both fully automated and manual face detection must be supported
- Frontal view face recognition will be realised using only a single known image from each individual.
- Automated face detection and recognition systems should be combined into a fully automated face detection and recognition system. The face recognition sub-system must display a slight degree of invariency to scaling and rotation errors in the segmented image extracted by the face detection sub-system.
- The frontal view face recognition system should be extended to a pose invariant face recognition system.

Unfortunately although we may specify constricting conditions to our problem domain, it may not be possible to strictly adhere to these conditions when implementing a system in the real-world.

2.4 Brief outline of the implemented system

Fully automated face detection of frontal view faces is implemented using a deformable template algorithm relying on the image invariants of human faces. This was chosen because a similar neural-network based face detection model would have needed far too much training data to be implemented and would have used a great deal of computing time. The main difficulties in implementing a deformable template based technique were the creation of the bright and dark intensity sensitive templates and designing an efficient implementation of the detection algorithm.

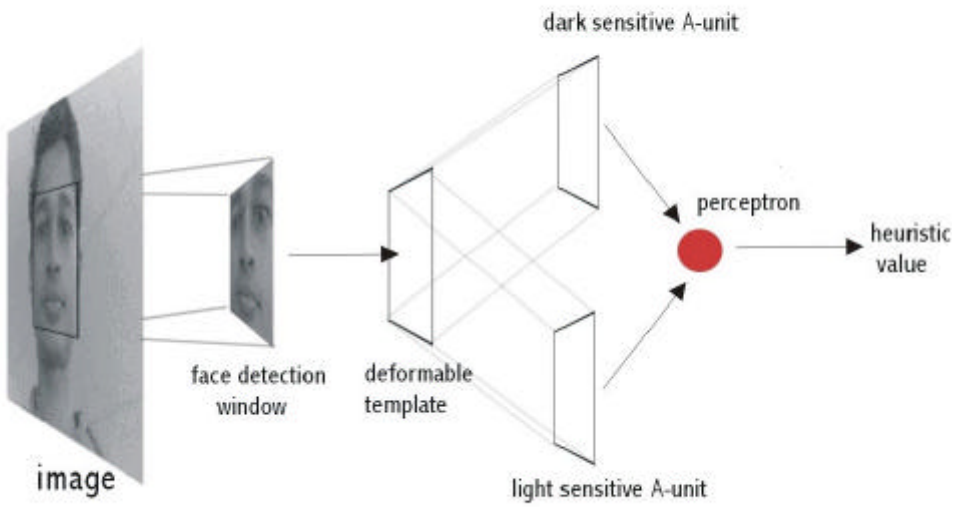


Figure 2.7: Implemented fully automated frontal view face detection model

A manual face detection system was realised by measuring the facial proportions of the average face, calculated from 30 test subjects. To detect a face, a human operator would identify the locations of the subject's eyes in an image and using the proportions of the average face, the system would segment an area from the image.

A template matching based technique was implemented for face recognition. This was because of its increased recognition accuracy when compared to geometrical features based techniques and the fact that an automated geometrical features based technique would have required complex feature detection pre-processing.

Of the many possible template matching techniques, Principal Component Analysis was chosen because it has proved to be a highly robust in pattern recognition tasks and because it is relatively simple to implement. The author would also liked to have implemented a technique based on Elastic Graphs but could not find sufficient literature about the model to implement such a system during the limited time available for this project.

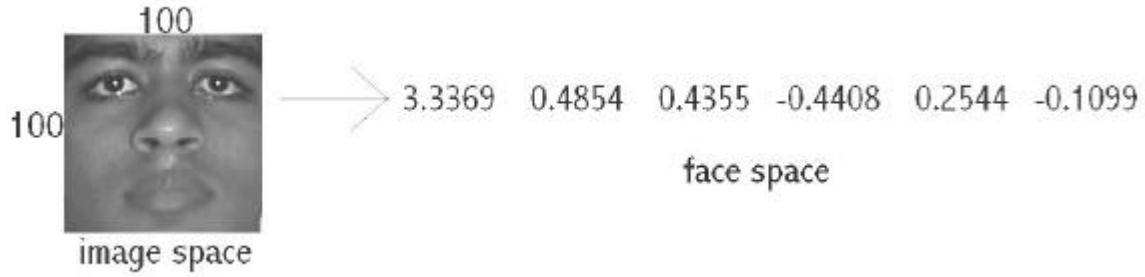


Figure 2.8: Principal Component Analysis transform from 'image space' to 'face space'.

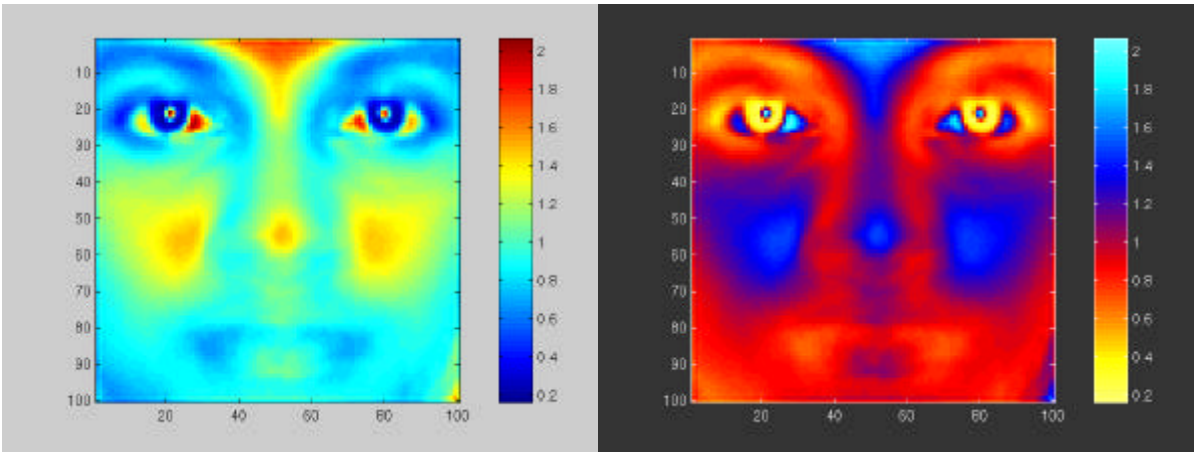
Using Principal Component Analysis, the segmented frontal view face image is transformed from what is sometimes called 'image space' to 'face space'. All faces in the face database are transformed into face space. Then face recognition is achieved by transforming any given test image into face space and comparing it with the training set vectors. The closest matching training set vector should belong to the same individual as the test image.

Principal Component Analysis is of special interest because the transformation to face space is based on the variation of human faces (in the training set). The values of the 'face space' vector correspond to the amount certain 'variations' are present in the test image.

3.1 Introduction

While some may regard face detection as simple pre-processing for the face recognition system, it is by far the most important process in a face detection and recognition system. However face recognition is not the only possible application of a fully automated face detection system. There are applications in automated colour film development where information about the exact face location is useful for determining exposure and colour levels during film development. The are even uses in face tracking for automated camera control in the film and television news industries.

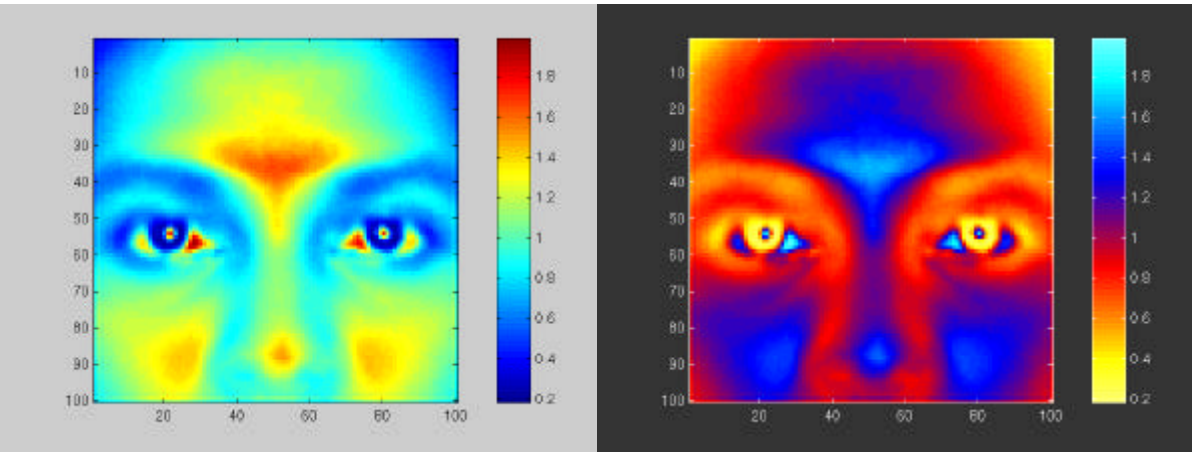
In this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the grey-scale intensity distribution of an average human face. The following 'average human face' was constructed from a sample of 30 frontal view human faces, of which 12 were from females and 18 from males. A suitably scaled colormap has been used to highlight grey-scale intensity differences.



scaled colormap scaled colormap (negative)

Figure 3.1: Average human face in grey-scale

The grey-scale differences, which are invariant across all the sample faces are strikingly apparent. The eye-eyebrow area seem to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) grey-levels. After a great deal of experimentation, the researcher found that the following areas of the human face were suitable for a face detection system based on image invariants and a deformable template.



scaled colormap scaled colormap (negative)

Figure 3.2 Area chosen for face detection (indicated on average human face in gray scale)

The above facial area performs well as a basis for a face template, probably because of the clear divisions of the bright intensity invariant area by the dark intensity invariant regions. Once this pixel area is located by the face detection system, any particular area required can be segmented based on the proportions of the average human face

After studying the above images it was subjectively decided by the author to use the following as a basis for dark intensity sensitive and bright intensity sensitive templates. Once these are located in a subject's face, a pixel area 33.3% (of the width of the square window) below this area will be segmented.

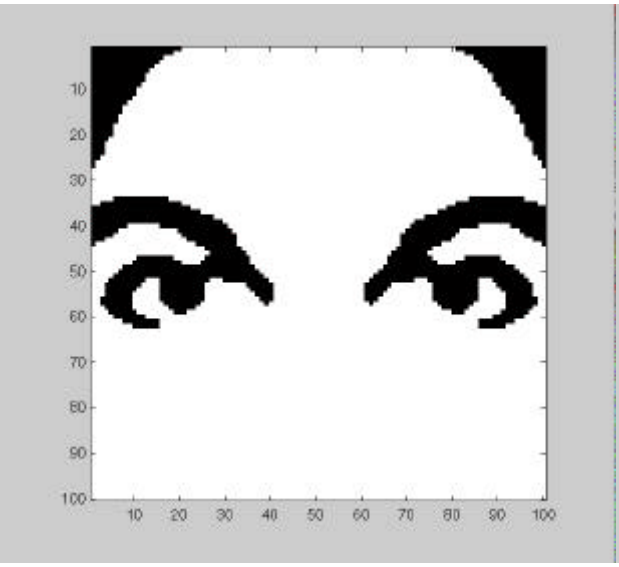


Figure 3.3: Basis for dark intensity invariant sensitive template

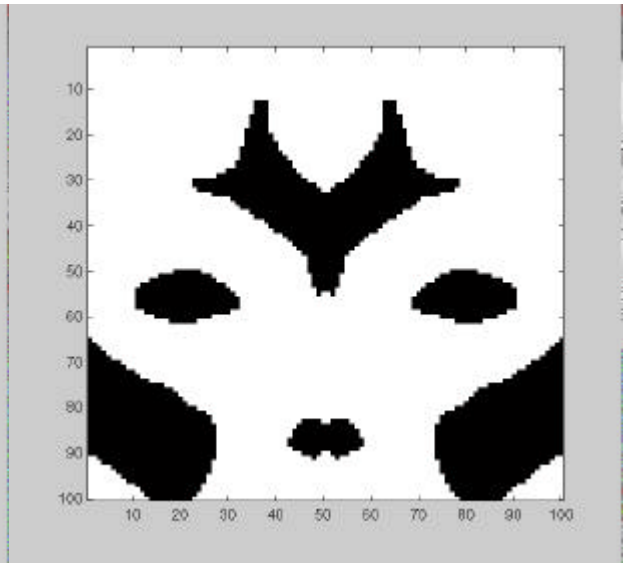


Figure 3.4: Basis for a bright intensity invariant sensitive template.

Note the slight differences which were made to the bright intensity invariant sensitive template (compare Figures 3.4 and 3.2) which were needed because of the pre-processing done by the system to overcome irregular lighting (chapter six).

Now that a suitable dark and bright intensity invariant templates have been decided on, it is necessary to find a way of using these to make 2 A-units for a perceptron, i.e. a computational model is needed to assign neurons to the distributions displayed in Figures 3.3 and 3.4.

3.2 Kohonen Feature Maps

Kohonen (1995) in his famous book *Self Organising Maps* stated that SOMs (i.e. Self Organising Maps) are not intended for pattern recognition but for clustering, visualisation, and abstraction. Out of the many different models described in *Self Organising Maps* are Kohonen Feature Maps (KFMs), also known as Kohonen self-organizing networks or topology-preserving maps, were found to be amazingly suitable for the problem of creating A-units for a perceptron.

Self-Organizing Maps are competitive networks that provide a "topological" mapping from the input space to the clusters (Kohonen, 1995). These were inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that

spatial or other relations among stimuli correspond to spatial relations among the neurons. So in SOMs, patterns in the input space near to each other will be mapped to output units near to each other. SOMs are one of the many neuro-computational models that use unsupervised learning. Unlike a typical Neural Network (which uses supervised learning), the training data for SOMs does not consist of input and desired output pairs. Training data consists of a set of input patterns and the topology of the SOMs adapts conforming to these patterns.

According to Krose and van der Smagt (1996) the main applications for SOMs are for,

- clustering: the input data may be grouped in 'clusters' and the data processing system has to find these inherent clusters in the input data. The output of the system should give the cluster label of the input pattern (discrete output);
- vector quantization: descretising continuous space. The inputs are n-dimensional vectors and the output is a discrete representation of the input space. The system has to find optimal discretisation of the input space;
- dimensionality reduction: the input data must be grouped in a subspace which has lower dimensionality than the original dimensionality of the input data. The system has to learn an optimal mapping, such that most of the variance in the input data is preserved in the output data;
- feature extraction: the system has to extract features from the input signal. SOMs are able to extract features without external supervision (no desired output specified). Feature extraction is similar to dimensionality reduction described earlier.

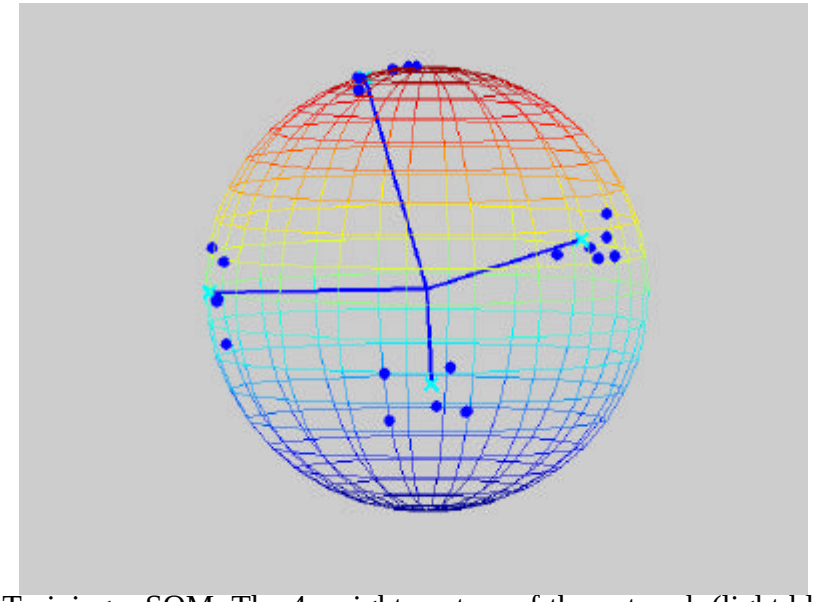


Figure 3.5 : Training a SOM. The 4 weight vectors of the network (light blue cross), rotate towards the centres of the four clusters in the input pattern

3.2.1 Developing a suitable Kohonen Feature Map neuro-computational model

In Kohonen Feature Maps (KFMs) the topological properties of the input pattern are directly reflected in the weights of the neurons in the output units (Jang et al., 1997). A similarity measure or mapping is selected between the input pattern and network units and a winning unit is considered to be the network unit with the closest distance to the mapped vector. Then KFMs update the winning unit's weight as well as then weights of the units within a predefined neighbourhood around the winning unit. These weights are usually updated by bringing them closer to the input vector. The following figures should help the reader better understand training KFMs.

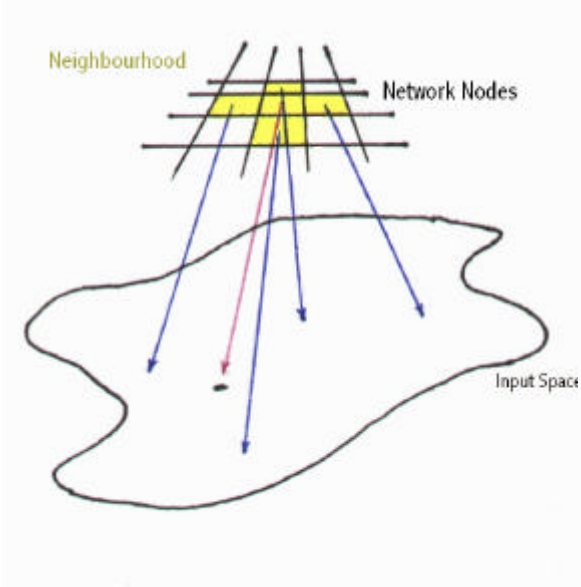


Figure 3.6 Training a KFM. Winner chosen as node with weight closest to the input vector

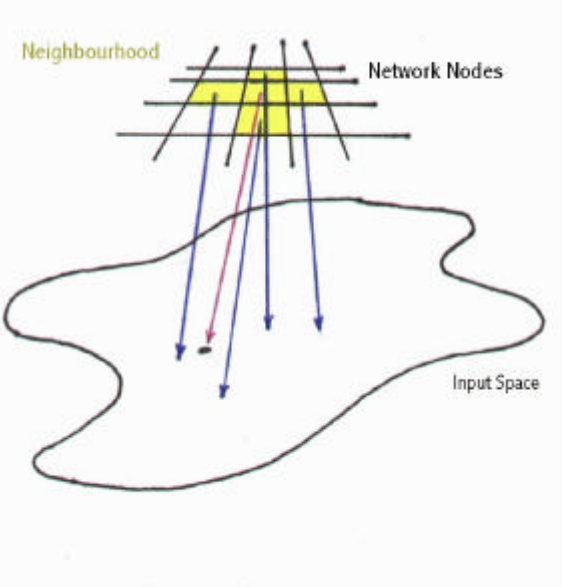


Figure 3.7 The weights of the winner and the weights of nodes in the winner's predefined neighbourhood are updated.

In the above figures, the KFM is simulated with an input vector (black dot). A 'winner' (node with red arrow) is chosen as the node whose weight has the closest Euclidean distance to the input vector. If the winning node's weight is w_c and the input vector is x,

$$\| x - w_c \| = \min \| x - w_i \| \quad \forall i.$$

Then, the weight of the winner (red arrow) and the weights (blue arrows) of all the nodes in the winner's predefined neighbourhood (yellow area) are updated. The degree that the weights are updated is proportional to their distance from x and η, the networks learning rate at that instance. So the weight update function is,

$$\Delta w_i = \eta\ (x\text{-}w_i), \text{ where } i \in \text{ winners neighbourhood}$$

Since it would be computationally expensive to calculate each winner's neighbourhood each time an input vector is submitted, neighbourhood function around the winning unit (c) is used to update w_i. Therefore using a Gaussian (Ω) neighbourhood function around c, the weight update function becomes,

$$\Delta w_i = \eta\ \Omega_c\ (x\text{-}w_i), \text{ where } i \in \text{ winners neighbourhood}$$

In which the Gaussian (Ω_c) is

$$\Omega_c(i) = \exp\ \left(\frac{-\| p_i - p_c \|^2}{2\sigma^2} \right)$$

where, p_i and p_c are the positions of the winning node (c) and node (i) on the network grid, while σ reflects the predefined size of the neighbourhood. The Gaussian is a suitable neighbourhood function since because of its unique shape, it can be made to decrease sharply where we want to specify our neighbour size.

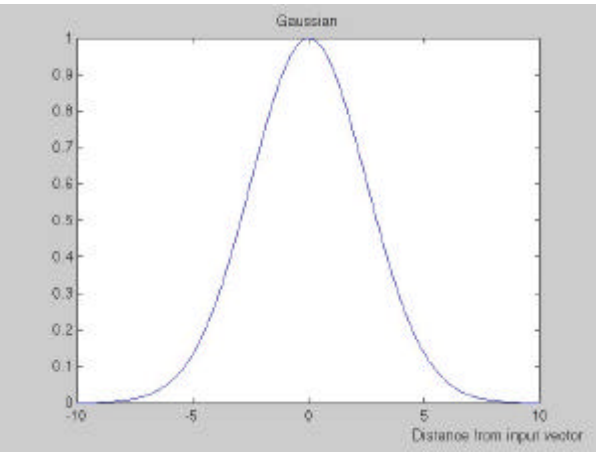


Figure 3.8 Guassian neighbourhood function. σ =0.4

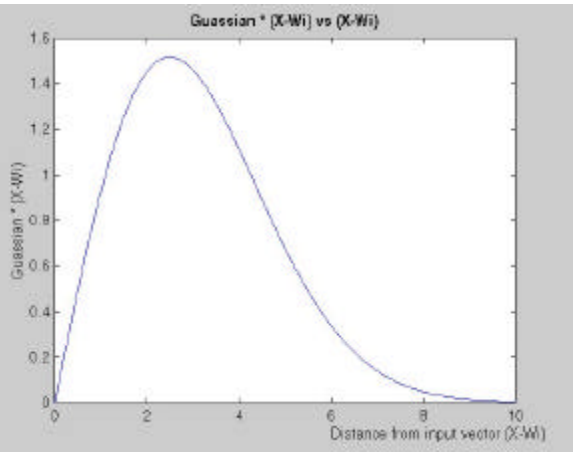


Figure 3.9 $\Omega_c\ (x\text{-}w_i)$ vs $(x\text{-}w_i)$

Interestingly the weight update can never cause the weight to go past the stimulating input vector because as shown in Figure 3.9, the weight update function contains,

Guassian x (distance from input vector)

The weight update function therefore decreases when it nears zero (note that the learning rate η is always less than zero), thereby not causing the node's weight to go pass the stimulating input vector.

Generally to achieve better convergence when training KFMs, and in fact almost all other SOMs, network models initially start with large neighbourhoods (σ) and gradually decrease their size as training progresses. The learning rate (η) is also gradually reduces when training the KFM or SOM. If these two principles are not followed, the network may get stuck in a local optima and not converge to resemble the input pattern's topology.

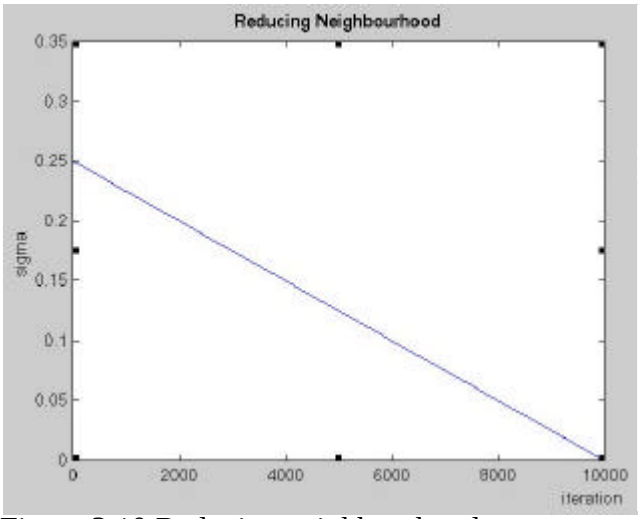


Figure 3.10 Reducing neighbourhood as training progresses up to 10000 iterations

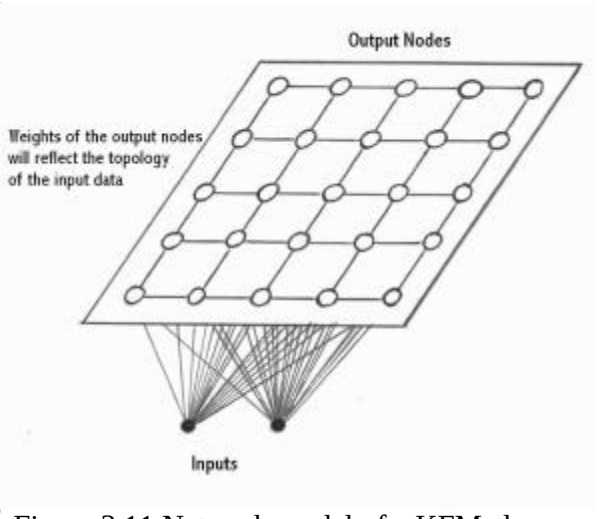


Figure 3.11 Network model of a KFM also known as a Kohonen self-organizing network

3.2.2 Grid space neighbourhood

The following network was created using the standard KFM algorithm. The neighbourhood of the winner was determined from the grid space (neighbouring nodes in network). However, the author decided to stray from the norm and use a uniform initial weight distribution as it was found to be better than a random distribution since the former spanned the whole input space and thus better suited the problem at hand.

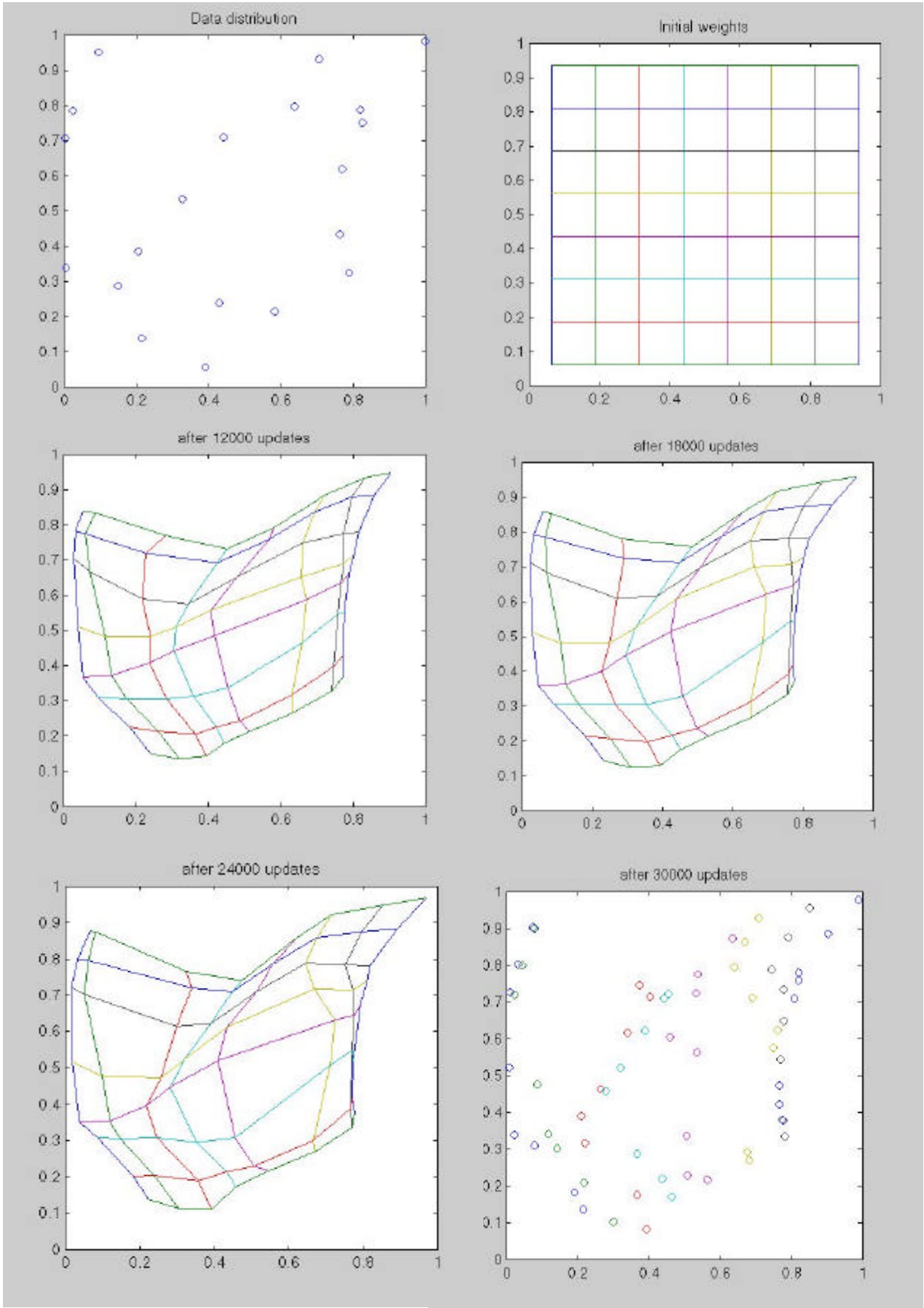


Figure 3.12 Training a Kohonen Feature Map

Note how the weight topology changes (figure 3.12) as the Kohonen Feature Map adapts to the stimulating input pattern. A random input pattern was generated to test the network.

When one examines figure 3.12 it is apparent that the KFM does not map the input distribution topology to an accuracy, which would enable us to create a face template. The author experimented with different neighbourhood and learning rate parameters to no avail, the network weights would not perfectly map the input pattern. Clearly the KFM algorithm would have to be modified. Although there are no network models specified in the literature that optimises KFMs, there are several concepts that are used for SOMs that may prove useful improve the performance the KFM model.

3.2.3 Input space neighbourhood

In the KFM algorithm that was used thus far, the neighbourhood was determined from the nodes around the winning node. In other words a network grid space neighbourhood was used. It is also possible to use an input space neighbourhood. Here all nodes with weights lying inside a predefined neighbourhood around the *input vector* are updated. So the weight update function for node *i* becomes,

$$\Delta w_i = \eta \, \Omega_x (x - w_i), \; \forall \; i$$

where x is the stimulating input vector. Notice that the Gaussian function is centred around input vector. This creates a neighbourhood around x and all the weights in the network are updated.

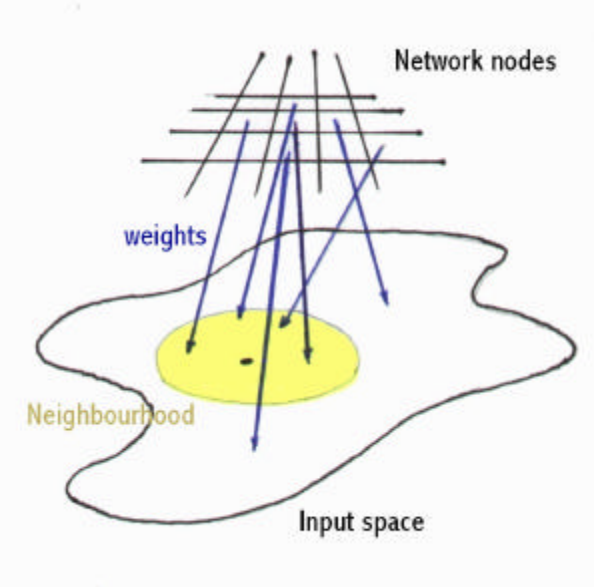


Figure 3.13 Training a KFM using an input space neighbourhood

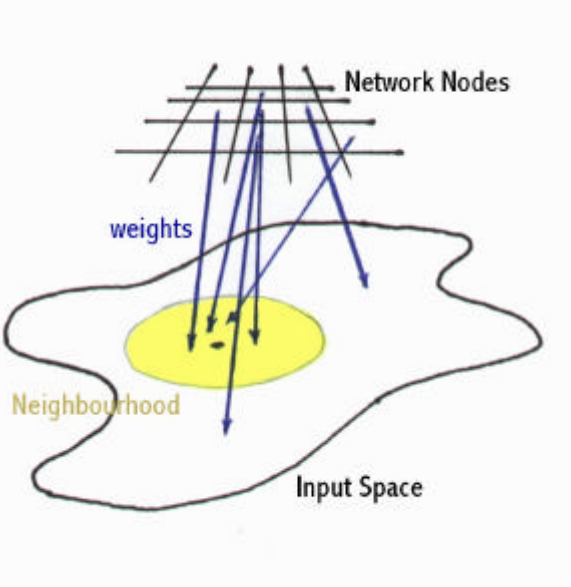


Figure 3.14 All weights in the input vectors predefined neighbourhood are updated.

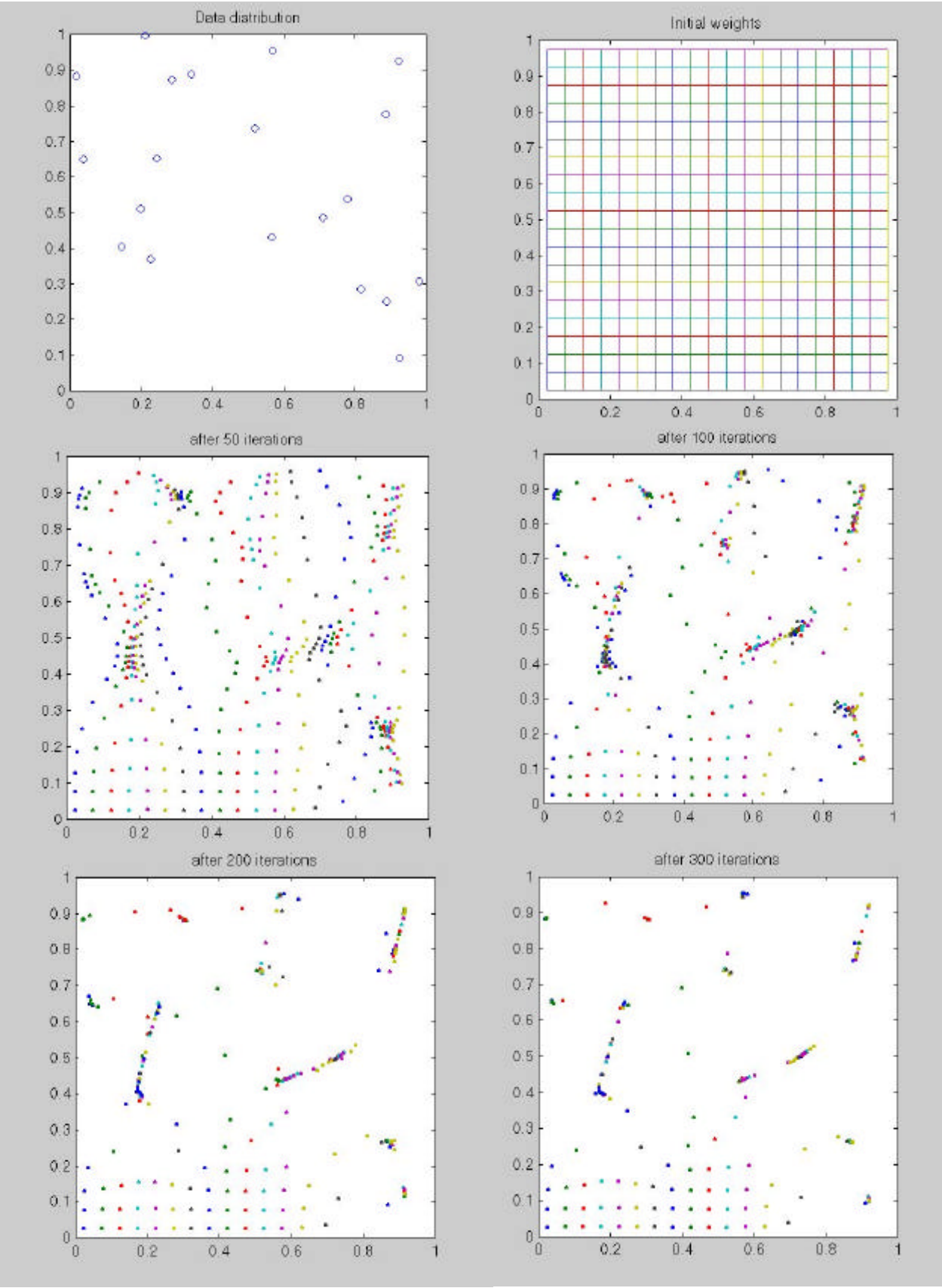


Figure 3.15 Training a Kohonen Feature Map with an input space neighbourhood

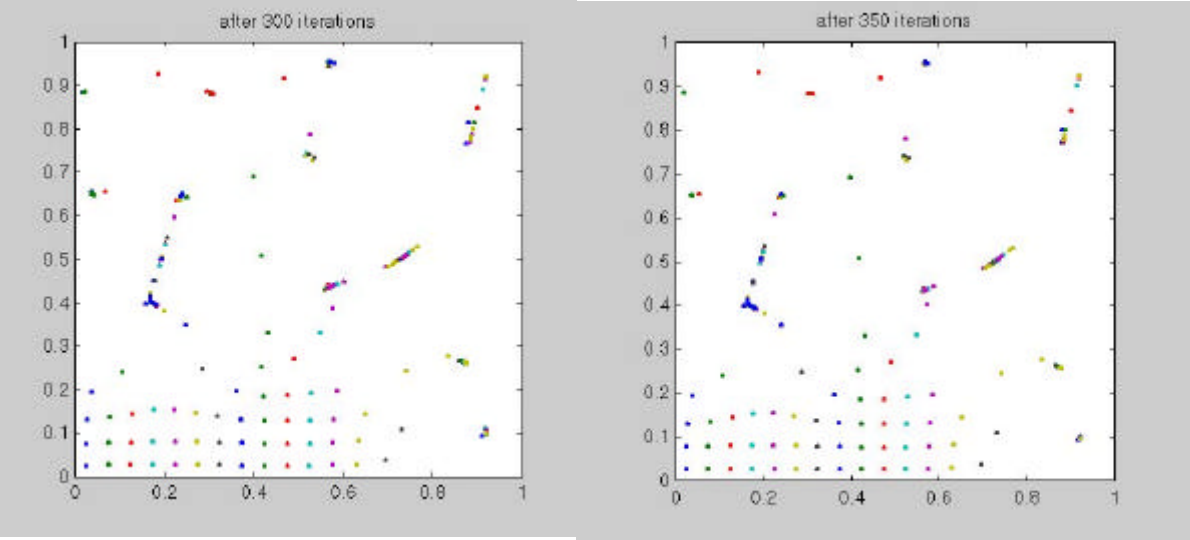


Figure 3.16 Training a Kohonen Feature Map with an input space neighbourhood(contd)

Using an input space neighbourhood, many of the network node's weights have very accurately mapped to the input pattern. Yet an interesting feature can be noticed in the above figures. As the learning rate and neighbourhood sizes decreased during training some weights had stopped being stimulated. This is especially apparent in the large area on the lower left-hand side of the weight space. This is clearly unacceptable. The author reminds the reader that the intention is to create the A-units of a perceptron for a deformable template algorithm for face detection.

3.2.4 Sensitivity

The obvious solution to this problem is to increase the sensitivity of neurons that are not being stimulated. The author's implementation of node sensitivity is as follows. Using our standard notation, the weight update formula becomes,

$$\Delta w_i = \eta \, \Omega_c \, (x - w_i) \, \tau \, s_i$$

where τ determines the effect of the sensitivity s_i. At each iteration s_i decremented by Δw_I,

$$s_i = s_i - \Delta w_i$$

and the affect of time (T) on sensitivity is incremented at each iteration to s_i

$$s_i = s_i + T$$

A typical value for T would be between 0.2 and 0.3. The following figure depicts the training of a Kohonen Feature map with an input space neighbourhood with node sensitivity.

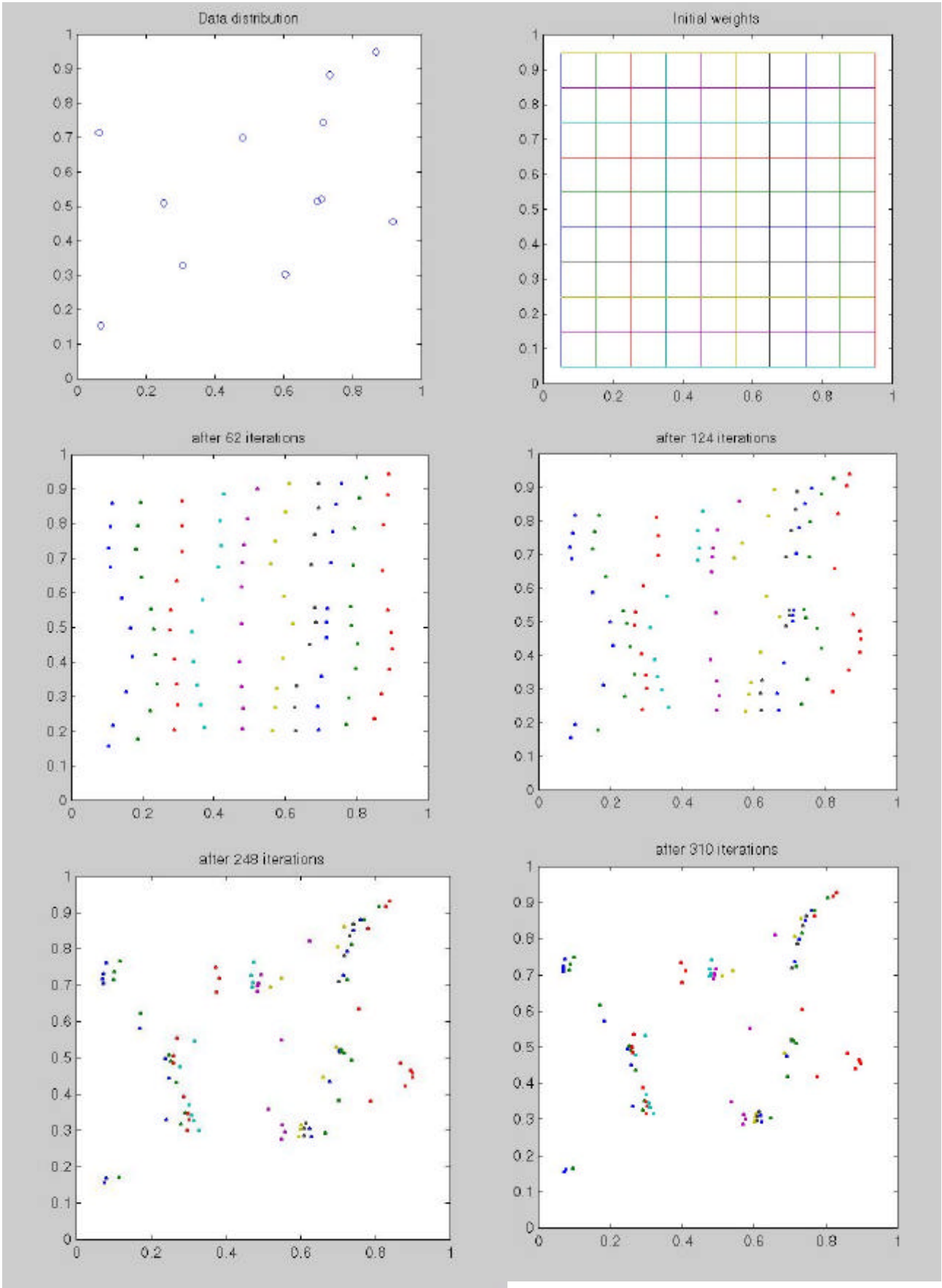


Figure 3.17 Training an input space Kohonen Feature Map with node sensitivity

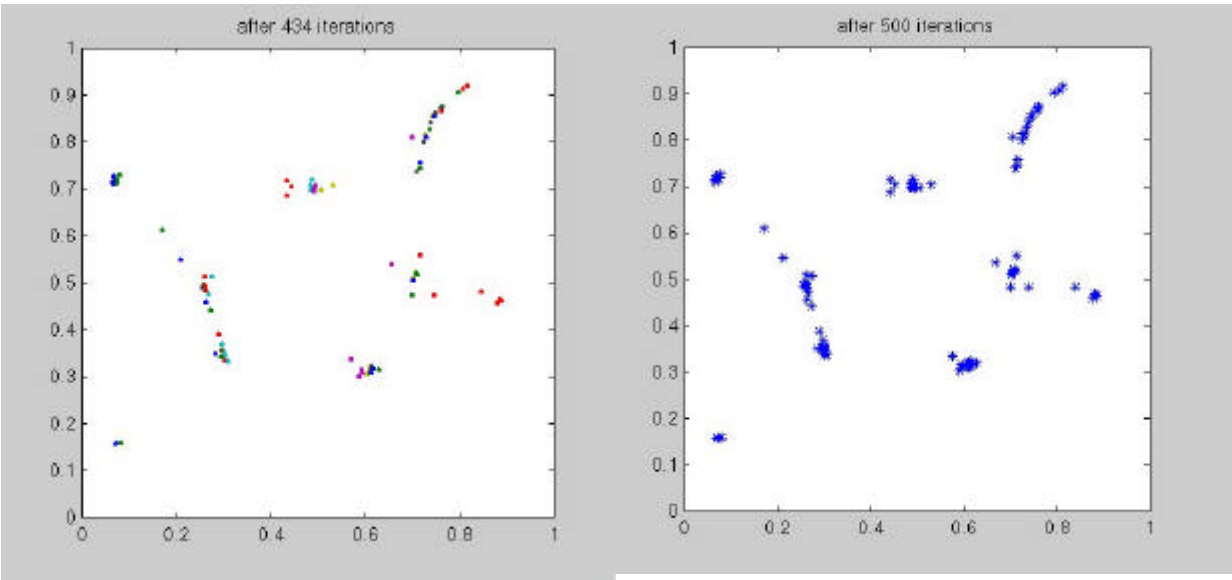


Figure 3.18 Training an input space Kohonen Feature Map with node sensitivity(contd)

Having closely mapped the KFM to the input pattern, a suitable network model has been finally found to create the deformable template. In fact the following two figure show the actual KFM maps which where generated for the face detection system. The reader is encouraged to compared these templates with the distributions that are being to mapped (Figures 3.3 and 3.4).

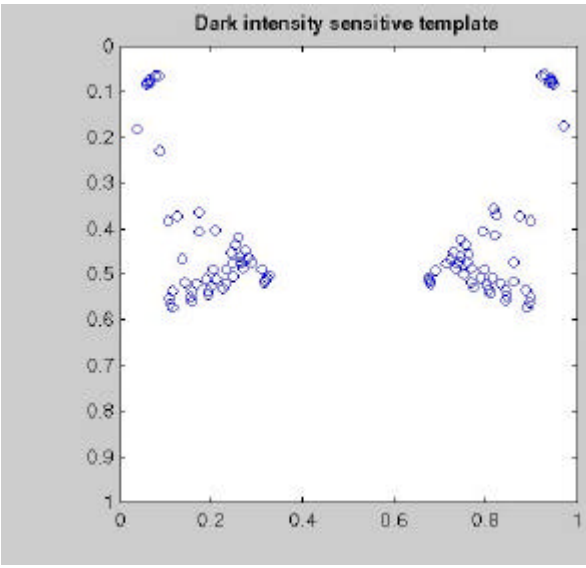


Figure 3.19 Dark intensity sensitive template

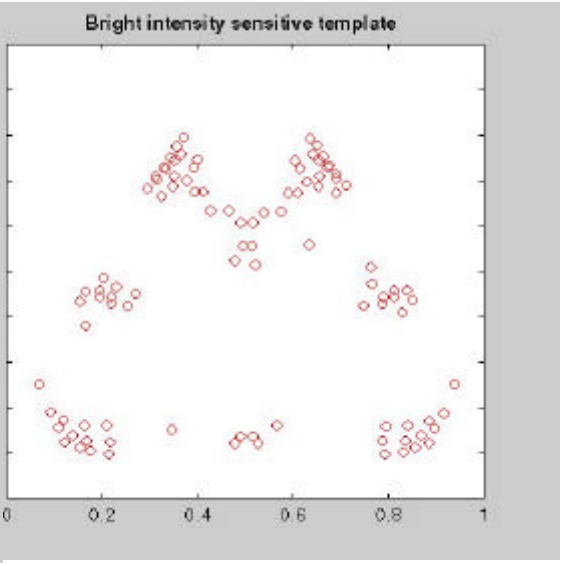


Figure 3.20 Bright intensity sensitive template

3.3 Deformable template algorithm

The dark and bright intensity invariant templates which have been created can be implemented in a deformable template algorithm. For example, if a 100x100 template is needed to check whether a certain 100x100 pixel area contains a face, the weight vectors in the dark and bright templates have to be simply multiplied by 100. Now the pixels in the areas that are indicated by the templates can be sampled to see if they match the dark and bright patterns

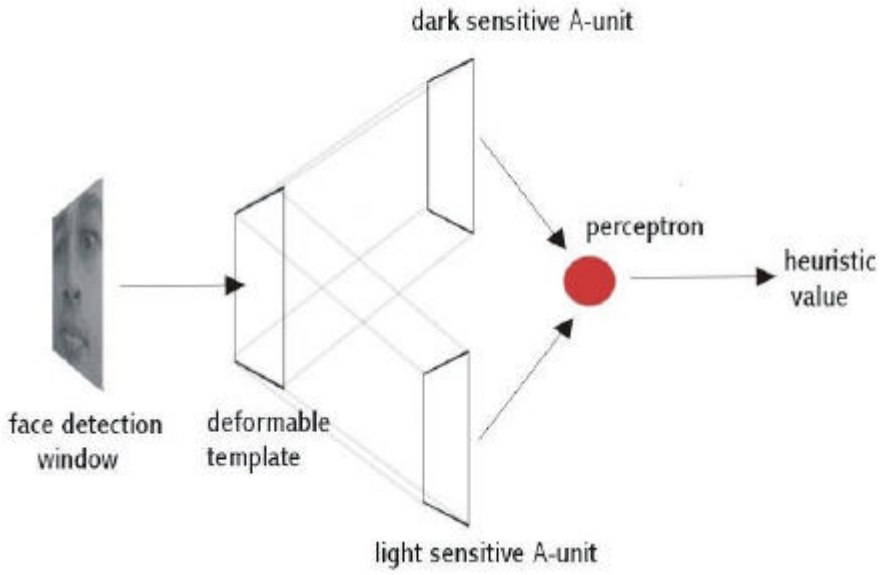


Figure 3.21 Deformable Template

However since individually sampling pixels for the templates is expensive, an efficient, simple implementation of a deformable template is needed. Remember that computer vision problems are by nature computationally expensive. Therefore, very simple algorithms to implement the system must be found.

Since computers are efficient at array indexing, it occurred to the author that this would be an ideal way to implement the deformable template. Once the deformable template has been expanded to match a particular window (for example, 100x100), the weight vectors of each template are converted to array indexes. Then the elements in the image (i.e. 100x100 pixel area), which can be regarded as a 2-dimensional array are extracted.

image(indexes) = (pixel intensities at indexed positions)

Therefore, to keep the face detection algorithm simple, extracted pixel intensities could be merely added. As a result, if the total added pixel intensities from the bright intensity invariant template is high and the total added intensities from the dark intensity invariant template is low, a high heuristic value will be output by the perceptron.

3.4 Development of the face search algorithm

An exhaustive search of an image for a face is clearly impossible since there are almost an infinite number of possible places where a face may be. A face may be anywhere from the upper left to the lower right corner of the image, scaled to almost fit the whole image or far away in the distance. An efficient search algorithm is essential for face detection in near real-time.

One possible way of reducing the search space in static images is to use the fact that frontal view faces are symmetrical (Saber and Tekalp, 1996). If this (vertical) line of symmetry can be found, the deformable template can be used only along this line of symmetry. After experimentation, it was discovered that trying to find a high correlation coefficient of pixel areas on either side of the line of symmetry was a suitable model to reduce the search space.

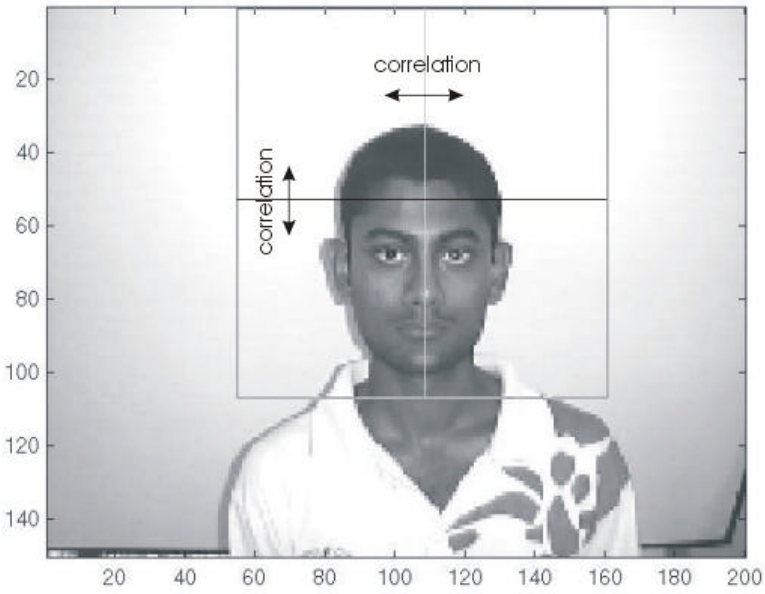


Figure 3.22 Checking correlation of vertical and horizontal pixel areas

Using correlation as a statistic of similarity is preferable to comparing the two pixel areas on the left and right of the (tested) line of symmetry. This is because the correlation coefficient measures the strength of the relation between the two pixel areas (Frank and Althoen, 1994) and is therefore less sensitive to lighting variations between the left and right halves. The element (i,j)of the correlation coefficient matrix is related to the corresponding element of the covariance matrix (C) by,

$$rR(i,j) = \frac{C(i,j))}{\sqrt{C(i,i)\,C(j,j)}}$$

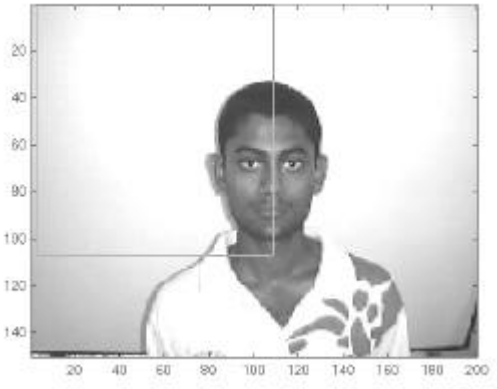
Where the covariance matrix C is.

$$C_{XY} = \frac{1}{N}\sum^{N}(x_i - \bar{x})(y_i - \bar{y})$$

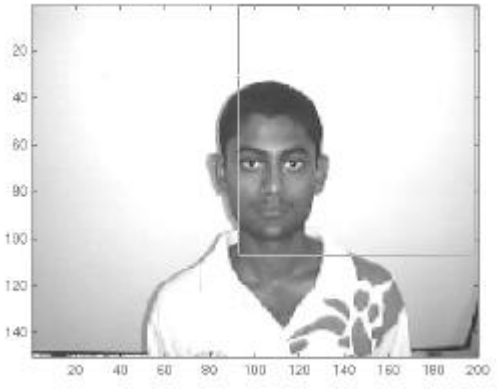
X and Y are the two pixel regions on either side of the (tested) line of symmetry.

To find the best line of symmetry (highest correlation coefficient) a window is run from left to right on the upper edge of the image and the window position, which resulted in the highest correlation coefficient, was regarded as containing the best line of symmetry. A threshold value could have been used here to reduce the search space. If a correlation coefficient value higher than a certain threshold value was obtained then the search could have been stopped at that particular window position. However, the researcher did not want to use a threshold value since there is a wide range in the best correlation coefficient value obtained from different frontal view face images.

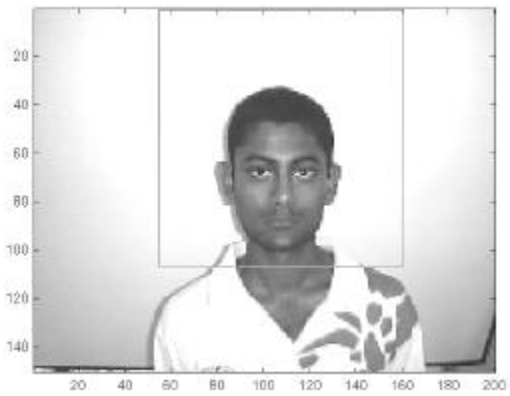
After experimenting with this model it was found that there may be several areas in an image which display a high degree of symmetry around a vertical axis. Often a plain (even) background would result in a undesirable line of symmetry. Therefore it was decided to look for areas (window positions) with high correlation around a vertical axis and low correlation around a horizontal axis (figure 3.22).



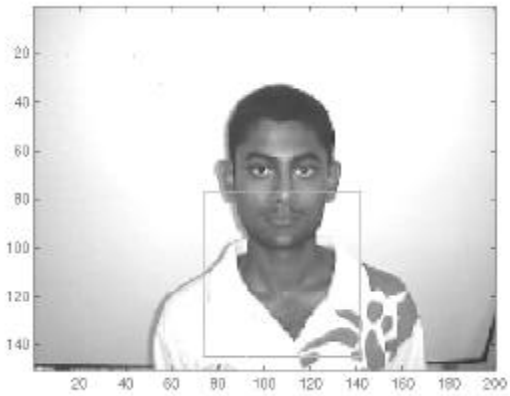
Start of search for symmetry



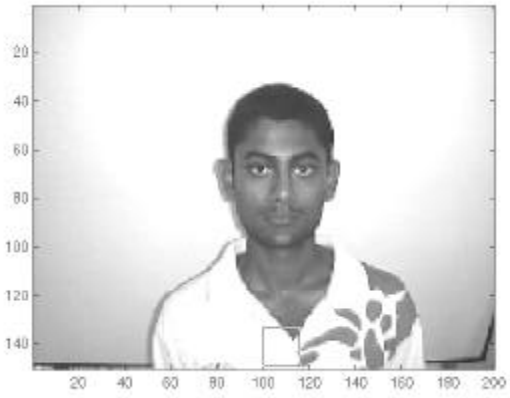
End of search for symmetry



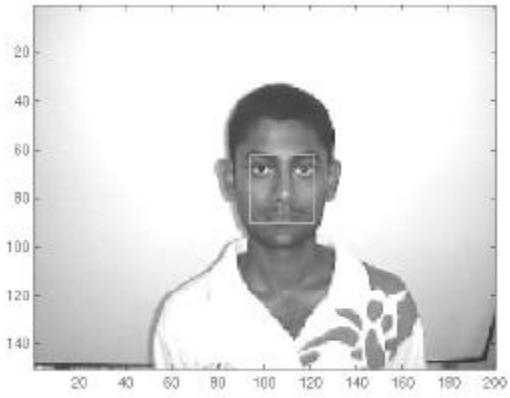
Successful symmetry search



Face detection in progress



End of face detection process



Successful face detection

Figure 3.23 Fully automated face detection procedure

When searching for a face's line of symmetry the particular pixel window that is considered has to be moved from the upper left to upper right of the image. The optimal distance that the window is moved has to be determined. There is a speed versus accuracy trade off here, with small 'jumps' of the window finding a the exact line of symmetry while large 'jumps'

being much faster in finding an approximate solution since less pixel windows are being evaluated. A perfect face detection algorithm which takes an hour to finish processing would have no real-world application. These decisions should be made depending on the platform the face detection system is implemented.

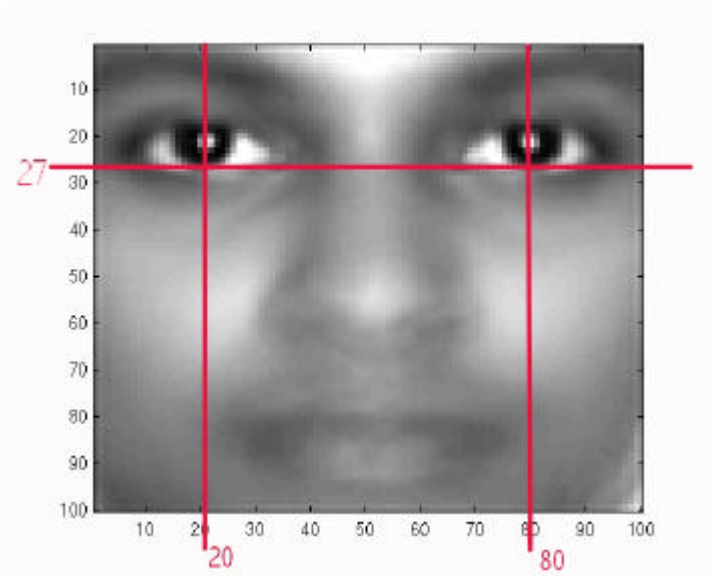
Once the line of symmetry has been found, the deformable template is used to determine the optimal face location. The size of the deformable template is reduced after each progression down the line of symmetry. Once again the amount by which the window size is decremented and size of the small 'jumps' or 'hops' of the window down the line of symmetry have to be determined. The window location, which caused the highest heuristic value (or 'faceness value') from the deformable template, is determined to contain a face. Once again, the decision was made not to use a threshold value since the very best face location and a high degree of accuracy was needed. Therefore an exhaustive search (limited only by the maximum and minimum pixel window sizes) was done down the line of symmetry. Some face detection examples are given below:

Figure 3.24 Fully automated face detection examples

It is obvious that the face detection output, on the lower right of Figure 3.24 is inaccurate. Here the best 'faceness' heuristic value does not correspond to the best face location, therefore an incorrect pixel area was segmented. The other three testing results are accurate to a high degree and may be called successful frontal view face detections. In chapter four the face detection system will be further optimised to reject false face detections and improve face detection accuracy.

3.5 Manual Face Detection

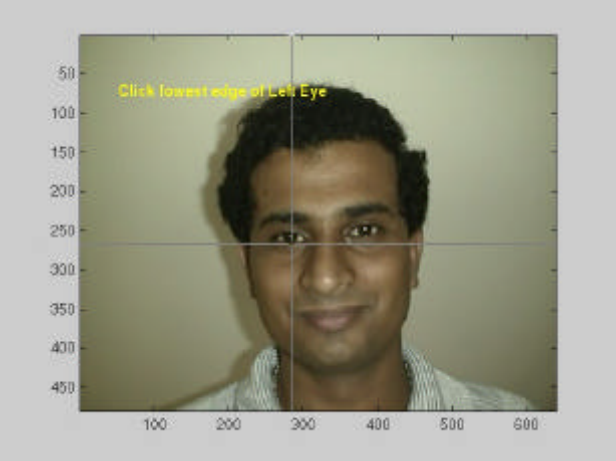
Besides a fully automated frontal view face detection system, a manual frontal view face detection system was also implemented. This was done because of the obvious improved accuracy a human operator could provide to the face detection process. In the system, the operator is asked to manually locate positions just under the subject's eyes, and the system thereby determines the exact face location.



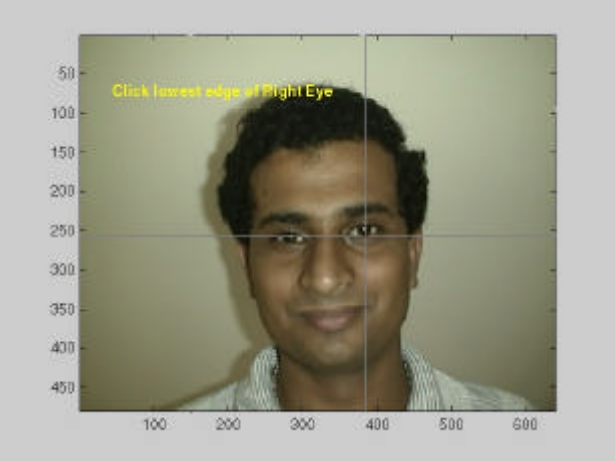
In the ideal frontal view segmented facial image for face recognition, the lower edge of each eye is 27% from the top of the image and the left and right eyes (operator's view point) are 20% and 80% from the left border of the image respectively

Figure 3.25 Average face in gray scale with manual face detection control points.

The ideal facial area to segment using manual face detection (Figure 3.25) was determined by examining the average face of 30 test subjects (the same average face was used to develop the deformable template). This segmented pixel area, which should be better than that obtained using fully automated face detection, can be used for automated face recognition



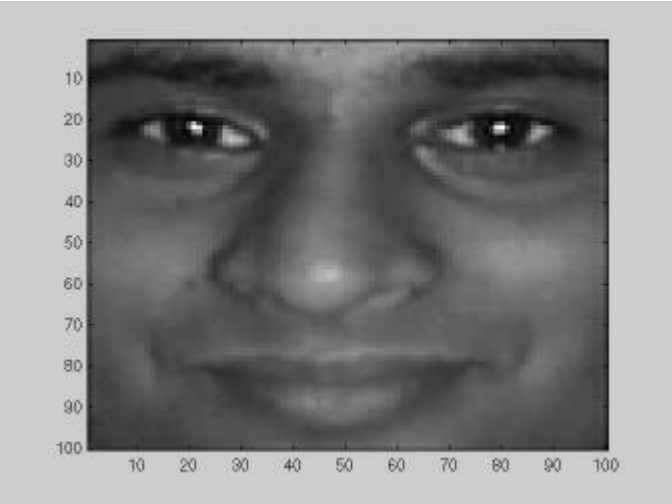
Operator instructed to click under eye on the left



Click under eye on the right

Image is rotated to make eyes perfectly horizontal. The system zooms in for increased accuracy and procedure repeated.

Operator clicks under both eyes again. From these locations the subject's frontal view face area is calculated.



Segmented face area

Figure 3.26 Manual face detection procedure

It would also have been possible to use more control points to determine the position of a subject's face in an image. For example, an operator can be instructed to click just under a subject's left and right eye, and then on the left and right edges of the subjects nose etc. However, the researcher decided to just use a single statistic (vector between lower edge of eyes) so as not to lose the natural variation between human faces. For example, if individual A has a long nose and individual B has a short nose, and if a manual face detection system centres and normalises face images based on eye width and nose length, a large part of the variation between the individual A's and B's faces is lost. Using a single statistic this does not occur. However, when one uses several control points to manually detect a face, the segmented face area would probably be more exact and the reader is encouraged to experiment with different combinations and test the face recognition accuracy of the segmented pixel area.

4.1 A human's innate face recognition system

Face recognition and the retention or memory of faces must be a crucial skill for our survival or evolution would not have given us these amazing abilities. We seem to have a natural preference for face like images and even a few weeks after birth, a new-born baby is attracted to face-like stimuli.

Much research has been done on the human face recognition system, and perceptual, developmental, neuro-psychological, neuro-physiological, and functional neuro-imaging studies have indicated that face recognition in primates is a *specialised* capacity in the ventral portions of occipito-temporal and frontal cortices and in the medial temporal lobes (Rodman et al., 1993). In fact, there is a condition called prosopagnosia, which is caused by brain injury, strokes or genetic factors. Suffers are unable to recognize faces while object recognition and other visual skills are largely unimpaired (Gauthier et al., 1999). Similarly there are patients with visual object agnosia, who are impaired at recognizing objects but who have normal face recognition abilities (Moscovitch et al., 1997). These findings seem to corroborate the theory that there are specialized areas in the brain that perform face recognition.

Although the areas of our brain, which perform face recognition, are known, the methodology of our innate face recognition system is not completely understood.

The most popular hypothesis among neuro-psychologists is that all of us have an average face prototype, assembled from all the faces we have seen throughout are lifetime. We then categorise each face we come across according to that particular face's variation from our average face prototype (Haxby et al.,1996 and de Haan et al., 1998). Work using caricatures of faces has collaborated this theory. When the Euclidean distance of an individual's face from the average face prototype is increased creating a caricature or exaggerated face, it was found that test subjects displayed increased recognition of that individual (Deffenbacher, 1998 and Brennan, 1982).

It occurred to the author that there might be other evidence to confirm the average face prototype hypothesis. For example, if an individual who has never been to China arrives at Shanghai, he would not be able to differentiate between the many oriental faces surround him. While after sometime our traveller would begin to notice differences among the faces of his Chinese friends. This may be because his initial average face prototype and the facial differences that he noticed, could not effectively describe variation among oriental faces. Any two Chinese faces would therefore seem similar. As our traveller stayed in Shanghai, his average face prototype and the variations he notices evolves to be able to differentiate between Chinese faces.

The statistical technique, which is used in this thesis for automated face recognition will be of special interest because it closely resembles our own innate face recognition system. This model promises recognition accuracy far in excess of a basic template matching technique, which involves comparing raw pixel intensity values.

4.2 Principal Component Analysis

Principal Component Analysis (or Karhunen-Loeve expansion) is a suitable strategy for face recognition because it identifies variability between human faces, which may not be immediately obvious. Principal Component Analysis (hereafter PCA) does not attempt to categorise faces using familiar geometrical differences, such as nose length or eyebrow width. Instead, a set of human faces is analysed using PCA to determine which 'variables' account for the variance of faces. In face recognition, these variables are called eigenfaces because when plotted they display an eerie resemblance to human faces.

Although PCA is used extensively in statistical analysis, the pattern recognition community started to use PCA for classification only relatively recently. As described by Johnson and Wichern (1992), *'principal component analysis is concerned with explaining the variance-covariance structure through a few linear combinations of the original variables.'* Perhaps PCA's greatest strengths are in its ability for data reduction and interpretation. For example a 100x100 pixel area containing a face can be very accurately represented by just 40 eigen values. Each eigen value describes the magnitude of each eigenface in each image.

Furthermore, all interpretation (i.e. recognition) operations can now be done using just the 40 eigen values to represent a face instead of the manipulating the 10000 values contained in a 100x100 image. Not only is this computationally less demanding but the fact that the recognition information of several thousand

4.3 Understanding Eigenfaces

Any grey scale face image I(x,y) consisting of a NxN array of intensity values may also be consider as a vector of N^2. For example, a typical 100x100 image used in this thesis will have to be transformed into a 10000 dimension vector!

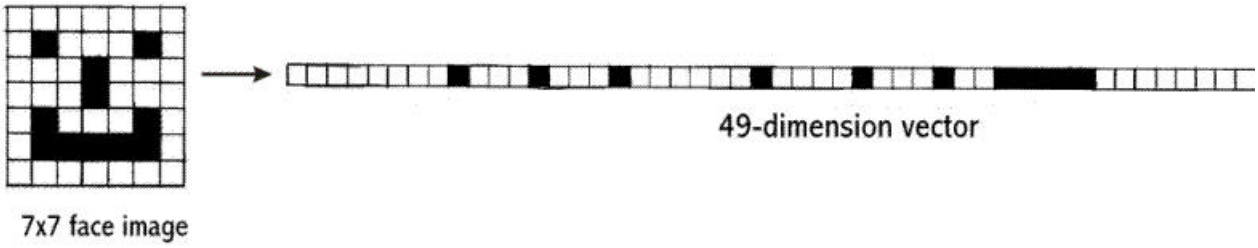


Figure 4.1 A 7x7 face image transformed into a 49 dimension vector

This vector can also be regarded as a point in 10000 dimension space. Therefore, all the images of subjects' whose faces are to be recognized can be regarded as points in 10000 dimension space. Face recognition using these images is doomed to failure because all human face images are quite similar to one another so all associated vectors are very close to each other in the 10000-dimension space.

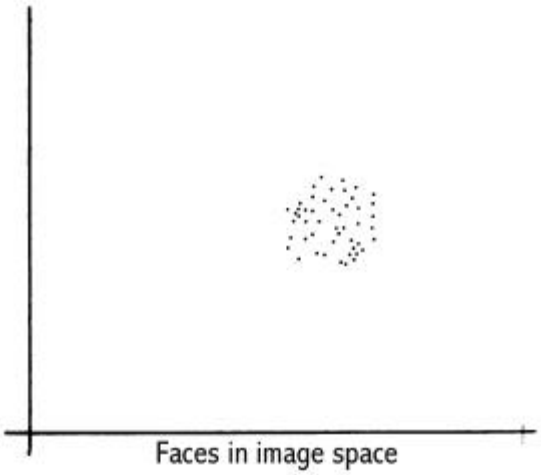


Figure 4.2 Faces in image space

Therefore classification of a new vector (image) would be a very sensitive process since even a slight change in the image would cause it to be nearer another face image than the subject's face in the face database.

The original variables (vectors) which described the face (pixel intensity at [1,1], pixel intensity at [1,2], ……) are highly correlated. With PCA, the researcher tried to find a better representation of faces by finding the specific vectors that account for the distribution of face images. These vectors will define the subspace of face images (sometimes called 'face space'). Face space will be a better representation for face images than image space which is the space which containing all possible images since there will be increased variation between the faces in face space (O'Toole et al. 1993).

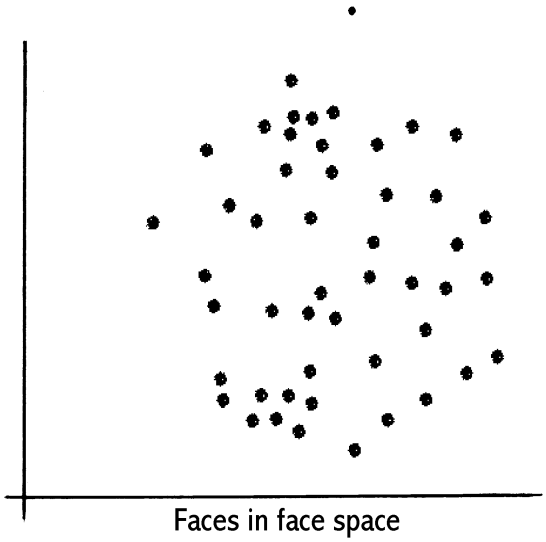


Figure 4.3 Faces in face space

The vectors that describe faces in face space are eigenfaces. These are in fact the eigenvectors of the covariance matrix of a set of mean subtracted face images (subtract the average face from each of the face images). Since a typical face image used in this thesis is 100x100 (therefore associated vectors 10000x1), and if there are 30 face images in the training set for PCA, the covariance matrix (C) would be:

$$C = X X^T$$

Here X, is a 10000x30 matrix containing the mean subtracted face images. Therefore, the covariance matrices dimensions would be 10000x10000. Calculating this matrix would be

an impossible task for most modern computers. This is one of the problems of using PCA in pattern recognition since high dimension vectors (i.e. images) are used. A computationally feasible method must be found to calculate eigenfaces.

When calculating eigenfaces, the number of faces (data points, i.e. 30) that are used will always be less than the dimension of the images (10000). Even if the reader wanted to recognize the whole population of China, just a few hundred Chinese faces (hopefully a representative sample) can be selected for Principal Component Analysis. Therefore instead of calculating $C=X\,X^T$, calculate

$$c=X^T\,X,$$

find c's eigenvectors, and thereby deduce the eigenvectors(eigenfaces) of C. In a PCA with 30 images (30 data points), c will be 30x30 and can easily be calculated.

Then the matrix of eigenvectors(v) and matrix of eigenvalues(λ) for c are a vectors and scalars that satisfy,

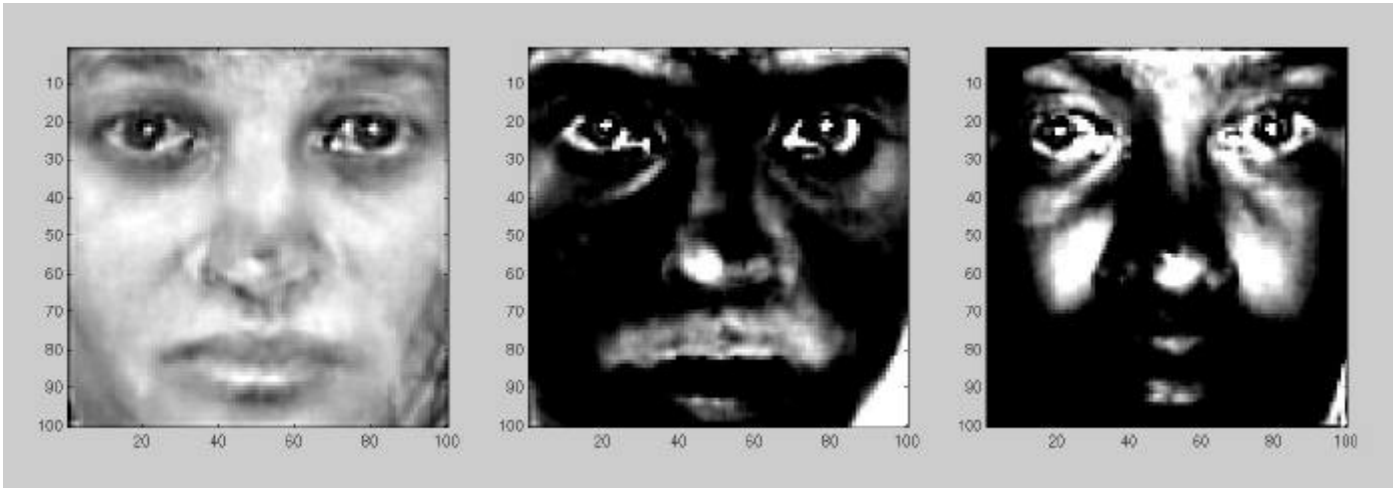
$$c\,v = \lambda\,v$$

All the eigenvectors, which were calculated, need not be used. Further, dimensionality reduction can be done by sorting the eigenvectors according to their associated eigenvalues and just taking (say 40) eigenvectors with the largest eigenvalues. These describe the greatest variation in human faces. Now that the eigenvectors of c have been found, the eigenvectors of C (eigenfaces) are in the matrix U where,

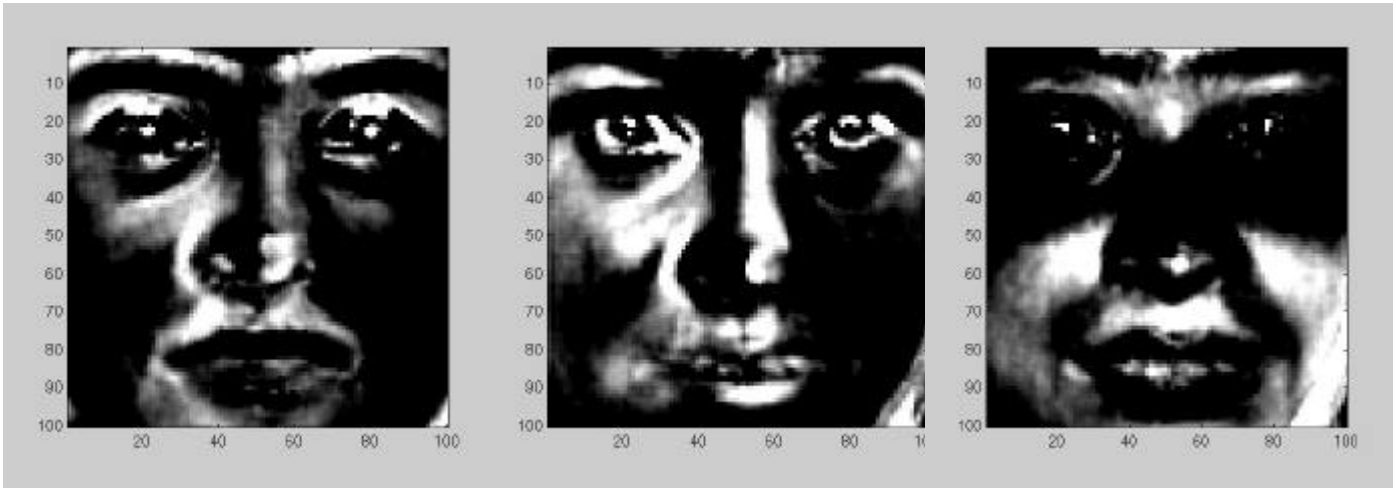
$$U=X\,v$$

Any face can be described using these eigenfaces. For a detailed description of Principal Component Analysis for face images the reader is encouraged to refer Turk and Pentland (1991a). Further details and proofs of the PCA results used in this thesis can be found in highly enjoyable, specialised mathematics textbooks such as Dunteman(1989), and Narayanaswamy and Raghavarao(1991) .

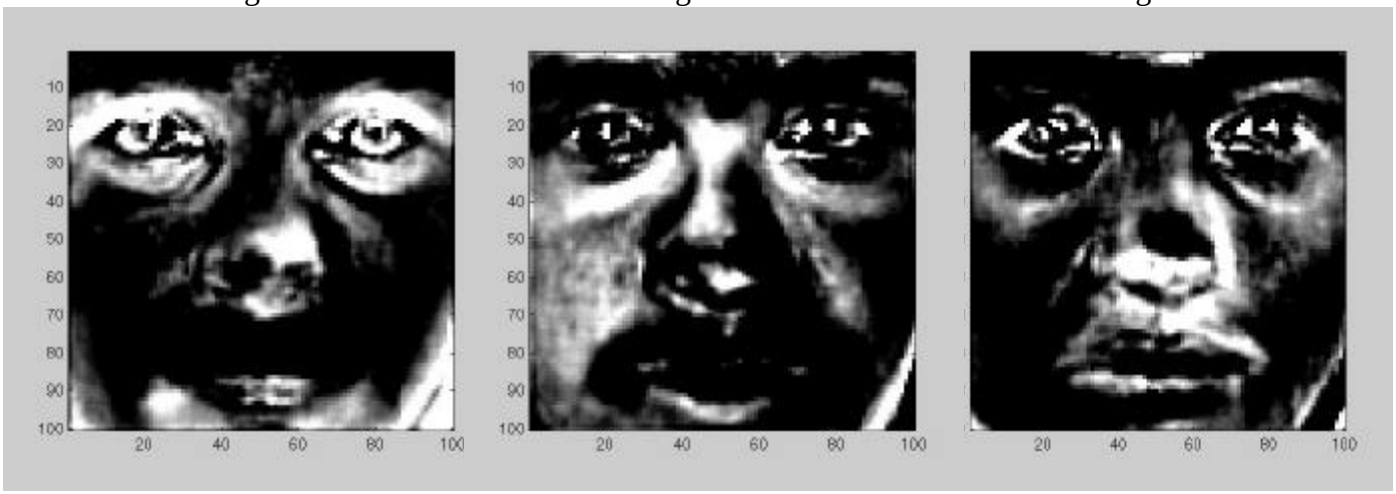
The following figures contain the first 18 eigenfaces calculated from 30 frontal view face images.



Eigenface 1 Eigenface 2 Eigenface 3

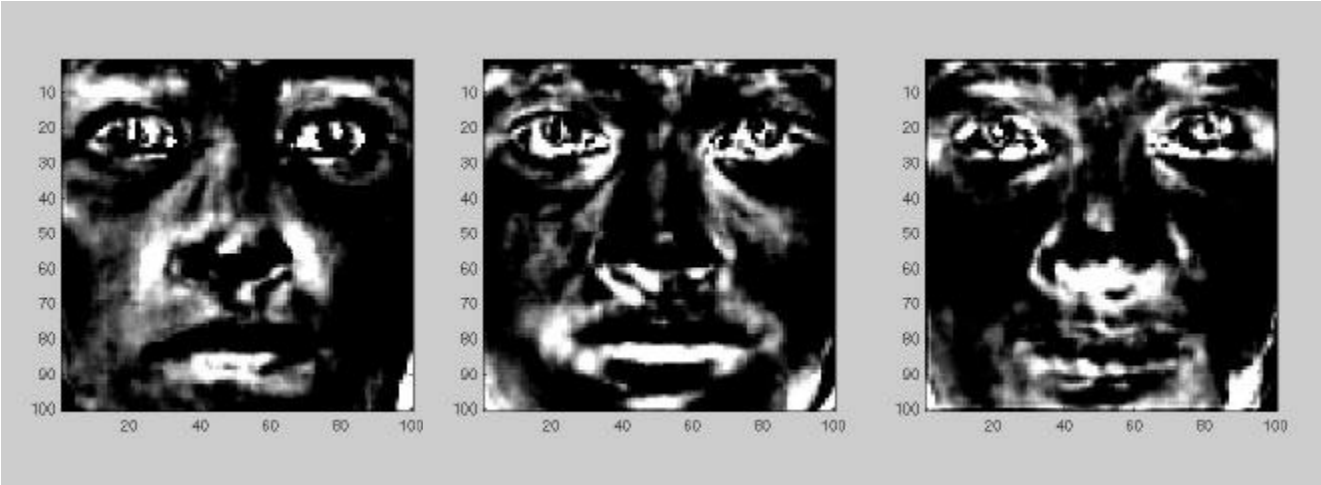


Eigenface 4 Eigenface 5 Eigenface 6

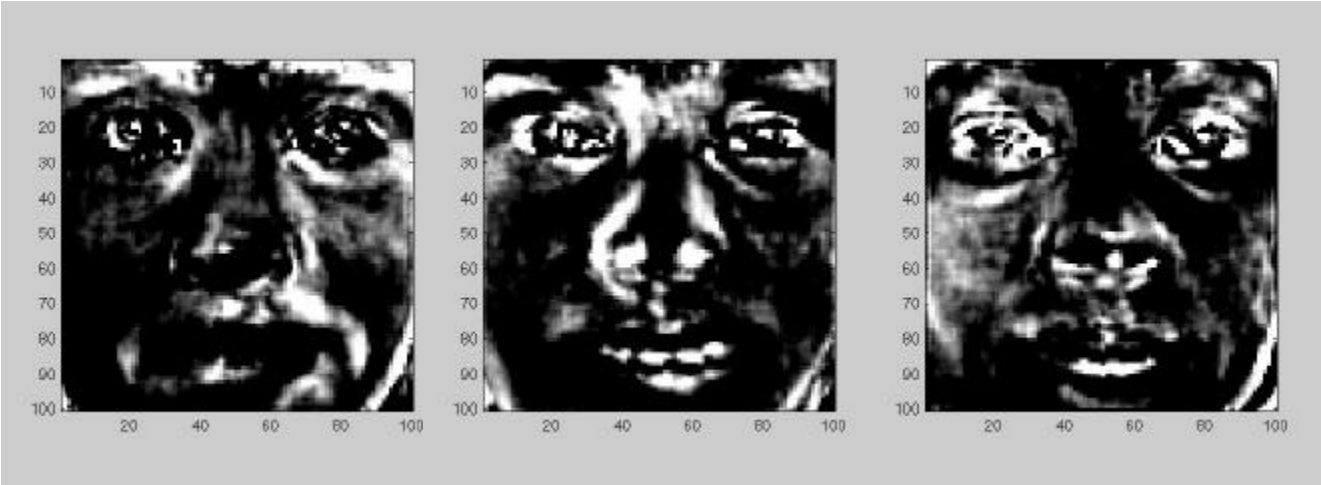


Eigenface 7 Eigenface 8 Eigenface 9

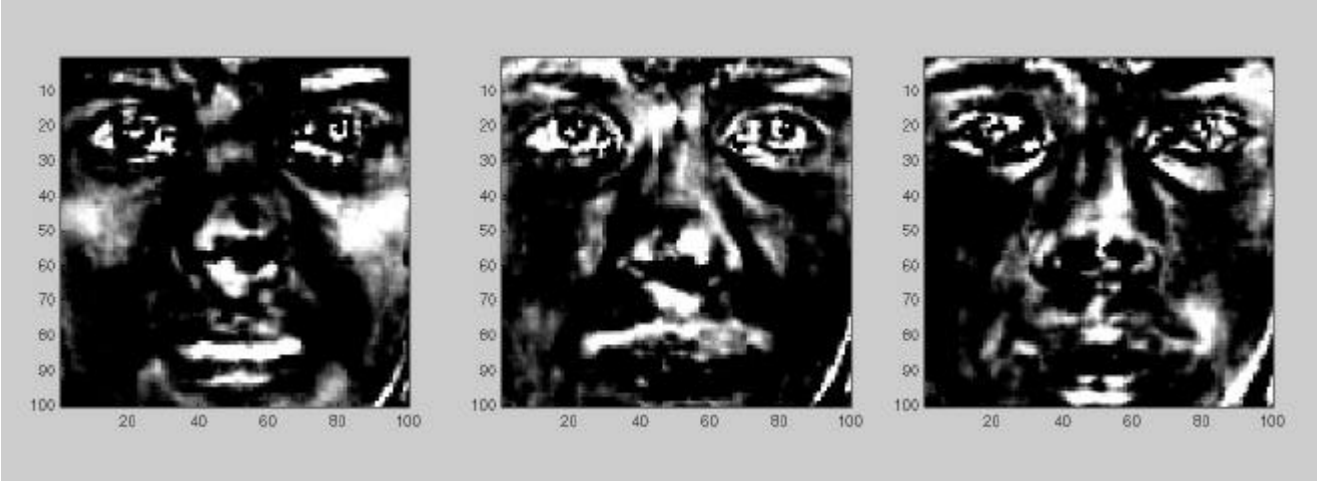
Figure 4.4 Eigenface 1 to 9 generated from 30 frontal view face images



Eigenface 10 Eigenface 11 Eigenface 12



Eigenface 13 Eigenface 14 Eigenface 15



Eigenface 16 Eigenface 17 Eigenface 18

Figure 4.5 Eigenfaces 10 to 18 generated from 30 frontal view face images

When a face is projected from image space to face space, its face space vector consists of values corresponding to each eigenface.

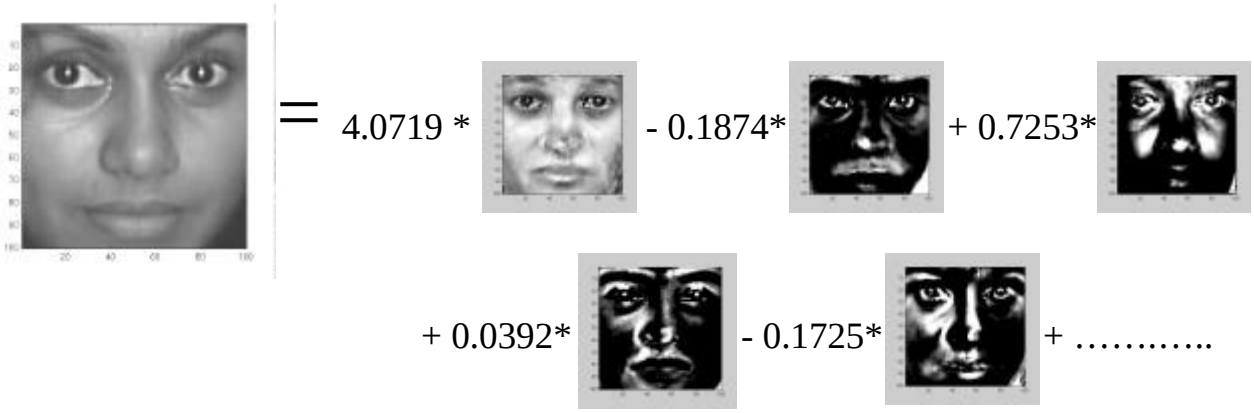


Figure 4.6 Graphical representation of the vector of a face in face space.

Increasing the number of eigenvectors (or eigenfaces) that are used to describe a face, as well as normalising the face space vector, will improve the accuracy and classification performance of the face recognition system.

4.4 Recognition

The transformation of a face from image space (I) to face space (f) involves just a simple matrix multiplication. If the average face image is A and U contains the (previously calculated) eigenfaces,

$$f = U * (I - A)$$

This is done to all the face images in the face database (database with known faces) and to the image (face of the subject) which must be recognized. The possible results when projecting a face into face space are given in the following figure.

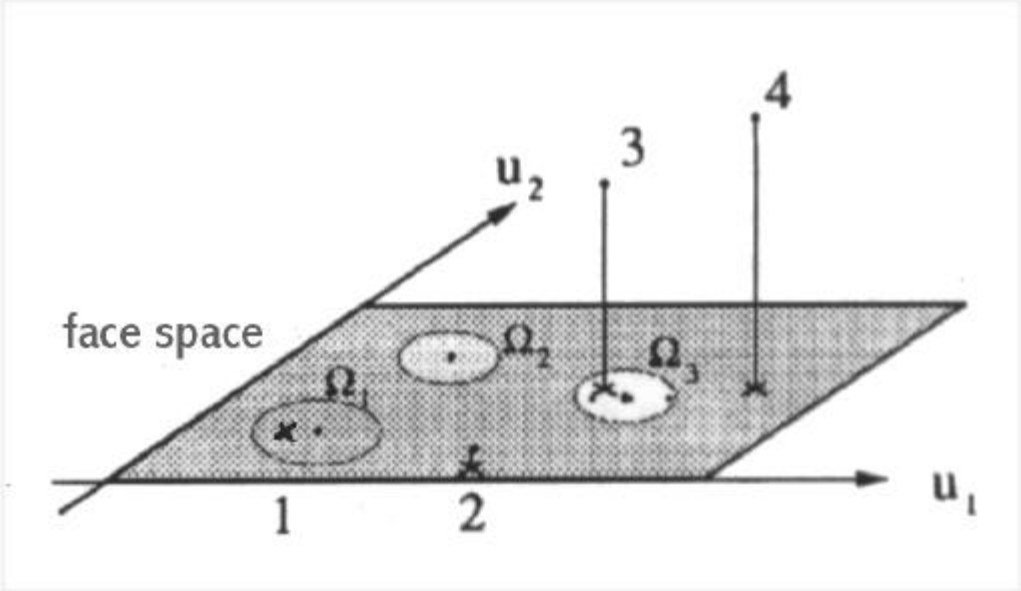


Figure 4.7 The four possible results when projecting an image into faces space. The face space is formed by just two eigenfaces (u_1 and u_2) and contains the faces of three known individuals (Ω_1, Ω_2 and Ω_3)

There are four possibilities:

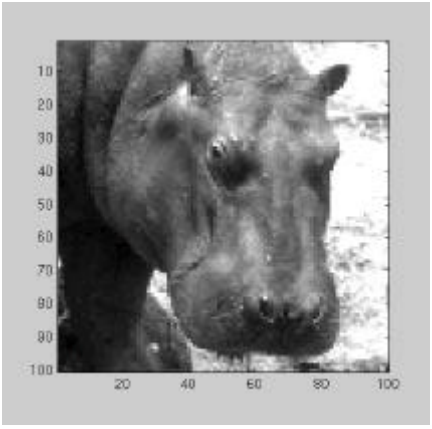
1. Projected image is a face and is transformed near a face in the face database
2. Projected image is a face and is not transformed near a face in the face database
3. Projected image is not a face and is transformed near a face in the face database
4. Projected image is not a face and is not transformed near a face in the face database

While it is possible to find the closest known face to the transformed image face by calculating the Euclidean distance to the other vectors, how does one know whether the image that is being transformed actually contains a face? Since PCA is a many-to-one transform, several vectors in the image space (images) will map to a point in face space (the problem is that even non-face images may transform near a known face image's faces space vector).

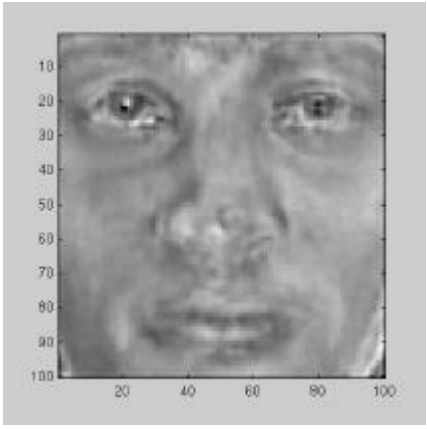
Turk and Pentland (1991a), described a simple way of checking whether an image is actually of a face. This is by transforming an image into face space and then transforming it back (reconstructing) into image space. Using the previous notation,

$$I' = U^T * U * (I - A)$$

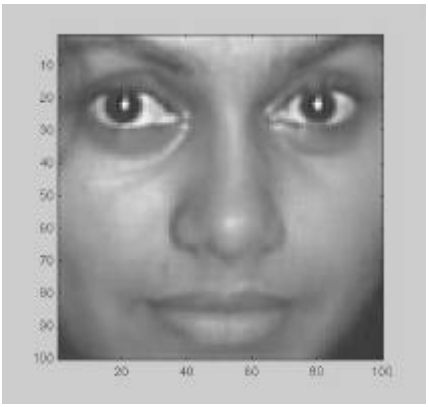
Where I' is the reconstructed image. Then the Euclidean distance between (I'+A) and I can be calculated to find out if I actually is of a face. The following figure describes this well.



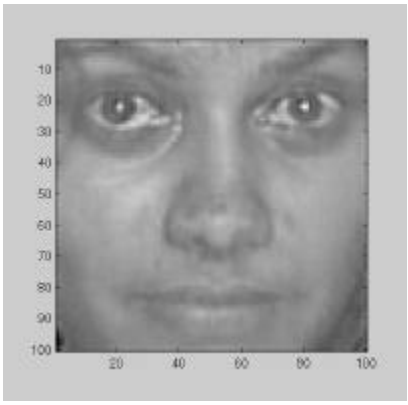
0.8235
0.0661
-0.8786
-0.4727
-0.0646
0.6642
-0.4840
-0.4501
-0.2506
0.1591
0.3359
0.0048
0.0745
……..



Hippo in image space | Hippo in face space | Reconstructed hippo in image space



0.7253
-0.0392
0.2896
-0.1725
-0.2642
-0.0014
-0.0814
-0.0054
-0.0623
-0.0965
-0.0879
0.0745
-0.0261
……



Face in image space | Face in face space | Reconstructed face in image space

Figure 4.8 Images and there reconstruction. The Euclidean distance between a face image and its reconstruction will be lower than that of a non-face image

With these calculations it is possible to verify that an image is of a face and recognise that face. O'Toole et al. (1993) did some interesting work on the importance of eigenfaces with large and small eigenvalues. They showed that the eigenvectors with larger eigenvalues convey information relative to the basic shape and structure of the faces. This kind of information is most useful in categorising faces according to sex, race etc. Eigenvectors with smaller eigenvalues tend to capture information that is specific to single or small subsets of learned faces and are useful for distinguishing a particular face from any other face. Turk and Pentland (1991a) showed that about 40 eigenfaces were sufficient for a very good description of human faces since the reconstructed image have only about 2% RMS. pixel-by-pixel errors.

4.5 Improving face detection using reconstruction

Reconstruction cannot be used as a means of face detection in images in near real-time since it would involve resizing the face detection *window* area and large matrix multiplication, both of which are computationally expensive. However, reconstruction can be used to verify whether potential face locations identified by the deformable template algorithm actually contain a face. If the reconstructed image differs greatly from the face detection *window* then the *window* probably does not contain a face. Instead of just identifying a single potential face location, the face detection algorithm can be modified to output many high 'faceness' locations which can be verified using reconstruction. This is especially useful because occasionally the best 'faceness' location found by the deformable template algorithm may not contain the ideal frontal view face pixel area.

Best heuristic location (94,65,20)

Actual face location (85,58,38)

Output from Face detection system

Heuristic	x	y	width
978	74	31	60
1872	74	33	60
1994	75	32	58
2125	76	32	56
2418	76	34	56
2389	79	32	50
2388	80	33	48
2622	81	33	46
2732	82	32	44
2936	84	33	40
2822	85	58	38
2804	86	60	36
2903	86	62	36
3311	89	62	30
3373	91	63	26
3260	92	64	24
3305	93	64	22
3393	94	65	20

Figure 4.9 Face detection output

Potential face locations that have been identified by the face detection system (the best face locations it found on its search) are checked whether they contain a face. If the threshold level (maximum difference between reconstruction and original for the original to be a face) is set correctly this will be an efficient way to detect a face. The deformable template algorithm is fast and can reduce the search space of potential face locations to a handful of positions. These are then checked using reconstruction. The number of locations found by the face detection system can be changed by getting it to output, not just the best face locations it has found so far but any location, which has a 'faceness' value, which for example is, at least 0.9 times the best heuristic value that has been found so far. Then there will be many more potential face locations to be checked using reconstruction. This and similar speed versus accuracy trade-off decisions have to be made keeping in mind the platform on which the system is implemented.

Similarly, instead of using reconstruction to check the face detection system's output, the output's correlation with the average face can be checked. The segmented areas with a high correlation probably contains a face. Once again a threshold value will have to be established to classify faces from non-faces. Similar to reconstruction, resizing the segmented area and calculating its correlation with the average face is far too expensive to be used alone for face detection but is suitable for verifying the output of the face detection system.

4.6 Pose invariant face recognition

Extending the frontal view face recognition system to a pose-invariant recognition system is quite simple if one of the proposed specifications of the face recognition system is relaxed. Successful pose-invariant recognition will be possible if many images of a known individual are in the face database. Nine images from each known individual can be taken as shown below. Then if an image of the same individual is submitted within a 30^o angle from the frontal view he or she can be identified.

Nine images in face database from a single known individual

Unknown image from same individual to be identified

Figure 4.10 Pose invariant face recognition.

Pose invariant face recognition highlights the generalisation ability of PCA. For example, when an individual's frontal view and 30^o left view known, even the individual's 15^o left view can be recognised.

Although good results can be obtained for pose invariant face recognition under controlled conditions, pose invariant face detection is extremely hard and successful template matching or traditional neural network strategy to deal with the problem has still not been proposed. Furthermore, gathering multiple views of an individual for a face database is not realistic since in most situations only a single known face for each subject is available.

Since the face was not isolated in the image, variations in the subject's clothing, hairstyle or the background would adversely effect recognition performance. The pose invariant face database and testing images gathered in this thesis were obtained under highly controlled conditions since the pose invariant face recognition system is not as robust as the frontal view system.

5.1 Overcoming irregular lighting

Very rarely will a face recognition system be presented with perfect lighting conditions with ambient lighting not causing distinct irregular shadows on a subject's face. If a face recognition system were to be used in real-world environments, under varying light conditions, it must be able to overcome irregular lighting. Therefore the following image processing techniques where used during the author's research to provide a degree of lighting invariency.

5.2 Normalising

The following image was captured under lighting conditions which to a human's visual system, do not seem that adverse for face recognition

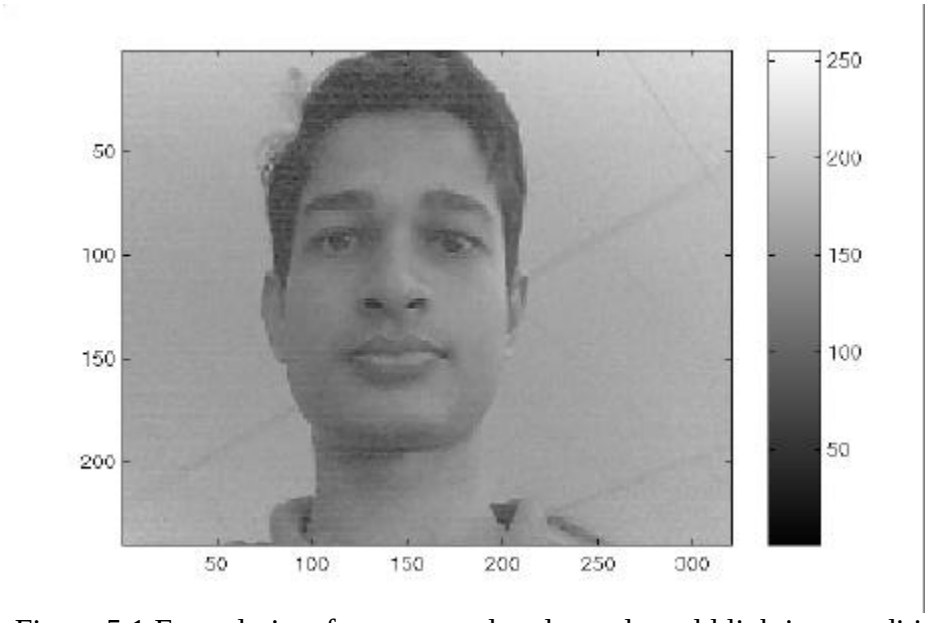


Figure 5.1 Frontal view face captured under real-world lighting conditions

However by examining this image under a suitably scaled colormap it is evident that there is extremely irregular lighting prevalent (Figure 5.2).

These conditions would adversely affect the performance of the whole face recognition and detection system and therefore must be adjusted.

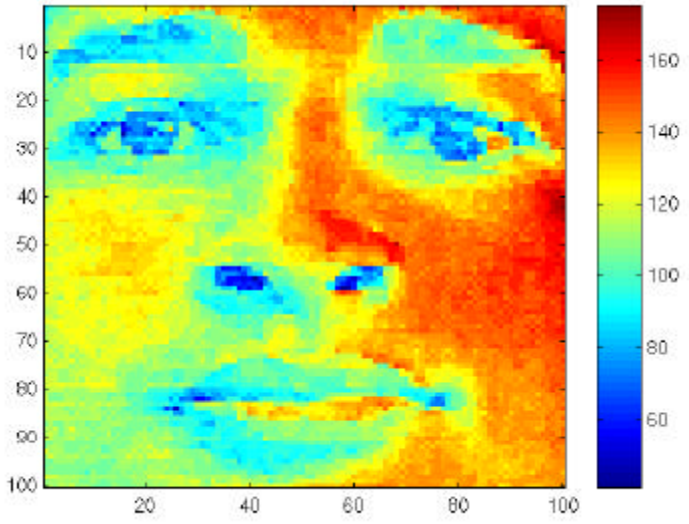


Figure 5.2 Face with a suitably scaled colormap

The worst case scenario is when there is a single directional light source on one side of the subject's face, similar to the above figure. Furthermore, most lighting irregularities in the real world are with a light source on either side of the subject. Because of the inherent contours of a human face vertical differences in lighting do not seem to affect the frontal face image a great deal. After examining many image processing techniques it was found that normalising the columns of the image matrix (normalising the image in the vertical direction) was an effective method of overcoming irregular lighting (Figure 5.3).

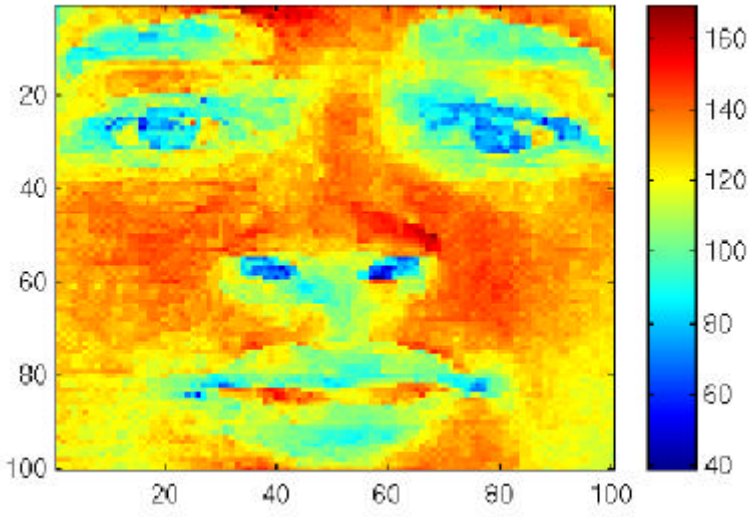


Figure 5.3 Face after normalising vertically

Normalising horizontally and even normalising horizontally after normalising in the vertical direction was found to give unsatisfactory results (Figures 5.4 and 5.5)

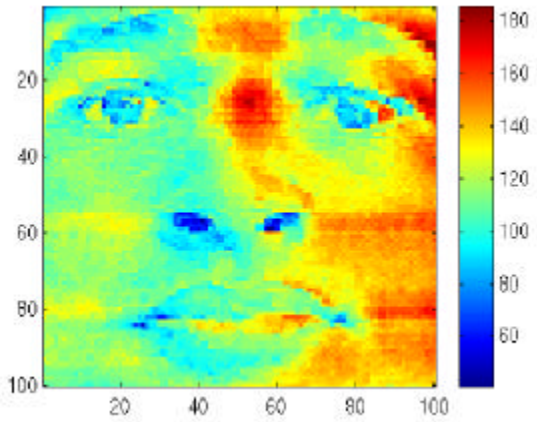


Figure 5.4 Face after normalising horizontally

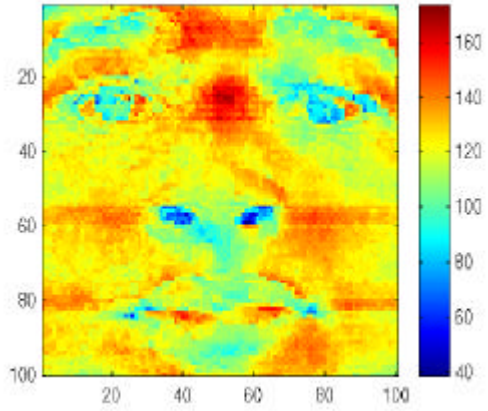


Figure 5.5 Face after normalizing vertically then normalizing horizontally.

Notice the horizontal bands that appear along rows with high intensity areas. Horizontal normalisation should not be used in similar computer vision problems.

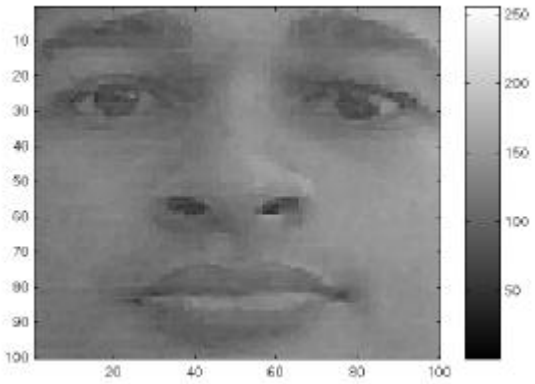


Figure 5.6 Face under real-world lighting

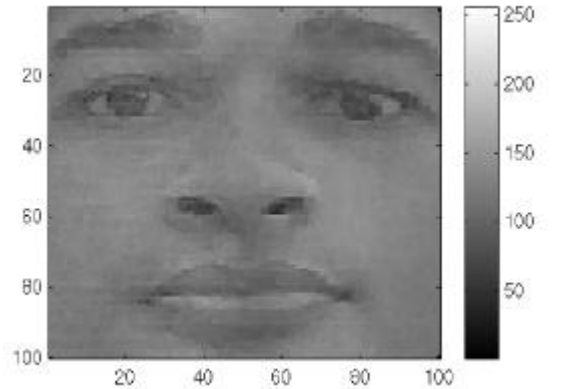
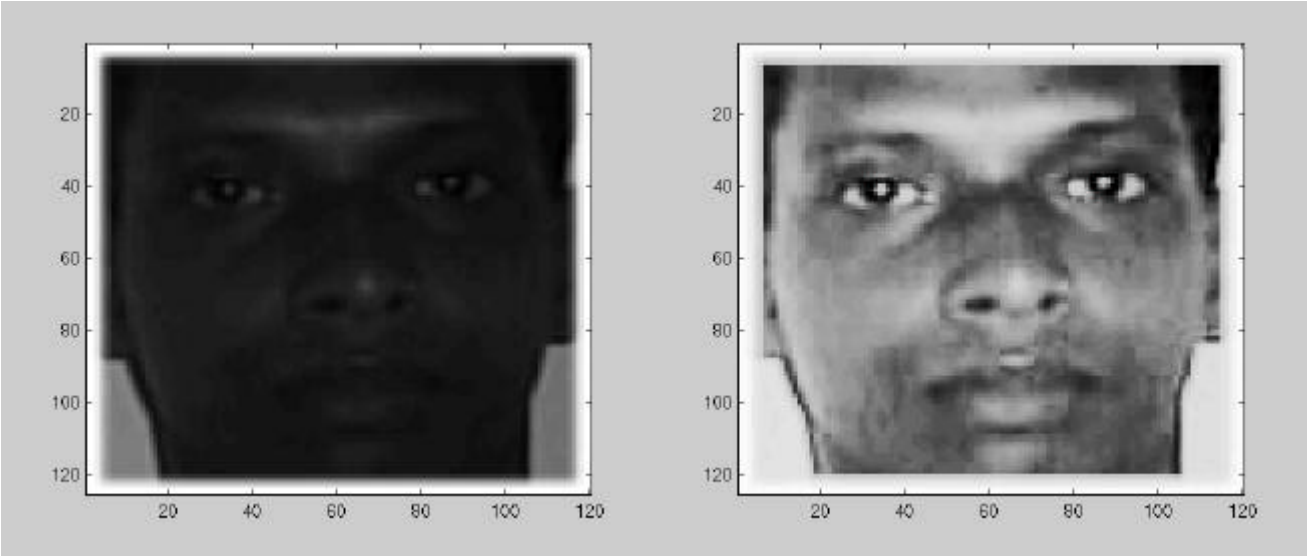


Figure 5.7 Face after vertical normalization

The original and vertically normalised images are shown above using the normal grey scale colormap. Although they both look similar to the human visual system the image on the right is far superior for reasoning using computer vision and for face recognition purposes.

5.3 Histogram Equalisation

The digital camera, which the author used to assemble the face database, had an automatic exposure, which unfortunately adversely affected the face images. Many of the images were too dark and not suitable for face recognition. Histogram equalisation equitably spreads the pixels of an image among the grey-level intensities thereby increasing the contrast of the image.



A very dark image. All grey-level intensities are low and close together

Histogram equalised image. Pixels were gathered into 20 grey-level intensities

Figure 5.8 Histogram equalisation

5.4 Order-Statistic Filtering

A serious error was made by the author when assembling the main face database. While the images in section 6.1.1 were captured without a flash, the author used digital camera with a flash when photographing subjects for this project's main face database. Using the flash frequently caused highlights in the subjects eyes which affected the vertical normalization pre-processing as described in the previous section and also adversely affected face recognition accuracy. Since over 450 photographs were taken in the main face database, re-taking the whole database was not an option. Also, the author did not want to manually alter the face images since this would detract from the value of the overall system. An image processing technique was needed to isolate and remove the highlights.

Figure 5.9 Highlights from camera flash

Figure 5.10 After order-statistic filtering

A 2 dimensional order statistic filter, which replaces each element in a 3x3 neighbourhood with the minimum element in the neighbourhood was found to provide satisfactory results. Order-statistic filtering encompasses many useful image processing techniques including the popular median filter. The reader is encouraged to read refer Haralick and Shapiro (1992) for further details.

System Testing

The following systems were implemented using Matlab 5.3 and tested on an Intel Pentium 233 MMX, with 64MB of RAM running Windows '95. This platform should be considered as the minimum hardware requirement since the face detection and recognition algorithms could have been modified for increased accuracy on a more powerful testing platform.

1) Fully automated face detection with verification
2) Fully automated face detection without verification
3) Manual face detection and automated face recognition
4) Fully automated face detection and recognition
6) Pose invariant face recognition

450 images from 30 test subjects were obtained to test the above systems. The data for testing the fully automated face detection system, manual face detection and automated face recognition system and the fully automated face detection and recognition consists of 4 frontal view images per test subject. The first image was taken under 'good' conditions with relatively constant lighting conditions on a white background. This would be used as the known frontal view face image in the face recognition system. The environment condition of the image was categorised by the researcher as 'A'.

The other three frontal view images were taken under worsening conditions with adverse lighting conditions and sometimes with a black background. These would be used as test images for the frontal view face recognition system. An effort was made to vary the lighting as much a possible in the environment which the images were gathered to test the systems' robustness. The environment condition of the image was categorised as 'B'.

Data for the pose invariant face recognition was gathered as follows. Nine known images from each individual were collected and three (unknown) images taken when the subject was posing in intermediate angles between the nine known images. The nine known images were taken with the test subject posing as in figure 4.10. Since pose invariant face recognition is not as robust as frontal view face recognition this data was gathered under very controlled conditions

The images from three test subjects had to be rejected since many of these had been very adversely affected by the automatic exposure of the digital camera used to obtain the face images. A subset of the face database can be found on the compact disk accompanying this thesis.

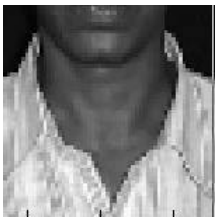
6.1 Fully automated face detection

The output of the face detection system (segmented face area) is subjectively evaluated to fall into the following categories:

1) **Successful** detection, **Excellent** face segment.

2) **Successful** detection, **Good** face segment

3) **Failure**



These are realistic categorisations since even Heisele and Poggio (1999) considered the following as face detections.

Figure 6.1 Face detection examples from Heisele and Poggio (1999)

6.1.1 Fully automated face detection testing without verification.

First, the fully automated frontal view face detection system was tested without verifying its results (without using reconstruction or correlation with the average face). The following table contains the testing results:

Condition of images	Total tested	Successful detection		Failures
		Excellent	Good	
A	27	17 (62%)	8 (29%)	2 (7%)
	27	**25 (92%)**		**2 (7%)**
B	60	27 (45%)	24 (40%)	9 (15%)
	60	**51 (85%)**		**9 (15%)**

Condition A refers to frontal view images taken with a plain white background under relatively controlled lighting conditions, while condition B images are those randomly taken from the set of images that were taken under worsening lighting conditions and sometimes with a black background. When considering all the trade-offs that were made, the adverse environment conditions and the degradation to the images caused by the digital camera's automatic exposure, the successful detection rate is quite high. While the successful detections with a good segment area will probably require modest alterations in the face window location to be done by the face recognition system, the excellent face segments will need little adjustment. The PCA recognition is extremely sensitive to scale and position variations so small adjustments to the face detection output will always be needed. Unfortunately the segmented areas from the face detection failures are totally useless for face recognition. These errors occurred when the best 'faceness' heuristic did not coincide with the actual best face location.

6.1.2 Fully automated face detection testing with verification.

Here correlation with the average face is used to verify the potential face locations proposed by the fully automated face detection system. Condition A and B images are as with the previous test.

Condition of images	Total tested	Successful detection		Failures
		Excellent	Good	
A	27	21 (78%)	6 (22%)	0 (0%)
	27	**27 (100%)**		**0 (0%)**
B	81	64 (79%)	17 (21%)	0 (0%)
	81	**81 (100%)**		**0 (0%)**

This has proved to be a very efficient automated face detection system. The deformable template effectively narrowed down the search space and then correlation with the average face identified the best face location. Furthermore, there were no detection failures when we used correlation to verify the output of the face detection system. The 6 good face detection results from condition A images are given in the following figure.

Figure 6.2 The 6 good face segments from 27 condition A images. All other face detections resulted in excellent face segments. Many of the above were not segmented perfectly because the subject's head was slightly at an angle or like in the lower right figure, the subject was not perfectly symmetrical.

6.2 Automated face recognition

6.2.1 Manual face detection and automated face recognition

These tests will investigate the robustness and accuracy of the Principal Component Analysis frontal-view face recognition system. The 27 images of condition A (as described earlier) were manually face detected and PCA transformed, creating a face database of known images. The author divided the condition of images submitted for recognition into three types,

1) Perfect conditions - the known images are submitted again
2) Good conditions - manual face detection(again) of condition A images
3) Normal/Adverse conditions - manual face detection of condition B images.

The faces in the images were of different sizes, positions and many subjects had even slightly rotated their heads. Further more, lighting conditions were intentionally varied by the researcher to strain the recognition system.

Condition of images	Total tested	Successful recognition	Recognition failures
Perfect	27	27 (100%)	0 (0%)
Good	27	25 (93%)	2 (7%)
Normal/Adverse	80	58 (73%)	22 (27%)

While the face recognition system's performance is strong on 'perfect' and 'good' images its performance on 'normal/adverse' condition images could be better. This is probably because only 27 eigenfaces could be used in the fully automated face recognition system. Turk and Pentland (1991a) recommend using at least 40 eigenfaces for face recognition. Unfortunately because of the limited number of testing subjects this was not possible. Since according to O'Toole et al. (1993), low eigenvalue eigenfaces are the most useful for recognition, increasing the number of eigenfaces should improve performance considerably and a recognition accuracy of over 90% is expected.

6.2.2 Fully automated face detection and recognition

Simply attempting the recognition of the output of the fully automated face detection system resulted in a recognition rate close to zero (0%). Principal Component Analysis is extremely sensitive to even slight variations in scale, position and rotation so it is extremely unlikely that the extracted segment would be suitable for recognition.

Slightly moving the extracted segment around in the hope of getting better recognition results was viewed as a crude solution to the problem and also was too computationally expensive to implement on the testing platform.

The author investigated using the Hough transform after edge detection on the extracted segment as a means of eye detection. Detecting the eyes would enable normalising the face (in fact, extracting the exact face position from the segment) thus vastly improving recognition accuracy. Eye detection would let us rotate the face to make the eye horizontal and scale the face exactly to match the calculated average face. Unfortunately sufficient literature was not available to understand and implement such a system.

6.2.3 Pose invariant face recognition

Pose invariant face recognition was tested without any detection what so ever. This was possible because the face images were carefully gathered by the researcher under controlled conditions. Testing results follow:

Testing condition	Total tested	Successful recognition	Recognition failures
controlled	92	86 (93%)	6 (7%)

There is a very high recognition rate, even higher that the frontal view face recognition system with face detection. The sole reason of the strong performance of the system was the controlled environment that the images taken. This is in total contrast to the frontal view images which were intentionally taken under adverse conditions. These testing results underline the fact that creating a computer vision system that performs in a controlled domain is far easier then producing a very robust vision system.

6.3 Standard face databases

The author would have liked to compare these testing results with that of other face detection and face recognition studies but such a comparison would be deceptive and may even reflect adversely on the author's system. This is because one cannot compare systems tested using different face databases. The author again emphasises the fact that the performance of these vision systems depend on the conditions under which they were tested.

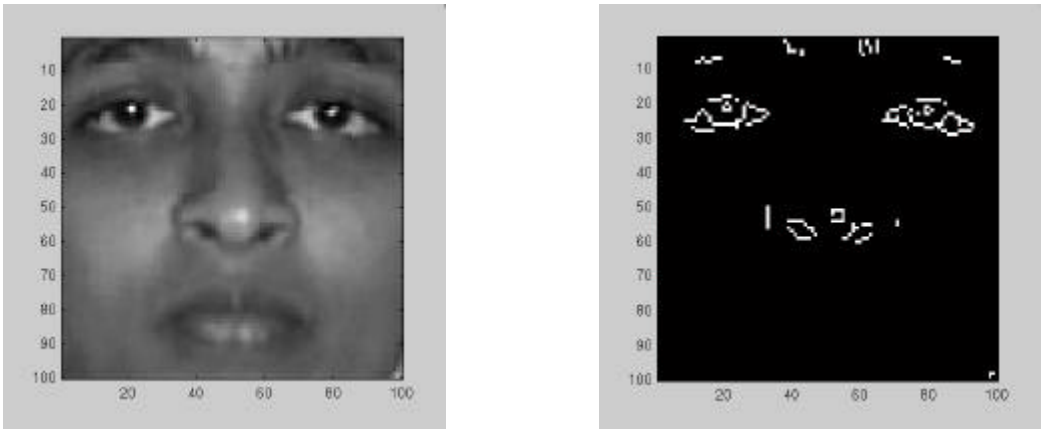
There are two facial image databases (containing both training and test data) which are widely used by researchers to test face detection and recognition systems. These enable systems to be tested on an even footing. The NATO'97 database is one that was created during the NATO Advanced Study Institute (ASI) on Face Recognition. The other database is the famous FERET (Face-Recognition Technology) database of the US Army Research Laboratory (ARL). FERET is in fact an on-going major program funded by the ARL and ARPA. The author was not able obtain any of these or any other standard face database, so was unable to compare the implemented systems with those of other researchers'.

7.1 Extensions to the implemented systems

The face database of the implemented system should be expanded to as many individuals as possible. Face recognition using Principal Component Analysis promises to be highly scalable so recognition of even over 10000 individuals should be possible. This is in contrast to a traditional neural network based technique. However, the real-world problems that arise when expanding the face database can only be found out by actually experimenting with a large face database.

Additional test subjects are also needed because there were insufficient eigenfaces calculated to perform recognition accurately. As we increase the size of the face database the recognition accuracy of the system should increase. In the author's opinion a minimum of 40 eigenfaces should be used for face recognition.

An eye detection system should be implemented to further normalise the area segmented by the face detection system. If the subject's eyes were accurately identified, the image could be transformed to make the eyes horizontal and the face scaled to a constant proportion of the segmented pixel area. This would let us implement a fully automated face detection and recognition system since PCA is far too sensitive to scale, rotation and shift, to perform recognition with the raw output from a face detection system. Fortunately there are well known techniques for circle detection which can also be used for eye detection.



Segmented Area from Face Detection system After edge detection using Sobel operators

Figure 7.1 Eye detection

Once the segmented area has been edge detected, the Hough transform can be used to identify the locations of circles in the image. The two large circles near the top of a segmented face would be the eyes. Due to the limited time available for this project the author could not implement an eye detection system.

Only the whole-face template was used for frontal view face recognition. The template matching strategy that was used can be expanded to use the whole face, left eye, right eye, nose and mouth templates for recognition. The problems associated in extracting these template areas and assigning weights to the Euclidean distances found by each template (closest known left eyes, closest known right eyes, etc) can be investigated.

If a real-world implementation of this system is needed, the face detection system should be optimised to suit the specific platform. There are several speed versus accuracy trade-off decisions which will have to be reconsidered depending on each computing environment that is used.

Besides the pose invariant recognition scheme that was implemented, recognition even with facial expressions is possible by taking multiple images of each known individual with a range of facial expressions.

The classification ability of PCA can be tested on other problems. While the recognition of human faces is an innate ability, we may not be able to differentiate between data points in other problems. PCA can then be used to find relationships which are not obvious to us. For example, perhaps PCA can be used in an automated orang-utan face recognition system!

7.2 Transforming frontal view face images for pose invariant face recognition

While the results for pose invariant face recognition were quite strong, the face database contained nine known faces for each individual to be identified. With more known data points in face space, recognition performance increased sharply. Unfortunately having nine known images per individual is not realistic (in most real-world situations only a single frontal view face image is presented as the known image), and therefore a great deal of research has been done on transforming frontal view face images to create 'virtual views' of the other poses. These 'virtual views' can then be added to the face database as additional known faces of an individual from different poses. There are two approaches to transforming or rotating human faces.

7.2.1 3D modelling techniques

These involve transforming the facial image to a 3D model, rotating the model and transforming the face back into a (2D) image. It is a 2D to 3D to 2D transformation. While this technique produces accurate results many implementations of this strategy require a frontal and a profile image per individual to create the 3D model (Lee and Thalmann, 1998) and is therefore once again not practical..

7.2.2 2D transformations

The other approach to the problem is to use a 2D to 2D parallel deformation to rotate the (2D) frontal view face image. Beymer and Poggio (1995) used an example based strategy to create prototype transformations from a frontal view face to a angled pose. These were then used in a pose invariant face recognition system

7.3 Support Vector Machines for pose invariant face detection

While traditional neural network based techniques are not ideally suitable for pattern recognition and computer vision problems, a new learning strategy has been proposed to enable neural networks to achieve high face recognition and face detection accuracy even though only a few examples (i.e. a small training set) were used for training (Osuna et al, 1997). This was done by using Support Vector Machines (SVM), which were developed by V. Vapnik and his team at the AT&T Bell Labs for training polynomial, neural network, or Radial Basis Functions classifiers.

A pose invariant face detection system could be attempted using SVMs, their ability to perform strongly even with a small training set would make them an ideal network model. A pose invariant face detection system together with a system for transforming frontal view face images would enable us to create a pose invariant face recognition system using only a single known frontal view face image per individual.

If all the above systems were implemented, not only would a fully automated face detection and recognition system be possible, but also a fully automated pose invariant face detection and recognition system. Further, even a fully automated real-time automated pose invariant face detection and recognition system could be realised by also using spatio-temporal filtering.

Conclusion

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable.

The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.

The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area.

The fully automated face detection and recognition system was not robust enough to achieve a high recognition accuracy. The only reason for this was the face recognition subsystem did not display even a slight degree of invariance to scale, rotation or shift errors of the segmented face image. This was one of the system requirements identified in section 2.3. However, if some sort of further processing, such as an eye detection technique, was implemented to further normalise the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system. Implementing an eye detection technique would be a minor extension to the implemented system and would not require a great deal of additional research.

All other implemented systems displayed commendable results and reflect well on the deformable template and Principal Component Analysis strategies.

The most suitable real-world applications for face detection and recognition systems are for mugshot matching and surveillance. There are better techniques such as iris or retina recognition and face recognition using the thermal spectrum for user access and user verification applications since these need a very high degree of accuracy.

The real-time automated pose invariant face detection and recognition system proposed in chapter seven would be ideal for crowd surveillance applications. If such a system were widely implemented its potential for locating and tracking suspects for law enforcement agencies is immense.

The implemented fully automated face detection and recognition system (with an eye detection system) could be used for simple surveillance applications such as ATM user security, while the implemented manual face detection and automated recognition system is ideal of mugshot matching. Since controlled conditions are present when mugshots are gathered, the frontal view face recognition scheme should display a recognition accuracy far better than the results, which were obtained in this study, which was conducted under adverse conditions. Furthermore, many of the test subjects did not present an expressionless, frontal view to the system. They would probably be more compliant when a 6'5" policeman is taking their mugshot!

In mugshot matching applications, perfect recognition accuracy or an exact match is not a requirement. If a face recognition system can reduce the number of images that a human operator has to search through for a match from 10000 to even a 100, it would be of incredible practical use in law enforcement.

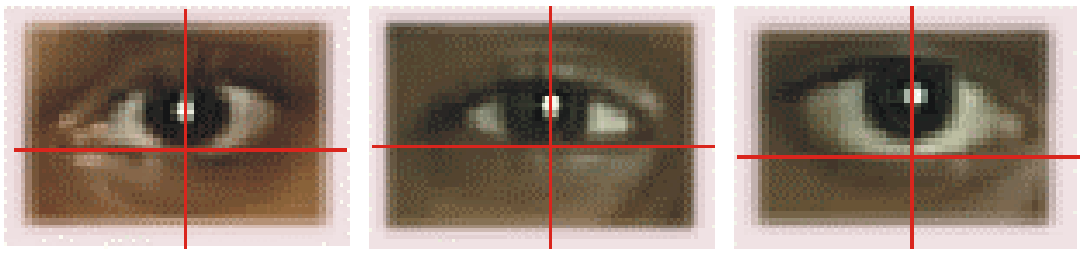
The automated vision systems implemented in this thesis did not even approach the performance, nor were they as robust as a human's innate face recognition system. However, they give an insight into what the future may hold in computer vision.

Appendix A - User Manual

"Big Brother - Frontal View system"

All possible operations of the "Big Brother - Frontal View system" are listed below

- Load Image - load an image into the system.
- Manual Face detection - the operator must manually detect the face according to the instruction given by the system. Since operator skill plays a great part in a manual face detection and automated face recognition system, the following figure are given as a guideline to the user.



The operator of the manual face recognition system must carefully mark the given point when the eye is in the particular position given above. Unfortunately, even slight deviations from this point will affect subsequent recognition accuracy.

- Automated Face Detection - the system will detection the face in the loaded image.
- Recognize - the system will attempt to recognize the manually or automatically detected face. The face database must be loaded into the system before face recognition can be carried out. The face of the closest known face in the database will be displayed. The user has the option of viewing the details of the next closest known face.
- Add to Face Database - the current face is added to the face database
- PCA - Compute eigenfaces of faces in the face database.

Similar functionality can be found in ""**Big Brother - Pose Invariant system**" except no face detection takes place and face recognition is with faces from different angles.

Appendix B - Code Listing

Mathwork's Matlab 5.3 is required to use the Matlab functions that follow. All the functions used in this thesis and a subset of the face database can be found on the compact disk enclosed with this thesis.

Training a Kohonen Feature Map

```
function mykfm;
%By L.S.Balasuriya
%Based on original by Jang et al.(1997)
%Training a KFM

% Initial Parameters ************************************************
side = 10;                  % size of network
pattern_n = 30;     % pattern_n is the number of input patterns
iteration_n = 100; % no. of iterations in training


%Input Data ********************************************************
x = rand(pattern_n, 1);
x=x+ sqrt(-1)*x;

%Rectangle
ix = 1:pattern_n;
x = x(ix);

%SOM learning parameters ******************************************
init_eta = 0.8;                 % Initial value of eta(learning rate)
final_eta = 0.1;                %Final value of eta
init_sigma = side/2;        % Initial value of sigma(neighbourhood)
final_sigma = 1;                %Final value of sigma

% Initial weights matrix *******************************************
w = (rand(side) + j*rand(side))/10 + 0.45*(1+j) ;      %weight matrix


%Input data distribution ******************************************
snapshotH = genfig('Snap shots of Kohonen feature map');
clf;
subplot(2,4,1); plot(x, '.'); axis square; axis([0 1 0 1]);
title('Data distribution');
title('(a)');
subplot(2,4,2); plot([w w.']); axis square; axis([0 1 0 1]);
title('Initial weights');
title('(b)');
```

```
% Initial animation objects ****************************************
animationH=genfig('Animation for Kohonen feature map');
lineH = plot(nan*real([w w.']), nan*imag([w, w.']), 'erase', 'back');
axis square; axis([0 1 0 1]);
set(gca, 'xtick', [], 'ytick', []);
titleH = text(0.5, 1.05, '');
set(titleH, 'erase', 'xor', 'horizontal', 'center');
xlabelH = text(0.5, -0.05, '');
set(xlabelH, 'erase', 'xor', 'horizontal', 'center');
if matlabv==4,
    inputH = line(nan, nan, 'erase', 'back', 'linestyle', '.', ...
        'markersize', 25, 'color', 'c');
elseif matlabv==5,
    inputH = line(nan, nan, 'erase', 'back', 'marker', '.', ...
        'markersize', 25, 'color', 'c');
else
    error('Unknown MATLAB version');
end
titleH = get(gca, 'title');
xlabelH = get(gca, 'xlabel');
[xx, yy] = meshgrid(1:side, 1:side);


% ====== main loop ====================================================
for k = 1:iteration_n
    eta=init_eta+(k-1)*(final_eta-init_eta)/(iteration_n-1);
    sigma=init_sigma+(k-1)*(final_sigma-init_sigma)/(iteration_n-1);

    input = x(rem(k, pattern_n)+1); % a value from x (sequentially step k)
    xw = input - w;                 %Euclidean distance matrix (from input)

    % (IM,JM) is the coordinates of the winning unit.
    dist = abs(xw);     % magnitude(matrix) of the complex elements of xw.
    min_dist = min(min(dist));   % minimum Euclidean distance to weight
    [IM, JM] = find(dist==min_dist); % coordinate of winner

    %The neighbourhood function NB is centered around (IM, JM)
    TMP = xx-IM + (yy-JM)*sqrt(-1);
    NB = exp(-TMP.*conj(TMP)/sigma);
    NB = NB.';

    %update weights of the winning unit and its neighbourhood
    w = w + eta*NB.*xw;

    %update animation objects
    tmp = [w, w.'];
    for i = 1:size(lineH),
        set(lineH(i), 'xdata', real(tmp(:, i)), ...
            'ydata', imag(tmp(:, i)));
    end
    set(inputH, 'xdata', real(input), 'ydata', imag(input));
    title_str = ['eta = ',num2str(eta),'  sigma = ',num2str(sigma)];
    count_str = ['count = ',int2str(k),'/',int2str(iteration_n)];
    set(titleH, 'string', title_str);
    set(xlabelH, 'string', count_str);
```

```
    drawnow;

    % ====== snapshots
    if k == 30,
        figure(snapshotH);
        subplot(2,4,3); plot([w w.']);
        axis square; axis([0 1 0 1]);
        title([Wts. after ' int2str(k) ' updates']);
        title('(c)');
        drawnow;
        figure(animationH);
    elseif k == iteration_n,
        figure(snapshotH);
        subplot(2,4,4); plot([w w.']);
        axis square; axis([0 1 0 1]);
        title([Wts. after ' int2str(k) ' updates']);
        title('(d)');
        drawnow;
    end
end

%=================================================================
```

Training a Kohonen Feature Map with Input Space Neighbourhood

```
function mykfm2;
%By L.S.Balasuriya
%Traing a KFM with a input space neighbourhood

% Initial Parameters ************************************************
side = 20;
pattern_n = 20;    % pattern_n is the number of input patterns
iteration_n = 500;

%Input Data ********************************************************
x(1:pattern_n) = rand(pattern_n, 1) + sqrt(-1)*rand(pattern_n, 1);

%SOM learning parameters *******************************************
init_eta = 0.5;          % Initial value of eta(learning rate)
final_eta = 0.1;         %Final value of eta
init_sigma = 1/10;       % Initial value of sigma(neighbourhood)
final_sigma = 0;         %Final value of sigma

% Initial weights matrix ********************************************
[wi, wj] = meshgrid(1:side, 1:side);
w=(wi/side)-(0.5/side)+sqrt(-1)*((wj/side)-(0.5/side));
```

```
%Intial Plots**********************************************************

figure;
plot(x, 'o'); axis square; axis([0 1 0 1]);
title('Data distribution');

figure;
plot([w w.']); axis square; axis([0 1 0 1]);
title('Initial weights');


%Main loop ============================================================
for k = 1:iteration_n
      eta=init_eta+(k-1)*(final_eta-init_eta)/(iteration_n-1);        %reduce%learning rate
   sigma=init_sigma+(k-1)*(final_sigma-init_sigma)/(iteration_n-1);
   % reduce neighbourhood

   input = x(rem(k, pattern_n)+1); % a value from x (sequentially step k)
   xw = input - w;                              %Euclidean distance matrix (from input)

   %(IM,JM) is the coordinate of the winning unit.
   dist = abs(xw); % magnitude(matrix) of the complex elements of xw.

   for p=1:side
      for q=1:side
            guass=exp(-(dist(p,q)^2)/(2*(sigma^2)));   %guassian function
            w(p,q)=w(p,q)+eta*guass*xw(p,q);  %  updated weighhts
      end
   end


%snap shots ***********************************************
if k==iteration_n,
   figure;
   plot(w,'*');axis square; axis([0 1 0 1]);
   title(['after ' int2str(k) ' iterations']);
else if 0 == mod((k/iteration_n)*100,10),
           figure;
           plot(w,'.'); %plot([w w.']);
           axis square; axis([0 1 0 1]);
           title(['after ' int2str(k) ' iterations']);
   end
end
end
% ======================================================================
```

Training a Input Space Neighbourhood Kohonen Feature map with node Sensitivity

```
function [w,s]=mykfm21;
%Returns final weights(s) and sensitivity matrix(s) of training
%KFM with a input space neighbourhood and node sensitivity

% Initial Parameters ************************************************
side = 10;
pattern_n = 12;     % pattern_n is the number of input patterns
iteration_n = 500;


%Input Data ********************************************************
x(1:pattern_n) = rand(pattern_n, 1) + sqrt(-1)*rand(pattern_n, 1);

%SOM learning parameters *******************************************
init_eta = 0.12;                % Initial value of eta
final_eta = 0.01;          %Final value of eta
init_sigma = 1/12;              % Initial value of sigma
final_sigma = 1/side;   %Final value of sigma
theta = 0.5;                    % affect of sensitivity matrix
timefactor = 1.05;

% Initial weights matrix ********************************************
[wi, wj] = meshgrid(1:side, 1:side);
w=(wi/side)-(0.5/side)+sqrt(-1)*((wj/side)-(0.5/side));

% Initial sensitivity matrix ****************************************
s=ones(side,side);

% Initial plots *****************************************************
figure;
plot(x, 'o'); axis square; axis([0 1 0 1]);
title('Data distribution');

figure;
plot([w w.']); axis square; axis([0 1 0 1]);
title('Initial weights');


%Main loop ========================================================
for k = 1:iteration_n
   eta=init_eta+((k-1).*(k-1).*(k-1))*...
```

```
    (final_eta-init_eta)/(iteration_n-1)^3;      % reduce learning rate
  sigma=init_sigma+((k-1).*(k-1).*(k-1))...
    *(final_sigma-init_sigma)/(iteration_n-1)^3; % reduce neighbourhood

  input = x(rem(k, pattern_n)+1); % a value from x (sequentially step k)
  xw = input - w;                 %Euclidean distance matrix (from input)

  % (IM,JM) is the coordinate of the winning unit.
  dist = abs(xw); % magnitude(matrix) of the complex elements of xw.




  for p=1:side
    for q=1:side
    guass=exp(-(dist(p,q)^2)/(2*((sigma+s(p,q))^2)));%guassian function
    movement=eta*guass*xw(p,q);          % delta weights
    w(p,q)=w(p,q)+movement;              % hopefully the updated weights
    s(p,q)=s(p,q)*(1-guass);
    end
  end

  s=s*timefactor;



%snapshots  ********************************************************
  if mod(k,fix(iteration_n/8))==0,
    figure;
    plot(w,'.'); axis square; axis([0 1 0 1]);
    title(['after ' int2str(k) ' iterations']);
  elseif k == iteration_n,
    figure;
    plot(w,'b*'); axis square; axis([0 1 0 1]);
    title(['after ' int2str(k) ' iterations']);
  end
end

%end Main loop ============================================================
```

Face Detection

```
function coordinates=face_detection(input_image);
%By L.S.Balasuriya
%Find face in input_image and return its location

%********load templates ************************************************
load dark_weight9; dark_weights=dark_weight9;
```

```
load light_weight9; light_weights=light_weight9;

load gray255map;

%********size of weight matrix ***************************************
[ignor,gridsize]=size(dark_weights);

%********height and width of input image ******************************
[height,width]=size(input_image);

%********Initial parameters *****************************************
int_image=im2double(input_image);

max_face=0.7;                    %max percentage size of face in input image
min_face=0.2;                    % min percentage size of face in input image

queue=0;t=1;                     %queue for faceness values

big=min(height,width);          %get largest value
max=round(big*max_face);% max pixel width/height of face in input image
max=max+mod(max,2);              %make even for facedec
min=round(big*min_face);  % min pixel width/height of face in input image

increment= 0.2;                  %increment of moving window
jump= round(width*0.015);  %increment in pixels
jump= jump+mod(jump,2); %make even for facedec


%********Initial graphics handles & display initial image **************
colormap(gray255map);
image(input_image);              %image is now displayed properly
hold on;                         %make graphics go on top of image

%********Initial rectangle position *************************************
y=0; x=0;
r=rectangle('Position',[x,y,max,max]);
set(r,'EraseMode','xor');


%********End Initial graphics handles ***********************************

detection= -10;                  %highest value for faceness that have obtained
faceness= -10;                   % value returned by facedec
winsize=max;                     %biggest winsize (max)


%********Search for the line of symmetry ********************************
y=1;      % top y (1)
x=1;      % left x (1)

while x+winsize<width & faceness<lazy_threshold
    set(r,'Position',[x,y,winsize,winsize]);
    drawnow;                                    % Update window position
    faceness=symmetry(int_image,x,y,winsize);
    queue(t)=faceness;t=t+1;
```

```
    if faceness>detection                    % try to get the best window
        detection=faceness;
        coordinates=[detection,x,y,winsize];
    end
    x=x+jump;                                %next x
end


%********End of the search for the line of symmetry ******************


symmetry=round(coordinates(2)+coordinates(4)/2);
detection=0;
faceness=0;




%********Search for Best face ***************************************

winsize=max; %biggest winsize (max)

while (winsize>=min)
    y=1;   % top y (1)

    %********make big/small to fit window size
    %********remember weights point to locations in window
    dark_positions=(dark_weights*winsize);
    light_positions=(light_weights*winsize);

    %********turn weight positions into matrix indicies
    index_dark_weights=round(real(dark_positions)-1)...
        *winsize+round(imag(dark_positions));
    index_light_weights=round(real(light_positions)-1)...
        *winsize+round(imag(light_positions));

    x=symmetry-winsize/2;

    while y+winsize<height & faceness<lazy_threshold
        set(r,'Position',[x,y,winsize,winsize]);
        drawnow;                             % Update window position
        faceness=facedec(int_image,x,y,winsize,...
            index_dark_weights,index_light_weights);
        queue(t)=faceness;t=t+1;
        if faceness>detection            % try to get the best window
            detection=faceness;
            coordinates=[detection,x,y,winsize];
        end
        y=y+jump;                                %next y
    end
    queue(t)=0;t=t+1;
    winsize=winsize-jump;                    %next winsize
```

```
end

%********end of the Search for Best face ******************************


%********Draw best faceness window
set(r,'Position',[coordinates(2),coordinates(3),coordinates(4),coordinates(4)]);
drawnow;

%=================================================================
```

Return Heuristic for Pixel Area's Vertical Symmetry

```
function sym=symmetry(int_image,x,y,winsize);
%by L.S.Balasuriya
%return heuristic for vertical symmetry

%********segment face detection window *********************************
window=int_image(y:y+winsize,x:x+winsize);

%********increase difference between highs.lows and normalise **********
norm_window=window-min(min(window));
norm_window=norm_window/max(max(window));

%********check vertical symmetry ************************************
left=window(1:winsize,round(winsize/2):-1:1);
right=window(1:winsize,round(winsize/2)+1:winsize);
up=window(round(winsize/2):-1:1,1:winsize);
down=window(round(winsize/2)+1:winsize,1:winsize);
c=corrcoef(left,right)-corrcoef(up,down);
sym=c(1,2);
```

Return Heuristic for Pixel Area's 'Faceness'

```
function faceness=facedec(int_image,x,y,winsize,...
    index_dark_weights,index_light_weights);
```

```
%By L.S.Balasuriya
%return 'faceness' of the face detection window in int_image
%dark weights and light weights are the dark and
%bright templates respectively
%x,y,winsize are details about the location and
%size of the face detection window

%********segment face detection window **********************************
window=int_image(y:y-1+winsize,x:x-1+winsize);

%********normalise
norm_window=window/max(max(window));

%********add intensities of pixels indicated by dark and bright templates
darkness=sum(sum(1-window(index_dark_weights)));
lightness=sum(sum(window(index_light_weights)));

%********faceness value ************************************************
faceness=(lightness+darkness);
```

Principal Component Analysis

```
function [U,R,E] = pcabigFn(X);
%By L.S. Balasuriya
%Achnowledgements to Doug Hundley
%Based on original by Marian S. Bartlett.
%Computes principal component analysis of mean subtracted matrix X
% Returns eigenfaces in U
%Facespace vectors of X in R
%Eigenvalues of Eigenvectors in E

%********get size of matrix *****************************************
[N,P] = size(X);

%********subtract mean **********************************************
mb=mean(X');
X=X-(ones(P,1)*mb)';

%********Find eigenvectors of B'B ***********************************
[V,D] = eig (1/(P-1)*(X'*X));


%********Sort eigenvectors  *****************************************
eigvalvec = max(D);
[seigvals, index] = sort(eigvalvec);
Vsort = V(:,[fliplr(index)]);

%********Deduce eigen vectors of BB' ********************************
U = X*Vsort;
```

```
%********Give eigvecs unit length. **********************************
length = sqrt (sum (U.^2));
U = U ./ (ones(N,1) * length);


R = B'*U;
E = fliplr(seigvals);
```

Face Recognition

```
function closeness=recognition(input_image,U,R);
%By L.S. Balasuriya
%Returns closeness of known face vectors in R to unknown image input_image
%R contains the Eigenfaces


%********image to vector ***********************************************
vinput=reshape(input,[10000 1]);


%********recognition ***************************************************
facespace=voutput'*U;


%********Eucleadian distance *******************************************
[p,ignor]=size(R);
distance_vecs=R-repmat(facespace,[p 1]);
distance=sum(abs(distance_vecs)')';

%********order of closeness to unknown face ***************************
[ignor,closeness]=sort(distance);
```

Manual Face Detection

```
function output_image=manual_facedec(input_image);
%By L.S.Balasuriya
%Returns segmented pixel area from input_image containing the face
image(input_image);

%********straighten face ***********************************************
%********operator clicks under eyes ************************************
[Lx,Ly]=nngetclk('Click lowest edge of Left Eye');
[Rx,Ry]=nngetclk('Click lowest edge of Right Eye');

width=(Rx-Lx)/0.6;
y=(Ly+Ry)/2;

left=round(Lx-width*0.4);
```

```
top=round(y-0.54*width);
width=round(width*1.4);

%area near face segmented ***********************************************
input_image=input_image(top:top+width,left:left+width,:);

%image rotated to straighten eyes and zoomed in ************************
angle=atan((Ly-Ry)/(Rx-Lx))*180/pi;
input_image=imrotate(input_image,-angle,'crop');

image(input_image);

%********manual face detection ***************************************
%********operator clicks under eyes *********************************
[Lx,Ly]=nngetclk('Click lowest edge of Left Eye');
[Rx,Ry]=nngetclk('Click lowest edge of Right Eye');

width=(Rx-Lx)/0.6;
y=(Ly+Ry)/2;

left=round(Lx-width*0.2);
top=round(y-0.27*width);
width=round(width);

%image turned into gray scale ********************************************
input_image=rgb2gray(input_image);

%face segmented and displayed ********************************************
output_image=input_image(top:top+width,left:left+width);
output_image=imresize(output_image,[100 100],'bicubic');
imagesc((output_image));
colormap(jet); colorbar;
```

Appendix C - The most beautiful girl in the world?

One of the interesting calculations of a face recognition system based on Principal Component Analysis is the average human face. Because there are several theories on the association between the average face and facial attractiveness, the author decided to investigate further.

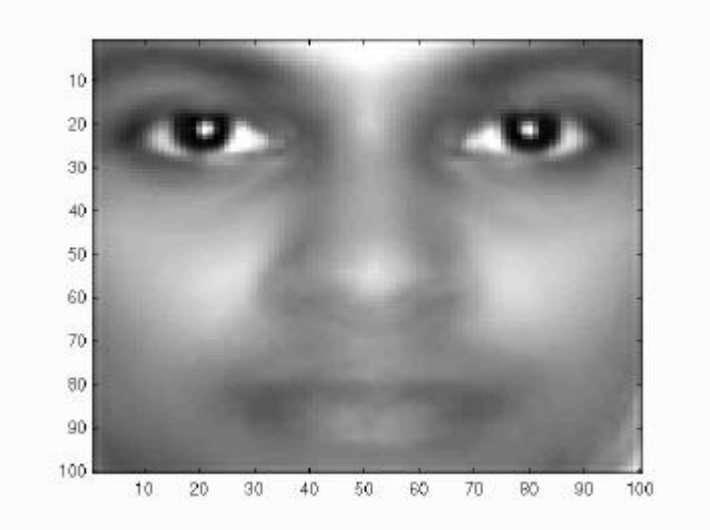


Figure 9.1 Average face from 30 test subjects (from both male and female subjects)

A study was conducted by Francis Galton in the 1870's to find out what the typical criminal looked like. Mugshots of hideous criminals were photographically blended to produce an average criminal's face. Much to Galton's surprise the face he produced was of a good-looking gentleman.

Over a century later, the most widely held theory about Galton's findings is that attractiveness is largely a matter of being close to the average (AAFPRS, 1997). While this may seem quite strange, it must be realised that having a face close to the average face is rare, since almost no one will have average eyes *and* an average nose *and* an average mouth etc. Almost all of us have some feature that is not average (for example, a long face). Averageness being attractive makes sense when one realises that being abnormal and having abnormal genes is usually harmful.

Besides an average face, men also seem to instinctively prefer women with full lips and a small jaw since these are a sign of high estrogen production and women prefer men with a strong jaw and a wide chin-as these are the result of high testosterone production. Further more, since the average faces each of us has assembled differs slightly from one another, our attraction to a certain individual may vary.

It seems that beauty really is in the eye of the beholder.

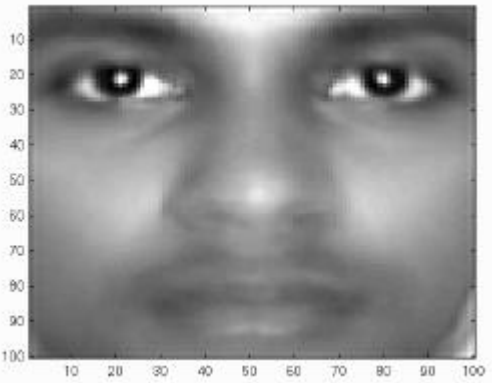


Figure 9.2 Average face from 18 male test subjects

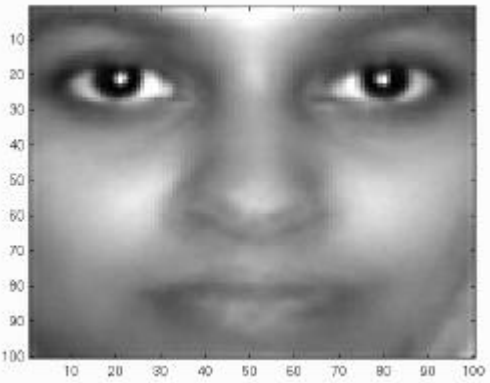


Figure 9.3 Average face from 12 female test subjects

BIBLIOGRAPHY

Adelson, E. H., and Bergen, J. R. (1986*) The Extraction of Spatio-Temporal Energy in Human and Machine Vision*, Proceedings of Workshop on Motion: Representation and Analysis (pp. 151-155) Charleston, SC; May 7-9

AAFPRS(1997). *A newsletter from the American Academy of Facial Plastic and Reconstructive Surgery*. Third Quarter 1997, Vol. 11, No. 3. Page 3.

Baron, R. J. (1981). *Mechanisms of human facial recognition.* International Journal of Man Machine Studies, 15:137-178

Beymer, D. and Poggio, T. (1995) *Face Recognition From One Example View*, A.I. Memo No. 1536, C.B.C.L. Paper No. 121. MIT

Bichsel, M. (1991*). Strategies of Robust Objects Recognition for Automatic Identification of Human Faces.* PhD thesis, , Eidgenossischen Technischen Hochschule, Zurich.

Brennan, S. E. (1982) *The caricature generator*. M.S. Thesis. MIT.

Brunelli, R. and Poggio, T. (1993), *Face Recognition: Features versus Templates*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052

Craw, I., Ellis, H., and Lishman, J.R. (1987). *Automatic extraction of face features*. Pattern Recognition Letters, 5:183-187, February.

Deffenbacher K.A., Johanson J., and O'Toole A.J. (1998*) Facial ageing, attractiveness, and distinctiveness*. Perception. 27(10):1233-1243

Dunteman, G.H. (1989) *Principal Component Analysis*. Sage Publications.

Frank, H. and Althoen, S. (1994). *Statistics: Concepts and applications*. Cambridge University Press. p.110

Gauthier, I., Behrmann, M. and Tarr, M. (1999). *Can face recognition really be dissociated from object recognition?* Journal of Cognitive Neuroscience, in press.

Goldstein, A.J., Harmon, L.D., and Lesk, A.B. (1971). *Identification of human faces*. In Proc. IEEE, Vol. 59, page 748

de Haan, M., Johnson, M.H. and Maurer D. (1998) *Recognition of individual faces and average face prototypes by 1- and 3- month-old infants*. Centre for Brain and Cognitive Development, Department of Psychology, Birkbeck College.

Hadamard, J. (1923) *Lectures on the Cauchy Problem in Linear Partial Differential Equations* , Yale University Press

Haralick, R.M. and Shapiro, L.G.. (1992) *Computer and Robot Vision,* Volume I. Addison-Wesley

Haxby, J.V., Ungerleider, L.G., Horwitz, B., Maisog, J.M., Rapoport, S.I., and Grady, C.L. (1996). *Face encoding and recognition in the human brain. Proc. Nat. Acad. Sci.* 93: 922 - 927.

Heisele, B. and Poggio, T. (1999) *Face Detection*. Artificial Intelligence Laboratory. MIT.

Jang., J., Sun, C., and Mizutani, E. (1997) *Neuro-Fuzzy and Soft Computing*. Prentice Hall.

Johnson, R.A., and Wichern, D.W. (1992) *Applied Multivariate Statistical Analysis*. Prentice Hall. p356-395

Kanade, T. (1973) *Picture processing by computer complex and recognition of human faces*. Technical report, Kyoto University, Dept. of Information Science.

Kaya, Y. and Kobayashi, K. (1972) *A basic study on human face recognition*. In S. Watanabe, editor, Frontiers of Pattern Recognition, page 265.

Kohonen, T. (1995), *Self-Organizing Maps*, Berlin: Springer-Verlag, p. VII

Krose, B., and van der Smagt, P. (1996), *An Introduction to Neural Networks*. The University of Amsterdam.

Langlois, J.H. and Roggman, L.A. (1990).*Attractive faces are only average*. Psychological Science, 1(2) 115-121.

Lee, W. and Thalmann, N.M. (1998) *Head Modelling from Pictures and Morphing in 3D with Image Metamorphosis based on triangulation*. MIRALab, CUI, University of Geneva.

Moscovitch, M., Winocur, G. and Behrmann, M. (1997) *What is special about face recognition? Nineteen experiments on a person with visual object agnosia and dyslexia but normal face recognition*. Journal of Cognitive Neuroscience, 9, 5, 555-604.

Narayanaswamy, C.R. and Raghavarao, D. (1991) *Principal component analysis of large dispersion matrices.* APSTAG 40:2. pp309-316.

Osuna, E., Freund, R. and Girosiy, F. (1997*) Training Support Vector Machines: an Application to Face Detection.* Proceedings of CVPR, Puerto Rico.

O'Toole, A.J., Abdi, H., Deffenbacher, K.A., and Valentin, D. (1993) *A low-dimensional representation of faces in the higher dimensions of the space*. Journal of the Optical Society of America A, 10, 405-411.

Rodman, H.R., Gross, C.G., and Scalaidhe, S.P.Ó. (1993) Development of brain substrates for pattern recognition in primates: physiological and connectional studies of inferior

temporal cortex in infant monkeys. In B. de Boysson-Bardies, S. de Schonen, P. Jusczyk, P. MacNeilage, and J. Morton, Eds., *Developmental Neurocognition: Speech and Face Processing in The First Year of Life.* Dordrecht: Kluwer Academic, pp. 63 - 75.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. New York: Spartan.

Rowley, H., Baluja, S. and Kanade, T. (1996) *Neural Network-Based Face Detection*. Computer Vision and Pattern Recognition.

Saber, E. and Tekalp, A., (1996). *Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry Based Cost Functions*, International Conference on Pattern Recognition (C8E.5)

Sinha, P. (1994) *Object Recognition via Image Invariants*: A Case Study. In Investigative Ophthalmology and Visual Science, volume 35, pages 1735-1740, Sarasota, Florida, May.

Sung, K. (1995) *Learning and Example Selection for Object and Pattern Detection*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA

Sung, K. and Poggio, T. (1994) *Example-based learning for view-based human face detection*. In Proceedings from Image Understanding Workshop, Monterey, CA

Turk, M. and Pentland, A. (1991a). *Eigenfaces for recognition*. Journal of Cognitive Neuroscience, 3(1), 71-86.

Turk, M. and Pentland, A. (1991b). *Face recognition using eigenfaces*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June Maui, Hawaii, p 586-591

Wang, J.Y.A., and Adelson, E.H. (1994*) Spatio-Temporal Segmentation of Video Data* Proceedings of SPIE on Image and Video Processing II, 2182:120-131 San Jose; February.

Wiskott et al. (1997) Wiskott, L., Fellous, J., Krüger, N., and Malsburg, C. *Face Recognition by Elastic Bunch Graph Matching*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):775-779

Yuille, A., Hallinan P., and Cohen ,D. (1992) *Feature Extraction from Faces using Deformable Templates*. International Journal of Computer Vision, 8(2):99-111

Zeki, S. (1993). *A vision of the brain.* Oxford: Blackwell.